SW-640L-TX-W

API Reference Manual

Application Programming Interface

Document Revision	v1.0
Document Date	August 2024
Supported Firmware	Firmware version v1.1.4 or later

Contents

1. API Requests Overview

1.1 Classification

All APIs can be divided into the following categories according to their functions:

1. **Device management**
   Manage the device name, reboot or reset device, etc...
2. **Input & output management**
   Manage input EDID, HDCP, output resolution, etc...
3. **Video switching**
   Many different approaches to switch video source
4. **Audio & auxiliary signal switching**
   Route audio signal and switch USB host
5. **Standby related & peripheral control**
   Control the device to standby and wake-up, send CEC and RS232 message
6. **Network & BYOD**
   Manage IP address and BYOD feature

1.2 Format or style

API commands can be mainly divided into the following categories:

- **gbconfig**: manage the configurations of the device
- **gbcontrol**: control the device to do something
- **gblayout**: adjust the features related to screen layout

In addition, some APIs support two different formats. The format or style of the newly added APIs is compatible with the ProAV presentation switcher, namely, these APIs start with `SET` or `GET`. Currently, this situation mainly occurs in some APIs related to video source switching. Based on this design, users only need to master one set of APIs to control the device to switch source.

1.3 Terms & conventions

1. The italic string in the message format indicates that the string should be replaced by the actual parameter in the actual message, and the italic string itself should not appear in the final message body. For example *`DeviceName`* Indicates the device name. The actual device name should be given in the actual message body.

2. Square brackets indicate optional parameters, for example `[WinNo]` Indicates an optional window number parameter. Obviously, optional parameters can only appear at the end of the message.

3. The combination of curly braces and vertical bars is used to specify the possible values for the parameter, for example `{in1 | in2 | in3 | in4}` Represents a parameter whose value can be one of `in1` to `in4`.

4. The combination of square brackets and vertical bars is used to specify the possible values for optional parameters, for example `[start | stop]` Indicates an optional parameter whose value can be `start` or `stop`.

5. Definition of common error response messages: `ERROR` If any error occurs when executing the API, the device can print specific prompt information or return this same message to indicate an execution error. The subsequent chapters will not explain it separately.

6. Some commonly used parameters

- Video (source) name
  `VideoName`
  Valid values are as follows:
  `usbc1, usbc2, hdmi3, hdmi4,`
  `airplay1, airplay2, airplay3, airplay4,`
  `miracast1, miracast2, miracast3, miracast4,`
  `dongle1, dongle2, dongle3, dongle4,`
  `chromecast1, chromecast2, chromecast3, chromecast4.`
  The four values in the above row 1 represent four wired sources, they correspond to the USB-C IN1, USB-C IN2, HDMI IN3, HDMI IN4 interface respectively. Most APIs use this parameter *`VideoName`* to refer to a video source.

- **Guide screen**
  `guide`
  A special video source which means to show guide screen.

- **Wired source**
  `WiredSrc`
  Valid values are `in1`, `in2`, `in3` and `in4`, they are equivalent to the previous *`VideoName`* values `usbc1`, `usbc2`, `hdmi3` and `hadmi4`.

- **BYOD source**
  `ByodSrc`
  Valid values are `byod1`, `byod2`, `byod3` and `byod4`, indicating one of four BYOD sources.

- **All input source**
  `Src`
  Valid values are `in1`, `in2`, `in3`, `in4`, `byod1`, `byod2`, `byod3` and `byod4`, represents one of four wired sources and four BYOD sources.

As mentioned in the previous chapter, the newly added APIs have different styles, they use `WiredSrc`, `ByodSrc` and `Src` to refer to a video source. Please note this difference when reading the subsequent chapters.

2. Command Details

2.1 Device management

2.1.1 Configure device name

Configure the device’s name. As a prompt, the new name will appear on the top-right corner of the screen if the operation is successful. As the factory default, the device name is the same as the device’s model.
The device name must be 1~20 characters in length, furthermore, it must include only letters, numbers, space and two special characters ('_' and '-').

Configure device name	
Command structure: `gbconfig --name` *`DeviceName`*	*Note:* Instead of *`DeviceName`*, the actual device name should be given in the message body.
Response structure: `(No text response message)`	
Command Example: `gbconfig --name MeetingRoom`	
Response Example: `(None)`	

2.1.2 Query device name

Obtain the device name.

Query device name	
Command structure: `gbconfig -s name`	*Note:* Instead of *`DeviceName`*, the actual device name is shown.
Response structure: *`DeviceName`*	
Command Example: `gbconfig -s name MeetingRoom`	
Response Example: `MeetingRoom`	

2.1.3 Query device information

Obtain the information about the device model and firmware version.

Query device information	
Command structure: `gbconfig -s device-info`	*Note:*
Response structure: `The device prints its model and firmware version.`	
Command Example: `gbconfig -s device-info`	

Response Example:
`MS340-A00`
`V1.2.0`	

2.1.4 Reboot device

Reboot the device manually.

Reboot device	
Command structure: `gbcontrol --reboot`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbcontrol --reboot`	
Response Example: `(None)`	

2.1.5 Reset device factory default

This command makes the device restore its factory default. To achieve this, the device will reboot to enter recovery or safe mode.

Reset device factory default	
Command structure: `gbcontrol --reset-to-default [guide]`	*Note:* If the request message carries the optional parameter `guide`, it means to restore the default guide picture only.
Response structure: `(No text response message)`	
Command Example: `gbcontrol --reset-to-default`	
Response Example: `(None)`	

2.2 Input & output management

2.2.1 Configure output resolution

The device will change its output resolution as the command designates or automatically. The first parameter indicates which output resolution is set. If you assign `auto` as the `Resolution` parameter, the device will select the best resolution according to the display's EDID.

The list of all available timings is below:
3840x2160P@60* 3840x2160P@50* 3840x2160P@30 3840x2160P@25 3840x2160P@24 1920x1080P@60 1920x1080P@50 1920x1080P@30 1920x1080P@25 1920x1080P@24 1680x1050P@60 1600x1200P@60 1440x900P@60 1366x768P@60 1280x1024P@60 1280x720P@60 1280x720P@50 1024x768P@60 800x600P@60 720x480P@60 and 640x480P@60
*Available for `out1` only.
As the factory default, both two outputs' resolutions are set to `auto`.

Configure output resolution	
Command structure: `gbconfig --output-resolution {out1 | out2} {Resolution | auto}`	*Note:* For `{out1 | out2}` use `out1` or `out2`.
Response structure: `(No text response message)`	
Command Example: `gbconfig --output-resolution out1 3840x2160P@60`	
Response Example: `(None)`	

2.2.2 Query output resolution

The device prints the resolution information of two outputs in two rows. For each row, the first field is the number of the output, the second field is the actual resolution. The third field `auto` is optional, which means the corresponding output resolution is set to `auto`. The term `disconnected` indicates that the corresponding output resolution is set to `auto` but the output is not connected to a display.

Query output resolution	
Command structure: `gbconfig -s output-resolution`	*Note:*

Response structure:
`out1 {Resolution | disconnected} [auto]`
`out2 {Resolution | disconnected} [auto]`	

Command Example: `gbconfig -s output-resolution`	

Response Example:
`out1 3840x2160@60 auto`
`out2 disconnected auto`	

2.2.3 Configure output HDCP

Configure whether the HDCP feature is enabled on the outputs. If it is enabled, HDCP 2.2 will be used on the output when the connected display supports HDCP 2.2, or else HDCP 1.4 will be used. If this feature is disabled, the output content will not be protected by HDCP. As the factory default, the HDCP feature is enabled, and the corresponding configuration is `y`.

Configure output HDCP	
Command structure: `gbconfig --hdcp-enable {y | n}`	*Note:* For `{y	n}` use `y` or `n`.
Response structure: `(No text response message)`	
Command Example: `gbconfig --hdcp-enable n`	
Response Example: `(None)`	

2.2.4 Query output HDCP

Query output HDCP	
Command structure: `gbconfig -s hdcp-enable {y	n}`	*Note:* For `{y	n}` use `y` or `n`.
Response structure: `{y	n}`	
Command Example: `gbconfig -s hdcp-enable`	
Response Example: `n`	

2.2.5 Query input connection status

If the parameter designates a specific wired source, it means to query whether the specific input is connected to a source, the device prints `connected` to indicate the source is connected, and vice versa.

If the parameter value is `all`, the device responds with four rows of texts to report the connection status of all wired sources.

Query input connection status	
Command structure: `GET VIDIN_CONNECT {`*`WiredSrc`*`	all}`	*Note:* For `WiredSrc` valid values are `in1, in2, in3` and `in4,` they are equivalent to the *`VideoName`* values `usbc1, usbc2, hdmi3` and `hdmi4.` For more information read **Terms & conventions** section.
Response structure: `VIDIN_CONNECT `*`WiredSrc`*` {disconnected	connected}` `or` `VIDIN_CONNECT in1 {disconnected	connected}` `VIDIN_CONNECT in2 {disconnected	connected}` `VIDIN_CONNECT in3 {disconnected	connected}` `VIDIN_CONNECT in4 {disconnected	connected}`	
Command Example: `GET VIDIN_CONNECT in2`	
Response Example: `VIDIN_CONNECT in2 disconnected`	

2.2.6 Report input connection status

When the connection status of a certain input changes, the device actively sends the response message. Its format is exactly the same as the response message of the previous command to query the status of a specific input source.

Report input connection status	
Command structure: `This API has no command message`	*Note:*
Response structure: `VIDIN_CONNECT` *`WiredSrc`* `{disconnected | connected}`	
Command Example: `(None)`	
Response Example: `VIDIN_CONNECT in2 connected`	

2.2.7 Query input signal status

If the parameter designates a specific source, it queries whether the specific source has a valid signal. The device prints `valid` to indicate the source has a valid signal, and vice versa. If the parameter value is `all`, the device responds with eight rows of texts to report the signal status of all sources.

Query input signal status	
Command structure: `GET VIDIN_SIG {Src | all}`	*Note:* For `Src` valid values are `in1, in2, in3, in4, byod1, byod2, byod3` and `byod4.` For more information read **Terms & conventions** section.

Response structure:
`VIDIN_SIG Src { no | valid }`
`or`
`VIDIN_SIG in1 {no | valid}`
`VIDIN_SIG in2 {no | valid}`
`VIDIN_SIG in3 {no | valid}`
`VIDIN_SIG in4 {no | valid}`
`VIDIN_SIG byod1 {no | valid}`
`VIDIN_SIG byod2 {no | valid}`
`VIDIN_SIG byod3 {no | valid}`
`VIDIN_SIG byod4 {no | valid}`	

Command Example: `GET VIDIN_SIG in2`	
Response Example: `VIDIN_SIG in2 valid`	

2.2.8 Report input signal status

When the signal status of a certain input changes, the device actively sends the response message. Its format is exactly the same as the response message of the previous command to query the status of a specific input source.

Report input signal status	
Command structure: `This API has no command message.`	*Note:*
Response structure: `VIDIN_SIG Src {no | valid}`	
Command Example: `(None)`	
Response Example: `VIDIN_SIG in2 valid`	

2.3 Video switching

2.3.1 Configure automatic switching

Configure the automatic switching feature. When it is enabled, the device supports the following functions:

1. **Automatically switch wired source**

- When a wired source is connected, starts displaying the video source automatically
- When a wired source is disconnected, stops displaying the video source automatically

Note: The setting of whether automatically switching the video source only works for wired sources. Due to the nature of BYOD sources (no persistent physical interface), the device always automatically switches to BYOD sources.

2. **Automatically switch layout**

If the multiview feature is enabled, when only a display is connected to the device, the device also can:

- When starting to display a video source (due to automatic switching or API triggering), if there is no idle child window (not occupied by any video source) in the current layout, try switching to a layout with more child windows.
- When stopping to display a video source (due to automatic switching or API triggering), if a multiview layout is being used, try switching to a layout with fewer child windows.

3. **Rollback switching**

When the device is in full-screen (multiview layout is unused) state, if the currently displayed video source is automatically stopped, it will automatically switch to display the last displayed video source. This feature is only for automatic switching scenarios. For API commands that have the function of "stopping displaying a certain video source," rollback switching is NOT performed. The reason is: rollback switching mechanism is designed to improve user experience in automatic switching scenarios, but it cannot accurately meet customer needs. When users use the API in a certain way, they can accurately realize their intentions through subsequent API calls, and rollback switching is unnecessary.

As the factory default, automatic switching is enabled; the configuration value is set to `y`.

Configure automatic switching	
Command structure: `gbconfig --auto-switch {y | n}`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --auto-switch n`	
Response Example: `(None)`	

2.3.2 Query automatic switching

Query the state of the automatic switching configuration.

Query automatic switching	
Command structure: `gbconfig -s auto-switch`	*Note:*
Response structure: `{y | n}`	
Command Example: `gbconfig -s auto-switch`	
Response Example: `n`	

2.3.3 Configure multiview feature

Configure whether the multiview feature is enabled. When disabled, the device displays the video source with full screen or single view mode.

Note: Even if the configuration is enabled, the multiview feature can be actually activated only when only one display is connected to the device.

As the factory default, multiview feature is enabled; the configuration value is set to `y`.

Configure multiview feature	
Command structure: `gbconfig --multiview {y | n}`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --multiview n`	
Response Example: `(None)`	

2.3.4 Query multiview feature

Query the state of the multiview configuration.

Query multiview feature	
Command structure: `gbconfig -s multiview`	*Note:*
Response structure: `{y | n}`	
Command Example: `gbconfig -s multiview`	
Response Example: `n`	

2.3.5 Start displaying video source

Instruct the device to start displaying a video source. Please refer to **Terms & conventions section** to get the details of the parameters *VideoName,* guide, and *Src*. The optional parameter WinNo is used to specify which window of the multiview layout is used to display the video source. The specific behavior is as follows:

1. If instructed to display the guide screen, the guide screen will be displayed in full screen on all outputs.
2. If the specified video source is already displayed, the device continues to judge:

   - If a multiview layout is being used, the request message carries the `WinNo` parameter, and it is different from the child window currently occupied by the video source, the video source is moved to the specified child window for display (the video source originally displayed in the child window is dropped), and the child window originally occupied by the video source is left blank without displaying any video source.
   - In other cases, no action is performed and the response message is returned directly.

3. If only one display is currently connected, the device executes the following logic:

   - If the multiview feature is disabled, the specified video source will be displayed in full screen on the output.
   - If the multiview feature is enabled and the request message does not carry the `WinNo` parameter, the device will start displaying the video source according to the multiview switching logic. Furthermore, if the automatic switching is enabled, the device may switch layout when necessary.
   - If the multiview feature is enabled and the request message carries the `WinNo` parameter, the device will start displaying the video source in the designated child window. The video source originally displayed in the child window will be dropped.

4. If two displays are currently connected to the device, the device executes the following logic:

   - If both outputs display the guide screen, the specified video source will be displayed on both outputs.
   - If both outputs display a certain video source, the specified video source will be displayed on output 2.
   - If both outputs display two different video sources, one will be dropped and the corresponding output starts displaying the specified video source.

Start displaying video source	
Command structure 1: `gblayout --start-video {`*`VideoName`*` | guide} [`*`WinNo`*`]`	For *`VideoName`*, valid values are as follows: `usbc1, usbc2, hdmi3, hdmi4, airplay1, airplay2, airplay3, airplay4, miracast1, miracast2, miracast3, miracast4, dongle1, dongle2, dongle3, dongle4, chromecast1, chromecast2, chromecast3, chromecast4.` `guide` is a special video source which means to show guide screen. *`WinNo`* indicates an optional window number parameter. For *`Src`* valid values are `in1, in2, in3, in4, byod1, byod2, byod3` and `byod4`. For more information read **Terms & conventions** section.
Response structure 1: `(No text response message)`	
Command structure 2: `SET SHOW {`*`Src`*` | guide } start [`*`WinNo`*`]`	
Response structure 2: `SHOW {`*`Src`*` | guide} start`	
Command Example 1: `gblayout --start-video usbc1 2`	
Response Example 1: `(None)`	
Command Example 2: `SET SHOW in1 start`	
Response Example 2: `SHOW in1 start`	

2.3.6 Stop displaying video source

Instruct the device to start displaying a video source. The specific behavior is as follows:

- If the video source is not actually displayed, no action is performed.
- If only one display is currently connected and the multiview feature is disabled, the device stops displaying the video source and switches to the guide screen.
- If only one display is currently connected and the multiview feature is enabled, the device stops displaying the video source according to the multiview logic (if automatic switching is enabled, the device may switch layout when necessary). If the stopped video source is the only video source currently displayed, the device switches to the guide screen after stopping the display.
- If currently, two outputs both display the specified video source, the device stops displaying the video source and switches to the guide screen on both connected displays.
- If currently, two outputs display two different video sources, the device stops displaying the specified video source and displays the rest of the video source on both connected displays.

If the `WinNo` is used as the parameter, it means to stop displaying the video source in the specified child window. If the window number exceeds the range (of the current layout) or no video source is displayed in the child windows, no actual action is performed.

Stop displaying video source	
Command structure 1: `gblayout --stop-video {`*`VideoName`*` | `*`WinNo`*`}`	For *`VideoName`*, valid values are as follows: `usbc1, usbc2, hdmi3, hdmi4, airplay1, airplay2, airplay3, airplay4, miracast1, miracast2, miracast3, miracast4, dongle1, dongle2, dongle3, dongle4, chromecast1, chromecast2, chromecast3, chromecast4.` `WinNo` indicates an optional window number parameter. For *`Src`* valid values are `in1, in2, in3, in4, byod1, byod2, byod3` and `byod4`. For more information read **Terms & conventions** section.
Response structure 1: `(No text response message)`	
Command structure 2: `SET SHOW {`*`Src`*` | `*`WinNo`*`} stop`	
Response structure 2: `SHOW {`*`Src`*` | `*`WinNo`*`} stop`	
Command Example 1: `gblayout --start-video 2`	
Response Example 1: `(None)`	
Command Example 2: `SET SHOW in1 stop`	
Response Example 2: `SHOW in1 stop`	

2.3.7 Display video source in full screen

Instruct the device to display the video source or guide screen in full screen on all outputs.

Display video source in full screen	
Command structure 1: `gbconfig --video-source {`*`VideoName`*` | guide }`	For *`VideoName`*, valid values are as follows: `usbc1, usbc2, hdmi3, hdmi4, airplay1, airplay2, airplay3, airplay4, miracast1, miracast2, miracast3, miracast4, dongle1, dongle2, dongle3, dongle4, chromecast1, chromecast2, chromecast3, chromecast4.` `guide` is a special video source which means to show guide screen. For `Src` valid values are `in1, in2, in3, in4, byod1, byod2, byod3` and `byod4`. For more information read **Terms & conventions** section.
Response structure 1: `(No text response message)`	
Command structure 2: `SET SW {`*`Src`*` | guide } [ out1 | out2 ]`	
Response structure 2: `SHOW {`*`Src`*` | guide } [ out1 | out2 ]`	
Command Example 1: `gbconfig --video-source hdmi3 out2`	
Response Example 1: `(None)`	
Command Example 2: `SET SW guide`	
Response Example 2: `SHOW guide`	

2.3.8 Query video source displayed status

Query whether a video source is being displayed. When the request message uses `all` as the parameter, the device returns the status of all video sources row by row.

Query video source displayed status	
Command structure: `GET SHOW {`*`Src`*` | all }`	*Note:* For `Src` valid values are `in1, in2, in3, in4, byod1, byod2, byod3` and `byod4`. For more information read **Terms & conventions** section.

Response structure:
`SHOW `*`Src`*` { start | stop }`
`or`
`SHOW in1 { start | stop }`
`SHOW in2 { start | stop }`
`SHOW in3 { start | stop }`
`SHOW in4 { start | stop }`
`SHOW byod1 { start | stop }`
`SHOW byod2 { start | stop }`
`SHOW byod3 { start | stop }`
`SHOW byod4 { start | stop }`	

Command Example: `GET SHOW in2`	
Response Example: `SHOW in2 stop`	

2.3.9 Query video output status

Query which video sources are displayed on a certain output. There are several situations:

- `disconnected` means the output is not connected to a display.
- `guide` means the output does not display any video source, and the guide screen is displayed on it.
- If the multiview layout is being used, the device returns all video sources information displayed on the output in descending order of the child windows. `null` indicates the corresponding child window is idle (no video source is displayed in it).

When the request message does not carry a parameter, the device returns the status information of two outputs in two rows.

Query video output status	
Command structure: `gbconfig -s video-source [out1 | out2]`	*Note:*

Response structure:
`{ guide | disconnected | {{`*`VideoName1`*` | null}`
`{`*`VideoName2`*` | null}...}}`
`or`
`out1 { guide | disconnected | {{`*`VideoName1`*` | null}`
`{`*`VideoName2`*` | null}...}}`
`out2 { guide | disconnected | {{`*`VideoName1`*` | null}`
`{`*`VideoName2`*` | null}...}}`	

Command Example: `gbconfig -s video-source`	

Response Example:
`out1 usbc1 null hdmi3 miracast1`
`out2 disconnected`	

2.3.10 Query layout list

Query the layout list. The device returns the ID and name of all layouts.

Query layout list	
Command structure: `gblayout --list`	*Note:*

Response structure:
`Layout #    Name`
`LayoutID1   [LayoutName1]`
`LayoutID2   [LayoutName2]`
`LayoutID3   [LayoutName3]`
`...`	

Command Example: `gblayout --list`	

Response Example:
`Layout #    Name:`
`0x100       layout0`
`0x101       layout1`
`0x102       layout2`
`0x103       layout3`
`0x104       layout4`	

2.3.11 Query layout

This command has 2 uses, described as follows:

1. When the request does not carry a parameter, the device returns the ID and name (if the layout has it) of the layout being used currently.
2. When the request specifies an ID of a layout, the device returns the position and size of all child windows of the specified layout. For the position and size data, they use a virtual coordinate where the screen resolution is always 16000x9000 to make them independent of the actual screen resolution. The asterisk `*` indicates the main window.

Query layout	
Command structure: `gblayout --get [LayoutID]`	*Note:*

Response structure:
`LayoutNo [LayoutName]`
`or`
`Layout#: LayoutNo Name: LayoutName WinNum windows`
`1* Win1X Win1Y Win1W Win1H`
`2 Win2X Win2Y Win2W Win2H`
`3 Win3X Win3Y Win3W Win3H`
`...`	

Command Example 1: `gblayout --get`	
Response Example 1: `0x101 SidebySide_DualView`	
Command Example 2: `gblayout --get 0x203`	

Response Example 2:
`Layout #: 0x0202 Name: LeftRight 2 windows`
`1* 0 2250 8000 4500`
`2 12000 2250 8000 4500`	

2.3.12 Switch layout

Switch the layout (of the child windows) used by multiview mode. Described as follows:

- `LayoutID` represents the number of the layout respectively.
- This API does not work when the multiview feature is disabled.
- Multiview feature can actually work only when only one display is connected to the device, so the request message does not need to specify the output.

As the boot default, single full-screen layout is used; the corresponding `LayoutID` is `0x100`.

Switch layout	
Command structure: `gblayout --set LayoutID`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gblayout --set 0x100`	
Response Example: `(None)`	

2.4 Audio & auxiliary signal switching

2.4.1 Configure output audio source

Switch the output audio source. The specific behavior of the device depends on the specific parameters:

1. If the parameter value is `auto`, it means automatic switching, always playing the audio of the most recently connected video source. If the input is disconnected, it will fall back and play the audio of the last connected video source.
2. If the parameter designates a wired source, as long as the corresponding video source is connected, the device will play the audio of this video source always. If the video source is disconnected, the device will switch the audio source according to the `auto` logic mentioned in the previous item.
3. If the parameter designates a BYOD source, the behavior is as follows:
    - If the specified BYOD source is invalid (disconnected), nothing happens.
    - If the specified BYOD source is valid, the device will play the audio of this BYOD source, but does not modify the saved configuration.
    - When the specified BYOD source becomes invalid, the device will load the saved configuration and behave accordingly. In short, playing a BYOD source's audio is a temporary and one-time change.

As the factory default, the configuration value is set to `auto`.

Configure output audio source	
Command structure: `gbconfig --audio-source {`*`VideoName`*` | auto}`	For *`VideoName`*, valid values are as follows: `usbc1, usbc2, hdmi3, hdmi4, airplay1, airplay2, airplay3, airplay4, miracast1, miracast2, miracast3, miracast4, dongle1, dongle2, dongle3, dongle4, chromecast1, chromecast2, chromecast3, chromecast4.` For more information read **Terms & conventions** section.
Response structure: `(No text response message)`	
Command Example: `gbconfig --audio-source usbc1`	
Response Example: `(None)`	

2.4.2 Query output audio source

Query the audio source being played currently. The details are as follows:

- In the returned message, the first parameter is the saved configuration. When switched to a BYOD source, the device does not modify the saved configuration, so the value of this parameter can only be a wired source or `auto`.
- The second parameter indicates the actually played audio source. If the device is displaying the guide screen and no audio source is played, it is represented by `null`.

Query output audio source	
Command structure: `gbconfig -s audio-source`	*Note:*
Response structure: `{ `*`VideoName`*` | auto } { `*`VideoName`*` | null }`	
Command Example: `gbconfig -s audio-source`	
Response Example: `auto miracast1`	

2.4.3 Configure USB host

Configure the output interface of the USB switcher. The details are as follows:

1. The parameter `USBHost` is used to designate a certain USB host interface. The value and meaning are as follows:
   - `usbc1` - The USB channel of the USB-C IN1 port
   - `usbc2` - The USB channel of the USB-C IN2 port
   - `usbhost1` - The USB HOST1 interface
   - `usbhost2` - The USB HOST2 interface
   - `wireless` - Connect the USB switcher to the main SoC for wireless conference and other purposes.
2. The parameter value `auto` means automatic switching; the USB host interface is always connected to the most recently connected USB host interface and follows the fallback principle.

As the factory default, the configuration value is set to `auto`.

Configure USB host	
Command structure: `gbconfig --usb-host { USBHost | auto }`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --usb-host usbc1`	
Response Example: `(None)`	

2.4.4 Query USB host

Query the current USB host interface. The optional `auto` means the USB switcher is configured as automatic switching.

Query USB host	
Command structure: `gbconfig -s usb-host`	*Note:*
Response structure: `USBHost [auto]`	
Command Example: `gbconfig -s usb-host`	
Response Example: `nusbc2 auto`	

2.5 Standby related & peripheral control

2.5.1 Configure automatic standby

Configure whether the automatic standby feature is enabled. As the factory default, the automatic standby feature is enabled, and the configuration value is set to `y`.

Configure automatic standby	
Command structure: `gbconfig --auto-standby { y | n }`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --auto-standby n`	
Response Example: `(None)`	

2.5.2 Query automatic standby

Query the state of the automatic standby configuration.

Query automatic standby	
Command structure: `gbconfig -s auto-standby`	*Note:*
Response structure: `{ y | n }`	
Command Example: `gbconfig -s auto-standby`	
Response Example: `n`	

2.5.3 Configure automatic standby timeout

The parameter `TimeOut` indicates how long the device will be idle (namely, displaying the guide screen) before entering standby mode. Its unit is in seconds. The value `0` means entering standby mode immediately when the device starts being idle.

This configuration takes effect only when the automatic standby feature is enabled. The value range is from `0` to `3600`. As the factory default, the automatic standby timeout is 2 minutes; the configuration value is set to `120`.

Configure automatic standby timeout	
Command structure: `gbconfig --auto-standby-time TimeOut`	*Note:* In the example, the automatic standby timeout is set to 6 minutes.
Response structure: `(No text response message)`	
Command Example: `gbconfig --auto-standby-time 360`	
Response Example: `(None)`	

2.5.4 Query automatic standby timeout

Query the value of the automatic standby timeout.

Query automatic standby timeout	
Command structure: `gbconfig -s auto-standby-time`	*Note:*
Response structure: `TimeOut`	
Command Example: `gbconfig -s auto-standby-time`	
Response Example: `360`	

2.5.5 Configure CEC command

Configure the stored CEC command used to power on/off the display. The first parameter indicates which command will be changed. `on` and `off` correspond to power on and off, respectively.

The second parameter `CECCode` is the actual CEC message string in hexadecimal format, with no space between adjacent bytes.

As the factory default, the power on message is `4004` and the power off message is `ff36`.

Configure CEC command	
Command structure: `gbconfig --cec-cmd { on | off } CECCode`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --cec-cmd on 4004`	
Response Example: `(None)`	

2.5.6 Query CEC command

Query a CEC command configuration. If the request message has no parameter, the device returns all two CEC messages in two rows.

Query CEC command	
Command structure: `gbconfig -s cec-cmd [on | off]`	*Note:*

Response structure:
`CECCode`
`or`
`on CECCode1`
`off CECCode2`	

Command Example: `gbconfig -s cec-cmd on`	
Response Example: `4004`	

2.5.7 Configure RS232 work mode

The RS232 port supports different work modes. The value and meanings of the parameter are as follows:

- `api` - Receive the API commands sent by the external controller.
- `com` - Conventional communication, control the peripherals such as display, and participate in standby-related power on/off display operations.

As the factory default, the RS232 port is used to control the peripherals, and the configuration value is set to `com`.

Configure RS232 work mode	
Command structure: `gbconfig --rs232-role {api | com | passthrough}`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --rs232-role api`	
Response Example: `(None)`	

2.5.8 Query RS232 work mode

Query the work mode of the RS232 port.

Query RS232 work mode	
Command structure: `gbconfig -s rs232-role`	*Note:*
Response structure: `{ api | com | passthrough }`	
Command Example: `gbconfig -s rs232-role`	
Response Example: `api`	

2.5.9 Configure RS232 settings

Modify the RS232 port settings. The parameter `RS232Settings` has the format link `9600-8n1`. The settings are applicable to all scenarios using the RS232 port unless the operation specifies the RS232 settings directly.

As the factory default, the configuration is `115200-8n1`.

Configure RS232 settings	
Command structure: `gbconfig --rs232-param RS232Settings`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --rs232-param 9600-8n1`	
Response Example: `(None)`	

2.5.10 Query RS232 setting

Query the settings of the RS232 port.

Query RS232 setting	
Command structure: `gbconfig -s rs232-param`	*Note:*
Response structure: `RS232Settings`	
Command Example: `gbconfig -s rs232-param`	
Response Example: `115200-8n1`	

2.5.11 Configure RS232 baud rate

Configure the RS232 port baud rate separately. The available values are `9600`, `19200`, `38400`, `57600`, `115200`.

As the factory default, the baud rate is `115200`.

Configure RS232 baud rate	
Command structure: `gbconfig --rs232-baudrate BaudRate`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --rs232-baudrate 9600`	
Response Example: `(None)`	

2.5.12 Query RS232 baud rate

Query the baud rate of the RS232 port.

Query RS232 baud rate	
Command structure: `gbconfig -s rs232-baudrate`	*Note:*
Response structure: `BaudRate`	
Command Example: `gbconfig -s rs232-baudrate`	
Response Example: `115200`	

2.5.13 Configure RS232 command

Configure the message string sent to the peripherals through the RS232 port. The values and meanings of the parameters are as follows:

1. The first parameter indicates which command to be configured. Except for `on` and `off` (used to power on/off the display), the caller can define new commands such as `volumeup`, `volumedown`, but the command name can't contain spaces. If the specified command has not been set before, it will be added; otherwise, the existing command will be modified.
2. The second parameter indicates the format of the subsequent third parameter `CmdStr`. `hex` means hexadecimal and `str` means a printable string.
3. The third parameter is the actual command string.
    - For hexadecimal format, there is no space between the adjacent bytes. For example: `112233445566`.
    - For string format, the spaces among the words will be regarded as part of the command string. For example: `Power on`.
4. The second and third parameters are optional. If they are omitted, it means to delete a command defined already.

As the factory default, no RS232 message is defined.

Configure RS232 command	
Command structure: `gbconfig --rs232-cmd {on|off|CmdName} [{hex|str} CmdStr]`	*Note:*
Response structure: `(No text response message)`	
Command Example 1: `gbconfig --rs232-cmd on str Power on`	
Response Example 1: `(None)`	
Command Example 1: `gbconfig --rs232-cmd on`	
Response Example 1: `(None)`	

2.5.14 Query RS232 command

Query an RS232 command configuration. If the command request has no parameter, the device returns all defined RS232 messages row by row.

Query RS232 command	
Command structure: `gbconfig -s rs232-cmd [on|off|CmdName]`	*Note:*

Response structure:
`{ hex | str } CmdStr1`
`or`
`on { hex | str } CmdStr1`
`off { hex | str } CmdStr2`
`CmdName3 { hex | str } CmdStr3`
`...`	

Command Example: `gbconfig -s rs232-cmd on`	
Response Example: `str Power on`	

2.5.15 Configure power on/off display via RS232

When the device enters or exits standby mode, in addition to powering on/off the display through CEC messages, the device can also send the corresponding commands through the RS232 port. This API is used to configure whether to send RS232 commands when powering on/off the display.

Note: The actual activation of sending power on/off commands through the RS232 port requires three conditions to be met:

- The RS232 port work mode is `com`, meaning it controls peripherals.
- The RS232 `on` or `off` message is defined.
- This configuration is set to `both`.

As the factory default, this configuration is set to `cec`. When the device enters or exits standby mode, the device sends the power on/off command through CEC messages only.

Configure power on/off display via RS232	
Command structure: `gbconfig --sinkpower-mode { cec | both }`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --sinkpower-mode both`	
Response Example: `(None)`	

2.5.16 Query power on/off display via RS232

Query the configuration.

Query power on/off display via RS232	
Command structure: `gbconfig -s sinkpower-mode`	*Note:*
Response structure: `{ cec | both }`	
Command Example: `gbconfig -s sinkpower-mode`	
Response Example: `cec`	

2.5.17 Enter or exit standby mode

Manually instruct the device to enter or exit standby mode. `on` means power on (exiting standby mode) and `off` means power off (entering standby mode).

Enter or exit standby mode	

Command structure:
`gbcontrol --sinkpower { on | off }`
`gbconfig --sinkpower { on | off }`
	Note:

Response structure: `(No text response message)`	
Command Example: `gbconfig --sinkpower off`	
Response Example: `(None)`	

2.5.18 Query standby mode

Query whether the device is in standby mode.

Query standby mode	
Command structure: `gbconfig -s sinkpower`	*Note:*
Response structure: `{ on | off }`	
Command Example: `gbconfig -s sinkpower`	
Response Example: `off`	

2.5.19 Send CEC or/and RS232 command

Instruct the device to send CEC or/and RS232 command. The first parameter is the name of the sent command, and the second optional parameter indicates the sending method. If the sending method is not specified, the configuration of `gbconfig --sinkpower-mode` is used.

If the RS232 work mode is not `com` or a specified RS232 command is not defined, the device sends the CEC command only.

Send CEC or/and RS232 command	
Command structure: `gbcontrol --send-cmd { on | off | CmdName } [rs232|cec|both]`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbcontrol --send-cmd on both`	
Response Example: `(None)`	

2.5.20 Send RS232 data

Send some data through the RS232 port and receive the response data. The details of the parameters are below:

1. `-b` parameter is used to set the RS232 port settings, which contain baud rate, data bits, parity, and stop bits. By default, `9600-8n1` is used.
2. `-r { on | off }` is used to set whether to add a carriage return at the end of the data. The default is `off`.
3. `-h { on | off }` is used to indicate whether the fifth parameter `CmdStr` is in hexadecimal format. The default is `off`. So the `CmdStr` will be sent by its printable ASCII format. If the value is `on`, `CmdStr` will be interpreted as hexadecimal characters.
4. `-t timeout` is used to designate the timeout in which this command will return. When the command returns, all data received from the RS232 port will be printed as hexadecimal format. The unit is in milliseconds, and its default is `0`, meaning that the command returns immediately, and no response data will be received.
5. `CmdStr` is the data to be sent.
6. `Size` is the size of the response data received before the command returns.

Note: This is a low-level command for testing and debugging mainly. The device executes this command unconditionally, regardless of any configuration. The caller must ensure that the command is called at the appropriate time.

Send RS232 data	
Command structure: `gbcontrol --serial [-b RS232Settings] [-r{ on|off}] [-h{on|off}] [-t timeout] CmdStr`	*Note:*
Response structure: `Response Size [ xx xx xx xx… ]`	
Command Example 1: `gbcontrol --serial Hello`	
Response Example 1: `Response 0`	
Command Example 2: `gbcontrol --serial -b 115200-8n1 -h on -t 500 67 65 74 20 73 74 61 74 65`	
Response Example 2: `Response 4 67 6F 6F 64`	

2.6 Network & BYOD

2.6.1 Configure network mode

Configure whether to enable the network isolation mode.

The device has five Gigabit Ethernet interfaces: three physical RJ45 network interfaces on the rear panel, and each of the two USB-C input ports has a GbE network adapter. The device can use these Ethernet interfaces in two different modes:

- **Transparent:** All Ethernet interfaces are interconnected together, and their actual functions are also equal. To use this mode with this API, the parameter value should be n.
- **Isolated:** Only the Gigabit Ethernet interface marked **CONTROL** can be used to control the device (via Web UI or API). The rest can be used for conventional service features, such as BYOD or network accessing. To use this mode with this API, the parameter value should be y.

To get the details, please refer to the PRD or user manual.

As the factory default, the configuration value is n.

Configure network mode	
Command structure: `gbconfig --network-isolation {y | n}`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --network-isolation y`	
Response Example: `(None)`	

2.6.2 Query network mode

Query the current network mode.

Query network mode	
Command structure: `gbconfig -s network-isolation`	*Note:*
Response structure: `{ y | n }`	
Command Example: `gbconfig -s network-isolation`	
Response Example: `n`	

2.6.3 Configure IP address

Configure the IP address. Here are the details:

1. In isolated network mode, the device has two sets of IP addresses, one for control and one for conventional service. The first parameter specifies which IP address to be set.
2. The second parameter gives the IP mode. Now DHCP and static are supported.
3. For static IP address, additional parameters are used to provide IP address, subnet mask, optional gateway, and DNS server.

As the factory default, DHCP is used.

Configure IP address	
Command structure: `gbconfig --ip-address {control | service} {dhcp|static IPAddress NetMask [Gateway[DNSServer]]}`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --ip-address service static 192.168.1.88 255.255.255.0 192.168.1.1`	
Response Example: `(None)`	

2.6.4 Query IP address

Query the IP address. If the request message does not carry a parameter, the device returns the information of all two IP addresses in two rows.

Query IP address	
Command structure: `gbconfig -s ip-address [control | service]`	*Note:*

Response structure:
`{ dhcp | static} IPAddress NetMask [ Gateway [ DNSServer ] ]`
`or`
`control { dhcp | static} IPAddress NetMask [ Gateway [ DNSServer ] ]`
`service { dhcp | static} IPAddress NetMask [ Gateway [ DNSServer ] ]`	

Command Example: `gbconfig -s ip-address control`	
Response Example: `dhcp 169.254.2.155 255.255.0.0`	

2.6.5 Configure USB-C network adapter

Each of the two USB-C input ports has a GbE network adapter. These two network adapters can be disabled to improve network security. This API configures whether to enable the aforementioned network adapters.

As the factory default, USB-C network adapters are enabled, with the configuration value set to `y`.

Configure USB-C network adapter	
Command structure: `gbconfig --nic-enable {y | n}`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --nic-enable n`	
Response Example: `(None)`	

2.6.6 Query USB-C network adapter

Query the state of whether the USB-C network adapter is enabled.

Query USB-C network adapter	
Command structure: `gbconfig -s nic-enable`	*Note:*
Response structure: `{ y | n }`	
Command Example: `gbconfig -s nic-enable`	
Response Example: `n`	

2.6.7 Configure telnet over TLS

The device supports telnet and telnet over TLS. They are mutually exclusive. This command configures whether to enable telnet over TLS.

If the configuration is disabled, the telnet service runs on TCP port 23. If the configuration is enabled, the telnet over TLS service runs on TCP port 24.

As the factory default, telnet over TLS is disabled; the configuration value is n.

Configure telnet over TLS	
Command structure: `gbconfig --telnet-over-tls { y | n }`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --telnet-over-tls y`	
Response Example: `(None)`	

2.6.8 Query telnet over TLS

Query the state of whether the telnet over TLS feature is enabled.

Query telnet over TLS	
Command structure: `gbconfig -s telnet-over-tls`	*Note:*
Response structure: `{ y | n }`	
Command Example: `gbconfig -s telnet-over-tls`	
Response Example: `n`	

2.6.9 Configure HTTPS

Configure whether the HTTPS feature is enabled. When HTTPS is enabled, the connection request of the HTTP port will be redirected to the HTTPS port to continue the service.

As the factory default, HTTPS is enabled; the configuration value is y.

Configure HTTPS	
Command structure: `gbconfig --https { y | n }`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --https n`	
Response Example: `(None)`	

2.6.10 Query HTTPS

Query the state of whether the HTTPS feature is enabled.

Query HTTPS	
Command structure: `gbconfig -s https`	*Note:*
Response structure: `{ y | n }`	
Command Example: `gbconfig -s https`	
Response Example: `n`	

2.6.11 Configure Wi-Fi mode

Configure the radio band and channel used by the Wi-Fi module. The first parameter designates the band, `2` means 2.4G and `5` means 5G. The second parameter `WifiChannel` designates the channel; its value range varies depending on the band:

Band	Values Range of Channel
2.4G	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
5G	36, 40, 44, 48, 149, 153, 157, 161

The value `auto` means the device chooses the best channel automatically.

As the factory default, the device uses the 5G band and chooses a channel automatically.

Configure Wi-Fi mode	
Command structure: `gbconfig --wifi-mode { 2 | 5 } {WifiChannel | auto}`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --wifi-mode 2 1`	
Response Example: `(None)`	

2.6.12 Query Wi-Fi mode

Query the Wi-Fi mode. The optional `auto` word means the channel is chosen by the device automatically.

Query Wi-Fi mode	
Command structure: `gbconfig --wifi-mode`	*Note:* In the above example, the Wi-Fi module works on the 5 GHz band, and channel 149 is chosen automatically.
Response structure: `{ 2 | 5 } WifiChannel [ auto ]`	
Command Example: `gbconfig --wifi-mode`	
Response Example: `5 149 auto`	

2.6.13 Configure Wi-Fi hotspot

Configure whether the hotspot (soft AP) is enabled.

Note: The soft AP is designed for BYOD applications mainly. Its performance is limited. If your purpose is to access the LAN through Wi-Fi connection, we suggest deploying a standalone Wi-Fi AP to achieve a better experience.

As the factory default, the soft AP is enabled; the configuration is `y`.

Configure Wi-Fi hotspot	
Command structure: `gbconfig --softap-enable { y | n }`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --softap-enable n`	
Response Example: `(None)`	

2.6.14 Query Wi-Fi hotspot

Query whether the soft AP is enabled.

Query Wi-Fi hotspot	
Command structure: `gbconfig -s softap-enable`	*Note:*
Response structure: `{ y | n }`	
Command Example: `gbconfig -s softap-enable`	
Response Example: `y`	

2.6.15 Configure Wi-Fi password

Configures the password of the soft AP. The password must be 8-20 characters in length, and it must include only letters, numbers, and two special characters ('_' and '-'). As the factory default, the soft AP password is `12345678`.

Configure Wi-Fi password	
Command structure: `gbconfig --softap-password WifiPassword`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --softap-password 88888888`	
Response Example: `(None)`	

2.6.16 Query Wi-Fi password

Query the Wi-Fi password.

Query Wi-Fi password	
Command structure: `gbconfig -s softap-password`	*Note:*
Response structure: `WifiPassword`	
Command Example: `gbconfig -s softap-password`	
Response Example: `12345678`	

2.6.17 Configure soft router

Configure whether to enable the soft router. Basing on the soft AP, the device can launch a built-in NAT module with which a device connected to the soft AP can access the LAN/WAN through the device's wired network interface.

To make the soft router work, please make sure:

- Soft AP feature is enabled because it is the basis of the soft router.
- The DNS server is configured correctly. If static IP is used, please specify a valid DNS server.

As the factory default, the soft router is enabled, the configuration is `y`.

Configure soft router	
Command structure: `gbconfig --softap-router { y | n }`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --softap-router n`	
Response Example: `(None)`	

2.6.18 Query soft router

Query whether the soft router is enabled.

Query soft router	
Command structure: `gbconfig -s softap-router`	*Note:*
Response structure: `{ y | n }`	
Command Example: `gbconfig -s softap-router`	
Response Example: `y`	

2.6.19 Configure BYOD

Configure whether to enable the BYOD (sink). Currently, this configuration works for Airplay Mirroring, Miracast, and ChromeCast. The dongle casting is not impacted by this configuration and is always enabled. As the factory default, BYOD is enabled, and the configuration is y.

Configure BYOD	
Command structure: `gbconfig --byod-enable { y | n }`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --byod-enable n`	
Response Example: `(None)`	

2.6.20 Query BYOD

Query whether the BYOD feature is enabled.

Query BYOD	
Command structure: `gbconfig -s byod-enable`	*Note:*
Response structure: `{ y | n }`	
Command Example: `gbconfig -s byod-enable`	
Response Example: `y`	

2.6.21 Configure BYOD access code

Configure BYOD access code (PIN). The parameter `AccessCode` consists of 4 digits. `auto` means the device automatically generates the access code and changes it dynamically. none means there is no access code. Currently, the access code works for Airplay Mirroring and Miracast only. It has no use for Chromecast and Dongle.

Note: Many devices can't support the access code well. To ensure the best screen casting experience, we recommend setting an access code only when absolutely necessary. As the factory default, no access code is configured, and the configuration is none.

Configure BYOD access code	
Command structure: `gbconfig --access-code {AccessCode | auto | none}`	*Note:*
Response structure: `(No text response message)`	
Command Example: `gbconfig --access-code 1234`	
Response Example: `(None)`	

2.6.22 Query BYOD access code

Query the BYOD access code. If the word `auto` follows the `AccessCode`, it means the access code is generated by the device dynamically. `none` means no access code.

Query BYOD access code	
Command structure: `gbconfig -s access-code`	*Note:*
Response structure: `{ AccessCode [ auto ] | none }`	
Command Example: `gbconfig -s access-code`	
Response Example: `8947 auto`	