iWall X

Video Wall Controller

Central Control Commands List

V1.0

Table of contents

1. Connection

The iWall X supports both **HTTP** and **RS232** connections.

2. HTTP Connection

Port: 80

Not support **keep-alive** mode.

The URL of POST or GET is **http://device IP address /index2.html**

The POST and Response format is json.

2.1 User and Authorization

2.1.1 Login

Request type	**POST**
Content	**Json:** **{** **"Command":** "Login", **"UserName":** "UserName", **"Password":** "Password MD5 character string" **}**
Function	The user logs in successfully and obtains the Session ID and user permission. After the login is successful, for all subsequent related accesses, the session_id field must be added to the HTTP Header, and the Session ID value obtained from the login must be filled in. Otherwise, the operation will fail to

	verify the user.
Response	**Json:** **{** **"Status":** "Ok"/"Error"; **" SessionId ":** The session id of this login; -- Valid when the Status is Ok **"DeviceType":** DeviceType **"ScreenGroup":** Screen videowall group permissions, 1 means it can be operated, 0 means it cannot be operated **"WebSwitch":** 0/1 -- whether there is permission for the presets switching on web **"AndroidApp":** 0/1 -- Whether there is permission for Android App **"ModalManage":** 0/1 -- whether there is a presets management authority **"ScreenSetting":** 0/1 -- whether there is permission to modify screen parameters **"UserManage":** 0/1 -- whether there is user management authority **"DeviceStatus":** whether the device is activated -- 0- not activated 1-temporary activation, only supports this time boot operation 2-permanent activation **"GUID":** "Device ID, currently the device SN"-- DeviceStatus is not equal to 2, that is, the device is not valid until it is permanently activated **"ActiveTime":** "Activation date, the format is: year-month-day"-- DeviceStatus equal to 2 is valid, that is, the device is valid only after permanent activation **"Message":** "error message"// valid when Status is Error **}**

2.1.2 Log out

Request type	**POST**
content	**Json :** **{** **"Command":** "Logout" **" User Name ":** "Username" **}**
Function	user logout
Response	**Json :** **{** **"Status":** "Ok"/"Error"; **"Message":** "error message" // valid when Status is Error; **}**

2.1.3 Modify user password

Request type	POST
content	**Json :** **{** **"Command":** "ModifyPwd"; **" User Name ":** "Username"; **"OldPassword":** "Old Password"; **"NewPassword":** "New Password" **}**
Function	Modify user password. Both the old password and the new password are unencrypted.
Response	**Json :** **{** **"Status":**"Ok"/"Error"; **"Message":**"error message" // valid when Status is Error; **}**

2.1.4 Add user

Request type	POST
content	**Json :** **{** **"Command":**"AddUser"; **" User Name ":**"Username"; **" Password ":**"MD5 encrypted password string"; **"Role":** User role; -- 0-administrator 1-operator 2-web operator **"ScreenGroup":** [Screen videowall group permissions, 1 means it can be operated, 0 means it cannot be operated] **"WebSwitch":** 0/1 -- whether there is permission for the presets switching on web **"ModalManage":** 0/1 -- whether there is a presets management authority **"ScreenSetting":** 0/1 -- whether there is permission to modify screen parameters **"UserManage":** 0/1 -- whether there is user management authority **}**
Function	Add user
Response	**Json :** **{** **"Status":**"Ok"/"Error";

	"Message":"error message" // valid when Status is Error; **}**

2.1.5 Delete user

Request type	**POST**
content	**Json :** **{** **"Command":"**DeleteUser"; **" User Name ": _"**Username"; **}**
Function	Delete users
Response	**Json :** **{** **"Status":"**Ok"/"Error"; **"Message":"**error message" // valid when Status is Error; **}**

2.1.6 Modify user permissions

Request type	**POST**
content	**Json :** **{** **"Command":"**ModifyUserRole"; **" User Name ": _"**Username"; **"Role":** user role -- 0-administrator 1-operator 2-web operator **"ScreenGroup":** [Screen videowall group permissions, 1 means it can be operated, 0 means it cannot be operated] **"WebSwitch":** 0/1 -- whether there is permission for the presets switching on web **"ModalManage":** 0/1 -- whether there is a presets management authority **"ScreenSetting":** 0/1 -- whether there is permission to modify screen parameters **"UserManage":** 0/1 -- whether there is user management authority **}**
Function	Modify user permissions
Response	**Json :** **{** **"Status":"**Ok"/"Error"; **"Message":"**error message" // valid when Status is Error; **}**

2.1.7 Get users list

Request type	**GET**
content	Command=GetAll Users
Function	Get the user list and return user names in a list.
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **“User List ”:** -- list of usernames **[** "User1","User2", ... **]** **}**

2.1.8 Get user permissions

Request type	**GET**
content	Command=Get UserRole&UserName= Username
Function	Get user permissions
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **"Role":** user role -- 0-administrator 1-operator 2-web operator **“ScreenGroup”:** [Screen videowall group permissions, 1 means it can be operated, 0 means it cannot be operated] **"WebSwitch":** 0/1 -- whether there is permission for the presets switching on web **"ModalManage":** 0/1 -- whether there is a presets management authority **"ScreenSetting":** 0/1 -- whether there is permission to modify screen parameters **"UserManage":** 0/1 -- whether there is user management authority **}**

2.2 Screens related

2.2.1 Get Screen videowall group parameters

Request type	**GET**
content	Command= GetGroup s Settings
Function	Get Screen videowall group parameters (whether to enable videowall grouping and group name).
Response	**{** **"Status":"**Ok"/"Error"-- characters **"Message":"**error message" // valid when Status is Error **"GroupName":** ["videowall group 1 Name","videowall group 2 Name","videowall group 3 Name","videowall group 4 Name"] -- characters **"GroupEnable":** [enable flag array of Screen videowall group 1, Screen videowall group 2, Screen videowall group 3, Screen videowall group 4, value 0/1] **"DisplayScreenEnable":** [Screen videowall group 1, Screen videowall group 2, Screen videowall group 3, Screen videowall group 4 image previewing enable flag array, value 0/1] **"LeftScreenEnable":** [Screen videowall group 1, Screen videowall group 2, Screen videowall group 3, Screen videowall group 4 left independent screen enable flag array, value 0/1] **"RightScreenEnable":** [Screen videowall group 1, Screen videowall group 2, Screen videowall group 3, Screen videowall group 4 right independent screen enable flag array, value 0/1] **}**

2.2.2 Get screen setup parameters

Request type	**GET**
content	Command= GetScrGroupSettings &ScreenGroup=Screen videowall group number [1~4]
Function	Obtain the Screen videowall group parameters of the specified number of the ScreenGroup. See Appendix for the resolution definition.
Response	**{** **"Status":"**Ok"/"Error"-- characters **"Message":"**error message" // valid when Status is Error **"Name":"**Screen videowall group name"-- characters **"LogCols":** the number of columns each physical screen is divided into logical split grids **"LogRows":** the number of rows each physical screen is divided into logical split grids **"Enable":** [the main screen, the previewing screen and the enable flag array of the left & right independent screens, the value is 0/1] **"PhyRows":** [the number of physical screen rows of main screen, previewing screen and left & right independent screens] **"PhyCols ":** [Number of physical screen columns of main screen, previewing screen and left & right independent screens] **"DisplayMode":** [display mode of main screen, previewing screen and left & right independent screens] **"CustomWidths":** [column 1 width, column 2 width ...] ----- The main screen custom resolution width of each column, valid when the main screen display mode with custom resolution (42) **"CustomHeights":** [Height of line 1, height of line 2 ...] ----- The height of each row of the main screen custom resolution, valid when the display mode of the main screen with custom resolution (42)

	"PhyWidth": [the pixel width of the physical screen of the main screen, the previewing screen and the left & right independent screens] **"PhyHeight":** [the pixel height of the physical screen of the main screen, the previewing screen and the left & right independent screens] **}**

2.2.3 Get a list of output resolutions

Request type	**GET**
content	Command= GetDisplayModeList
Function	Obtain the list of output resolutions, see Appendix for resolution definitions.
Response	**{** **"Status":"**Ok"/"Error"-- characters **"Message":"**error message" // valid when Status is Error **"ModeList":** -- output resolution list **[** 1, 5, 6,... **]** **}**

2.2.4 Set screen parameters

Request type	**POST**
content	{ **"Command":"**SetScrGroupSettings" **"ScreenGroup":** Screen videowall group number, value [1~4] **"Name":** "Screen videowall group name"-- characters **"LogCols":** the number of columns each physical screen is divided into

	logical split grids **"LogRows":** the number of rows each physical screen is divided into logical split grids **"Enable":** [the main screen, the previewing screen and the enable flag array of the left & right independent screens, the value is 0/1] **"PhyRows":** [number of physical screen rows of main screen, previewing screen and left & right independent screens] **"PhyCols":** [Number of physical screen columns for main screen, previewing screen and left & right independent screens] **"DisplayMode":** [display mode of main screen, previewing screen and left & right independent screens] **"CustomWidths":** [column 1 width, column 2 width ...] ----- The main screen custom resolution width of each column, valid when the main screen display mode is custom resolution (42) "**CustomHeights":** [Height of line 1, height of line 2 ...] ----- The height of each row of the main screen custom resolution, valid when the display mode of the main screen is custom resolution (42) **"PhyWidth":** [the pixel width of the physical screen of the main screen, the previewing screen and the left & right independent screens] **"PhyHeight":** [the pixel height of the physical screen of the main screen, the previewing screen and the left & right independent screens] }
Function	On the Screen videowall group specified by ScreenGroup, set the screen parameters. See Appendix for the resolution definition.
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **}**

2.2.5 Get a list of output ports

Request type	**GET**
content	Command=GetVOList
Function	Get a list of output ports

Response	**{**
	"Status": "Ok"/"Error"-- characters
	"Message": "error message" // valid when Status is Error
	"FrameVOList": --- list of all frame output ports
	[
	{
	"FrameNO": frame number 1-4
	"MatrixMode": matrix mode list, 2 means 2 in 6 out, 5 means 5 in 3 out, 4 means 4 in 4 out, 0 means no matrix card inserted
	[4,2]
	“CardVOList”: -- single box output port list
	[
	{
	"CardNO": output card number
	“PortVOList”: -- Single card output port list
	[1,2,3,4]---port number
	}
	]
	}
	]
	}

2.2.6 Get a list of screen ports

Request type	**GET**

content	Command=GetWTYVOList&ScrGroup=Screen Number
Function	Get a list of screen ports. Screen number value range: Screen videowall groups (1~4)

Response	**{**
	"Status":"Ok"/"Error"-- characters
	"Message":"error message" // valid when Status is Error
	"VOList": --- list of all output ports
	[
	{
	"ID": Port ID,
	"Name": "port name"
	},
	...
	{
	"ID": Port ID,
	"Name": "port name"
	}
	]
	}

2.2.7 Set output port name

Request type	**POST**

content	{
	"Command": "SetVOName"
	"ID": Output port ID, obtained through the GetWTYVOList interface
	"Name": "Port Display Name"
	}

Function	Set output port name
Response	**{**

	"Status":"Ok"/"Error" **"Message":"**error message" // valid when Status is Error **}**

2.2.8 Get a list of faulty output ports

Request type	**GET**
content	Command=GetErrorPortList
Function	Get a list of failed output ports
Response	**{** **"Status":"**Ok"/"Error"-- characters **"Message":"**error message" // valid when Status is Error **"ErrorPortNo List":** -- list of the number of output ports with faults **[** 1, 2, 3,...,16 **]** **"VONameList":** --List of output port names **[** "Output 1","Output 2",...,"Output N" **]** **}**

2.2.9 Set the fault output port

Request type	**POST**
content	{ **"Command":"**SetErrorPortList" **"ErrorPortNoList":** ------ list of faulty output ports

	[1, 2, 3,...,16] }
Function	Set the fault output port
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **}**

2.2.10 Get list of matrix mode ports

Request type	**GET**
content	Command=GetMatrixPortList
Function	Get a list of matrix mode ports
Response	**{** **"Status": "**Ok"/"Error"-- characters **"Message": "**error message" // valid when Status is Error **"MatrixPortList":** -- matrix mode port list **[** **{** **"PortNo":** the number of output ports, counting from 1 **"PortOriName": "**Port Original Name" **"PortName":"**Port display name, the default is not modified to use the initialization name" **"VideoId":** signal source ID, -1 means no window is opened **"VideoName": "**Signal source name" **}** **]**

	}

2.2.11 Open video windows under matrix mode

Request type	**POST**

content	{
	"Command": "MatrixControl"
	"VideoId": the video signal ID, which is obtained from the background through the GetWtyActiveVideo interface, and -1 means to clear all windows
	"PortNo": the number of output ports, counting from 1
	}

Function	Open video windows under matrix mode
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **}**

2.2.12 Set matrix port name

Request type	**POST**

content	{
	"Command": "SetMatrixPortName"
	"PortNo": the number of output ports, counting from 1
	"PortName": "Port Display Name"
	}

Function	Set matrix port name
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error

	}

2.2.13 Get screen ON/Off setting parameters

Request type	**GET**
content	Command=GetScreenPowerSwitchInfo
Function	Get the screen ON/Off setting parameters.
Response	**{** **"Status": "**Ok"/"Error"-- characters **"Message": "**error message" // valid when Status is Error **“PowerOn”:** “Power ON all screens” **“PowerOff”:** “Power off all screens” **"ComBaudrate":** baud rate, recommend 9600 **"Com Parity":** parity, default is 2 - no parity **"ComStopBit":** stop bit, default is 1 **}**

2.2.14 Set screen switch ON/Off parameters

Request type	**POST**

content	{
	"Command":"Set ScreenPowerSwitchInfo"
	“PowerOn”: “ Power ON all screens ”
	“PowerOff”: “Power off all screens”
	"ComBaudrate": baud rate
	"ComParity": Parity
	"ComStopBit": stop bit, default is 1
	}

Function	Set the screen ON/Off parameters, the serial port command format is 0x21 0x45 0xAC 0x5D; The recommended baud rate is 9600, and the range of baud rates is 115200, 19200, 9600, 4800, 2400, 1200, 300; Parity defaults is no parity (2), and the value range is 0-even parity 1-odd parity 2-no parity 3-space; The stop bit is 1 by default, and the value range is 1 and 2.
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **}**

2.2.15 Get screen control parameters

Request type	**GET**
content	Command=GetScrControl
Function	Get screen control parameters.
Response	**{** **"Status":"**Ok"/"Error"-- characters **"Message":"**error message" // valid when Status is Error **"ShowBgPic":** Whether to display the background picture 0-hide, 1-display **}**

2.2.16 Screen control

Request type	**POST**
content	{ **"Command":** "ScreenCtrl" **"Control":** "On"/"Off"/"BgOn"/"BgOff"

	}
Function	Control operates as follows: On – powers on the screen Off – Power off the screen BgOn – show background picture BgO ff – Do not displaybackground picture
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **}**

2.2.17 Get signal source cropping parameters

Request type	**GET**
content	Command=GetClip&VideoId=Signal source ID
Function	Get the cropping parameters of the signal source.
Response	**{** **"Status":"**Ok"/"Error"-- characters **"Message":"**error message" // valid when Status is Error **"Left":** cropped left pixels **"Top":** cropped top pixels **"Right":** cropped right pixels **"Bottom":** cropped lower pixels **}**

2.2.18 Set signal source cropping

Request type	**POST**
Content	{

	"Command": "SetClip" **"VideoId":** "Signal source ID, obtained by obtaining a valid signal source interface" **"Left":** cropped left pixels **"Top":** cropped top pixels **"Right":** cropped right pixels **"Bottom":** cropped lower pixels }
Function	Set the cropping parameters of the signal source
Response	**{** **"Status":**"Ok"/"Error" **"Message":**"error message" // valid when Status is Error **}**

2.3 Video windows related

2.3.1 Open a new video signal window

Request type	**POST**
Content	{ **"Command":** "NewWindow" **"ScreenGroup":** Screen videowall group number, value [1~4] **"ScreenType":** screen type -- 0: main screen -- 1: previewing screen -- 2: left independent screen -- 3: right independent screen **"Type":** "LocalVideo" **"VideoId":** the ID of the video signal, which is obtained from the

	background through the GetWtyActiveVideo interface **"VideoName": "**Video Signal Name" **"WindowId":** window ID value, the ID value is the unique index of the window and cannot be repeated **"WindowName": "**window name"-- usually use the name of the video source **"WindowRect":** [x1, y1, x2, y2] }
Function	On screen, open a video window
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **}**

2.3.2 Open multiple video windows

Request type	**POST**
content	{ **"Command": "**NewWindowList" **"ScreenGroup":** Screen videowall group number, value [1~4] **"ScreenType":** Screen type -- 0: main screen; -- 1: previewing screen -- 2: left independent screen; -- 3: right independent screen **"WindowList":** -- list of windows [{ **"Type": "**LocalVideo" **"VideoId":** the ID value of the video signal -- the ID value is

	obtained from the background through the GetWtyActiveVideo interface **"VideoName":** "Video Signal Name" **"WindowId":** Window ID value, the ID value is the unique index of the window and cannot be repeated **"WindowName":** "window name"-- usually use the name of the video source **"WindowRect":** [x1, y1, x2, y2] }] }
Function	Open multiple video windows at a time on the specified videowall group set by ScreenGroup.
Response	**{** **"Status":**"Ok"/"Error" **"Message":**"error message" // valid when Status is Error **}**

2.3.3 Quick opening windows

Request type	**POST**
Content	{ **"Command":** "FullScreenQuickWindow" **"ScreenGroup":** Screen videowall group number, value [1~4] **"ScreenType":** screen type -- 0: main screen; 1: previewing screen -- 2: left independent screen; 3: right independent screen **"WindowCols":** -- number of window columns (1-30) **"WindowRows":** -- number of window rows (1-20) **"WindowList":** [

	{
	"Type": "LocalVideo"
	"VideoId": the ID value of the video signal -- the ID value is obtained from the background through the GetWtyActiveVideo interface
	"VideoName": "Video Signal Name"
	"WindowId": Window ID value, the ID value is the unique index of the window and cannot be repeated
	"WindowName": "window name"-- usually use the name of the video source
	}
	]
	}

Function	On the screen, quickly open video windows on the whole videowall

Response	**{**
	"Status":"Ok"/"Error"
	"Message":"error message" // valid when Status is Error
	"RectList":
	[
	{
	"WindowRect": [x1, y1, x2, y2], Status returns OK and valid
	}
	]
	}

2.3.4 Close the video windows

Request type	**POST**
content	{ **"Command": "**CloseWindow" **"ScreenGroup":** Screen videowall group number, value [1~4]

	"ScreenType": screen type -- 0: main screen; 1: previewing screen -- 2: left independent screen; 3: right independent screen **"WindowId":** Window ID value }
Function	Closes the specified window on the screen.
Response	**{** **"Status":**"Ok"/"Error" **"Message":**"error message" // valid when Status is Error **}**

2.3.5 Clear all video windows

Request type	**POST**

content	{
	"Command": "ClearScreenWindow"
	"ScreenGroup": Screen videowall group number, value [1~4]
	"ScreenType": screen type
	-- 0: main screen; 1: previewing screen
	-- 2: left independent screen; 3: right independent screen
	"ScreenCol": -- the column where the window is located (counting from 1)
	"ScreenRow": -- the row where the window is located (counting from 1)
	}

Function	Clear the specified video windows on the specific videowall group specified by ScreenGroup.
Response	**{** **"Status":**"Ok"/"Error" **"Message":**"error message" // valid when Status is Error **}**

2.3.6 Moving window

Request type	**POST**

content	{
	"Command": "MoveWindow"
	"ScreenGroup": Screen videowall group number, value [1~4]
	"ScreenType": screen type
	-- 0: main screen; 1: previewing screen
	-- 2: left independent screen; 3: right independent screen
	"WindowId": Window ID value, the ID value is the unique index of the window and cannot be repeated
	"WindowRect": [x1, y1, x2, y2]
	"SetTopMost": 0/1 – - 1 means to set the current window to the top
	"IsMoving": 0/1 – - 1 move window, 0 modify window size
	}

Function	Move the position of the specified window, and you can set whether to set the window to the top.
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **"WindowRect":** [x1, y1, x2, y2] Window coordinates, Status returns OK and valid **}**

2.3.7 Switch video source

Request type	**POST**
content	{ **"Command": "**ChangeVideo" **"ScreenGroup":** Screen videowall group number, value [1~4]

	"ScreenType": screen type -- 0: main screen; 1: previewing screen -- 2: left independent screen; 3: right independent screen **"Type":** "LocalWindow" **"VideoId":** the ID value of the video signal, which is obtained from the background through the GetWtyActiveVideo interface **"VideoName":** "Signal source name" **"WindowId":** Window ID value }
Function	Switch the video source of the specified window.
Response	**{** **"Status":**"Ok"/"Error" **"Message":**"error message" // valid when Status is Error **}**

2.3.8 Modify the window layers

Request type	**POST**

content	{
	"Command": "SetWindowPos"
	"ScreenGroup": Screen videowall group number, value [1~4]
	"ScreenType": screen type
	-- 0: main screen; 1: previewing screen
	-- 2: left independent screen; 3: right independent screen
	"Type": "MoveDown"/"MoveUp"/"MoveTop"/"MoveBottom"
	"WindowId": Window ID value
	}

Function	Modify the window layers. MoveDown: Move the window down one level MoveUp: Move the window up one level

	MoveTop: The window is moved to the top layer MoveBottom: The window is moved to the bottom layers
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **}**

2.3.9 Close all windows

Request type	**POST**

content	{
	"Command": "CloseWindow"
	"ScreenGroup": Screen videowall group number, value [1~4]
	"WindowId": -1
	}

Function	Close all windows on the screen videowall group
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **}**

2.3.10 Get the list of current windows

Request type	**GET**
Content	Command=GetCurrentModal&ScreenGroup=Screen videowall group number, value [1~4]
Function	Get a list of all windows on the Screen videowall group
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error

	"MainScrList": -- list of main screen windows, arranged from bottom to top
	[
	{
	"WindowName": "window name"
	"WindowId": window ID
	"Type": "LocalWindow"
	"VideoId": video source ID
	"WindowRect": [x1, y1, x2, y2]
	}
	]
	"LeftScrList": -- the list of independent screen windows on the left, the content is the same as above
	[
	...
	]
	"RightScrList": -- the list of independent screen windows on the right, the content is the same as above
	[
	...
	]
	}

2.3.11 Get the list of presets windows

Request type	**GET**
content	Command= GetOneModal &ScreenGroup=Screen videowall group number, value [1~4] & ModalName= preset name
Function	Get a list of all windows of the preset
Response	**{**

	"Status": "Ok"/"Error"
	"Message": "error message" // valid when Status is Error
	"MainScrList": -- list of main screen windows, arranged from bottom to top
	[
	{
	"WindowName": "window name"
	"**WindowId":** window ID
	"Type": "LocalWindow"
	"**VideoId":** video source ID
	"**WindowRect":** [x1, y1, x2, y2]
	}
	]
	"LeftScrList": -- the list of independent screen windows on the left, the content is the same as above
	[
	...
	]
	"RightScrList": -- the list of independent screen windows on the right, the content is the same as above
	[
	...
	]
	}

2.4 Presets management

2.4.1 Save current preset

Request type	**POST**

content	**{** **"Command":** "SaveModal" **"ScreenGroup":** Screen videowall group number, value [1~4] **"ModalName":**"Preset Name" **"GroupName":**"Preset group name" **}**
Function	Save all current windows to the specified preset.
Response	**{** **"Status":**"Ok"/"Error" **"Message":**"error message" // valid when Status is Error **}**

2.4.2 Call the existing preset

Request type	**POST**
content	**{** **"Command":** "LoadModal" **"ScreenGroup":** Screen videowall group number, value [1~4] **"ModalName":**"preset Name" **}**
Function	Call an existing preset
Response	**{** **"Status":** "Ok"/"Error" **"Message":** "error message" // valid when Status is Error **"MainScrList":** -- list of main screen windows, arranged from bottom to top [{ "**WindowName":** "window name" "**WindowId":** window ID

	"Type": "LocalWindow" "**VideoId":** video source ID "**WindowRect":** [x1, y1, x2, y2] }] **"LeftScrList":** -- the list of independent screen windows on the left, the content is the same as above [...] **"RightScrList":** -- the list of independent screen windows on the right, the content is the same as above [...] **}**

2.4.3 Delete the existing preset

Request type	**POST**
content	**{** **"Command": "**RemoveModal" **"ScreenGroup":** Screen videowall group number, value [1~4] **"ModalName":** "preset Name" **}**
Function	Delete all current presets.
Response	**{** **"Status": "**Ok"/"Error" **"Message": "**error message" // valid when Status is Error **}**

2.4.4 Get all the presets of the current screen

Request type	**GET**
content	Command=GetAllModal&ScreenGroup=Screen videowall group number, value [1~4]
Function	Get all the presets on the current local screen, and return the names in a list.
Response	**{** **"Status":** "Ok"/"Error" **"Message":** "error message" // valid when Status is Error **"ModalList":** -- List of scenario names **[** "preset Name 1","preset Name 2", … **]** **}**

2.4.5 Get the currently loaded preset

Request type	**GET**
content	Command=GetCurrModal&ScreenGroup=Screen videowall group number, value [1~4]
Function	Get the currently loaded presets of the local Screen videowall group.
Response	**{** **"Status":** "Ok"/"Error" **"Message":** "error message" // valid when Status is Error **"Modal":** "Preset Name" -- Scenario name **}**

2.5 Video signal related

2.5.1 Get a list of valid video signals

Request type	**GET**
content	Command=GetWtyActiveVideo
Function	Get a list of available video sources for the host. If the commands executes successfully, it will return a signal list in the form of an array.
Response	**{** **"Status":** "Ok"/"Error" **"Message":** "error message" // valid when Status is Error **"VideoList":** -- signal list **[** **{** **"Type":** "HDMI"/"UHD" **"Name":** "signal name" **"Id":** signal ID, -- use this ID value when opening a window with a signal **"Mode":** signal source format, -- that is, signal source resolution **}** **]** **}**

2.5.2 Set video signal name

Request type	**POST**
content	**{** **"Command": "SetViName"** **"VideoId":** Source ID **"VideoName":** "Signal name" **}**

Function	Set the video signal name.
Response	**{** **"Status":** "Ok"/"Error" **"Message":** "error message" // valid when Status is Error **}**

2.5.3 Get all signal list function interface

Request type	**GET**
content	Command=GetVideoState
Function	Get all signal list function interface. If the commands executes successfully, it will return a signal list in the form of an array.
Response	**{** **"Status":** "Ok"/"Error" **"Message":** "error message" // valid when Status is Error **"CardState":** -- signal list **[** **{** "CardId": card slot number 1-16, "CardName": "Card Slot Name", "SubList": -- subcard list **[** **{** "Sub": sub-card number 1-2, "SubType": board type, see appendix, "SubName": "Subcard Name", "PortList": ---Port list [{ "Port": port number 1-4,

	"Id": port ID, which is the signal source ID, "PortName": "Port Name", "VideoMode": input resolution, see Appendix for detailed resolution }, ...] **},** ... **]** **},** **...** **]** **}**

2.6 Audio signal related

2.6.1 Get the current audio channels

Request type	**GET**
content	Command= GetAudioChn&Frame=Frame number [1-8]
Function	Get the current audio channel.
Response	**{** **"Status":** "Ok"/"Error" **"Message":** "error message" // valid when Status is Error **"AudioChn":** The current audio channel, that is, the video signal ID, -1 means no channel is selected **}**

2.6.2 Set audio channel

Request type	POST

content	{
	"Command": "Set AudioChn"
	"VideoId": The ID is the video signal ID; an ID of -1 means that the controller does not have any output audio
	"EnableAudio": Enable or disable the audio of this channel, valid when the ID is not -1
	}

Function	Set the audio channel.

Response	{
	"Status": "Ok"/"Error"
	"Message": "error message" // valid when Status is Error
	}

2.7 Scrolling text and background picture related

2.7.1 Get scrolling text information

Request type	GET
content	Command=GetWTYOSD&ScreenGroup=Group[1-4]
Function	Get scrolling text information. x1 is the horizontal coordinate of the upper left corner of the scrolling text window, y1 is the vertical pixel coordinate of the upper left corner of the scrolling text window, x2 is the horizontal pixel coordinate of the lower right corner of the scrolling text window, and y2 is the vertical pixel coordinate of the lower right corner of the scrolling text window.
Response	{ "Status": "Ok"/"Error" "Message": "error message" // valid when Status is Error

	"ShowOsd": Whether to display scrolling texts, 0-not display, 1-display **"FontName": "**font name" **"FontSize":** font size **"Text":** "scrolling text Content" **"TextColor": "**Text color in format RRGGBB" **"BgColor": "**Background color, the format is RRGGBB" **"ScrollSpeed":** scrolling speed, 0-still, 1-very slow, 2-slow, 3-quick, 4-very fast **"Alpha":** Transparency 0-255 **"ScrRange":** [x1, y1, x2, y2] **}**

2.7.2 Update scrolling text bitmap file

Request type	**POST**
content	**{** **"Command": "**UpdateWTYOsdData" **"ScreenGroup":** Screen videowall grouping [1-4] **"FileName":** "scrolling text bitmap file, the suffix is .dat, including the relative path of the file /Caption/" **"FileLen":** file size **"MD5":** "MD5 check code of file" **"TextLine":** the number of text lines **"RowHeight":** the height of each row **}**
Function	Update the scrolling text bitmap file. TextLine and RowHeight are obtained from the dot matrix interface generated by the dynamic library.
Response	**{** **"Status":** "Ok"/"Error" **"Message":** "error message" // valid when Status is Error **}**

2.7.3 Set scrolling text information

Request type	**POST**
content	**{** **"Command":** "SetWTYOsdText" **“ScrGroup”:** Screen videowall group number, value [1~4] **"FontName":** "font name" **“FontSize”:** font size **"Text":** "content" **“TextColor”:** "text color", such as"RRGGBB" **“B g Color”:** "background-color", such as"RRGGBB" **“ScrollSpeed”:** Scroll speed, 1-stationary, 2-very slow, 3-slow, 4-quick, 5-very fast **"Alpha":** Transparency 0-255 **“ScrRange”:** [x1, y1, x2, y2] **}**
Function	Set scrolling text information. x1 is the horizontal coordinate of the upper left corner of the scrolling text window, y1 is the vertical pixel coordinate of the upper left corner of the scrolling text window, x2 is the horizontal pixel coordinate of the lower right corner of the scrolling text window, and y2 is the vertical pixel coordinate of the lower right corner of the scrolling text window. To modify the font name, font size, and content, it is necessary to generate a dot matrix first, and use UpdateWTYOsdData to send the dot matrix to the server, and then send the command to set scrolling text information.
Response	**{** **"Status":** "Ok"/"Error" **"Message":** "error message" // valid when Status is Error **}**

2.7.4 Clear scrolling texts

Request type	**POST**

content	**{** **"Command":** "ClearOsdText" **"ScrGroup":** Screen videowall group number, value [1~4] **}**
Function	Clear scrolling texts.
Response	**{** **"Status":** "Ok"/"Error" **"Message":** "error message" // valid when Status is Error **}**

2.8 System related

2.8.1 Find device

- Obtained by UDP multicast, port 5000, multicast address 239.255.255.250, protocol content in JSON format.

  Client sends content:

```
{
"Command":"FindViewTop",
"IpAddr":"Client IP"
}
```

  Server response content:

```
{
"Command":"ResponseToFind",
"HostIp":"Server IP",
"HostName":"Server Name",
"HostType": device type,
"DeviceSN":"Device SN serial number",
"TargetIp":"Client IP"
}
```

2.8.2 Get IP address

Request type	**GET**
content	Command= GetIpAddr
Function	Get IP address

Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **"IP":"**IP address" **“Mask”:"**device mask" **“Gateway”:"**Gateway" **}**

2.8.3 Set IP address

Request type	**POST**
content	**{** **"Command":"**SetIpAddr" **"IP":"**IP address" **“Mask”:"**device mask" **“Gateway”:"**Gateway" **"DNS1":"**Primary DNS" **"DNS2":"**Alternate DNS" **}**
Function	Set the IP address.
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **}**

2.8.4 Get server language

Request type	**GET**
content	Command=GetLanguage
Function	Get the language of the server.
Response	**{**

	"Status":"Ok"/"Error" **"Message":"**error message" // valid when Status is Error **"Language":** Background language 0-English 1-Simplified Chinese 2-Traditional Chinese **}**

2.8.5 Set server language

Request type	**POST**
content	**{** **"Command":"**SetLanguage" **"Language":** Background language 0-English 1-Simplified Chinese 2-Traditional Chinese **}**
Function	Set the server language.
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **}**

2.8.6 Activate device

Request type	**POST**
content	**{** **"Command":"**ImportKey" **"LicenseKey":"**Activation code"--Temporary activation code: 888888 **}**
Function	Activate the device.
Response	**{** **"Status":"**Ok"/"Error" **"ActiveTime":"**Activation date, the format is: year-month-day"-- the device is valid only if the permanent activation is successful **"Message":"**error message" // valid when Status is Error

	}

2.8.7 Get the version

Request type	**GET**
content	Command= GetVersion
Function	Get the version number of the embedded server.
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **"Version":** --server version number (AA .BB.CCC.DD) **[** AA, BB, CCC, DD – shaping **]** **}**

2.8.8 Get MAC address

Request type	**GET**
content	Command= GetMacAddress
Function	Obtain MAC address information.
Response	**{** **"Status":"**Ok"/"Error" **"Message":"**error message" // valid when Status is Error **"Addr":** -- MAC address information **[** XX,XX,XXX,XX,XXX,XX – shaping **]** **}**

2.8.9 Obtain authorization information

Request type	**GET**
content	Command=GetLicInfo
Function	Obtain authorization information.
Response	**{** **"Status":"**Ok"/"Error"//Reply Error when the authorization expires **"Message":"**error message" // valid when Status is Error **"DeviceType":** device type -- see appendix **“DeviceSN”: “** Device Serial Number ” **"LicenseType":** authorization type 0- unlimited, 1- limited-time version **"ExpYear":** authorization expiration time: year-2000 Valid when LicenseType=1 **“ExpMonth”:** Authorization Expiry Date: Month Valid when LicenseType=1 **“Exp Day”:** Authorization expiration time: day Valid when LicenseType=1 **“ExpHour”:** Authorization expiration time: hours Valid when LicenseType=1 **“ExpMinute”:** Authorization expiration time: minutes Valid when LicenseType=1 **"HDInputCnt":** the number of high-definition video signals (HDMI, DVI, SDI, VGA) **“_4KInputCnt”:** Number of 4K video signals **"OutputCnt":** Display the number of ports (the sum of the number of HD and 4K outputs) **"_4KOutputCnt":** 4K output quantity **“bWebControl”:** 0/1 -- whether there is Android APP permission **}**

2.8.10 Reset

Request type	**POST**
content	**{** **"Command":** "Remove All Settings" **}**
Function	Restore to factory settings. All settings are removed.
Response	**{** **"Status":** "Ok"/"Error" **"Message":** "error message" // valid when Status is Error **}**

3. RS232 Connection

3.1 Config

Baud Rate: 9600

Parity: None

Trafic control: None

Bit: 8

Stop: 1

3.2 Packet format

The commands sent by the central control to the device and the responses sent by the device to the central control are all encapsulated into data packets. Each data packet starts with the $ character and ends

with the ^, as shown in the following figure:

Op represents an operation command, one character.

Scr represents the screen group number to be operated, one character. For iWall X, it is 1 (the iWall X 1U doesn't support multiple video wall groups)

$	Op	Scr	DATA	^

3.3 Serial Command

3.3.1 Get the Presets list

Op	1
Scr	One byte, values 1 to 4, specifying the video wall group number
DATA	No
Application sample	Get videowall group One Preset: $11^ Get videowall group Two Preset: $12^ Get videowall group Three Preset: $13^ Get videowall group Four Preset: $14^

Equipment response：

Op	1
DATA	A number of preset names separated by spaces (UTF8 format). For example:

	preset one preset two

3.3.2 Switching presets

Op	3
Scr	One byte, values 1 to 4, specifying the video wall group number
DATA	Name of preset (UTF8 format)
Application sample	Switch screen group one "preset five": $31 preset five^ Switch screen group two "preset ab": $32 ab^ Switch screen group three "preset 123ab": $33123 ab ^ Switch screen group four "preset 6a": $34 preset 6a ^ Preset Name "1" Code：24 33 31 31 5E Preset Name "2" Code：24 33 31 32 5E Preset Name "3" Code：24 33 31 33 5E

Equipment response：

Op	3
DATA	Ok or Error

3.3.3 Video switching under matrix mode

Op	4

Scr	No
DATA	[Matrix Mode Command]
Application sample	Commands: Input 1 to Output 1: $4[1X1]^ Input 2 to Outputs 1, 3 and 4: $4[2X1,3,4]^ Input 1 to all Outputs: $4[1ALL]^ Inputs and Outputs one-by-one: $4[ALL1]^ Close Output 3: $4[0X3]^ Close Outputs 1, 2, 4, and 6: $4[0X1,2,4,6]^ Close all Outputs: $4[0ALL]^ Hexadecimal as below: Input 1 to Output 1: 24 34 5B 31 58 31 5D 5E Input 2 to Outputs 1, 3 and 4: 24 34 5B 32 58 31 2C 33 2C 34 5D 5E Input 1 to all Outputs: 24 34 5B 31 41 4C 4C 5D 5E Inputs and Outputs one-by-one: 24 34 5B 41 4C 4C 31 5D 5E Close Output 3: 24 34 5B 30 58 33 5D 5E Close Outputs 1, 2, 4, and 6: 24 34 5B 30 58 31 2C 32 2C 34 2C 36 5D 5E Close all Outputs: 24 34 5B 30 41 4C 4C 5D 5E

Equipment response:

Op	4
DATA	Ok or Error

3.3.4 Get the controller IP

Op	6
Scr	No
DATA	No
Application sample	$6^

Equipment response:

Op	6
DATA	Controller IP. Ror instance: IP:[192.168.1.250]

3.3.5 Set the controller IP address

Op	7
Scr	No
DATA	IP address
Application	Set the Address IP 192.168.1.200: $7192.168.1.200^

sample	

Equipment response:

Op	7
DATA	Ok or Error

Appendix 1: Resolution Definition

Value	Resolution
0	800*600_56
1	800*600_60
2	800*600_72
3	800*600_75
4	800*600_85
5	1280*1024_60
6	1280*1024_75
7	PAL
8	640*480_60
9	640*480_72
10	640*480_75
11	640*480_85
12	1024*768_60
13	1024*768_70
14	1024*768_75
15	1024*768_85
16	1280*720_25
17	1280*720_30
18	1280*720_50
19	1280*720_60
20	1920*1080_25
21	1920*1080_30
22	1920*1080_50
23	1920*1080_60
24	1920*1080_50I
25	1920*1080_60I
26	NTSC
27	1440*900_60

28	1440*900_75
29	1440*900_85
30	1360*768_60
31	1360*768_75
3 2	1400*1050_60
3 3	1400*1050_75
3 4	1280*768_60
3 5	1280*768_75
3 6	1280*768_85
3 7	1280*1024_85
3 8	1600*1200_60
3 9	1680*1050_60
4 0	1680*1050_50
4 1	3840*2160
4 2	custom resolution
4 3	Failed to detect
4 4	no signal

Appendix 2: Device Types

Value	Resolution
0	2004
1	2008
2	2012B
3	2016
4	2016B
5	2024
6	3004
7	3008
8	3012B
9	3016
10	3016B
11	3024

Appendix 3: Port ID and Position Conversion Formula

Port ID, window number, card slot number, subcard number, and port number conversion formula:
Port ID =(((window number 0-7)<<10)|((card slot number 0-15)<<6)|((subcard number 0-1)<<4)|(port number 0-7))
Port number (0-7) = ((Port ID)&0x3F) // bit 0~ 3
Subcard number (0-1) =(((Port ID)>> 4)&0x1F) // bit 4~5
Card slot number (0-15) = (((Port ID)>>6)&0x3F) // bit 6 ~ 9
Frame number (0-7) = (((Port ID)>>10)&0x7) // bit 10 ~1 2
The window number is 0 when there is no multi-windows.

Appendix 4: Card types

The card types are as follows:

Value	Card type
0	4K30 vertical input sub card
1	2K horizontal input sub card
2	2K horizontal plug-in output sub card
3	2K vertical input sub card
4	2K vertical plug output sub card
5	4K60 input sub card
6	4K60 output sub card
16	No card