Appmainprog Communication Protocol

Version	Revision History	Editor	Time
V1.1	initial version	Pingan.Liu	2017-8-30
V1.2	Add Get AP list item	jin.wang willem.zhang	2017-9-27 2017-11-21
V1.3	Add 'bssid' that is one of 'network' attributes	Jin.wang Willem zhang	2017-10-11 2017-11-21
V1.4	Modify 'process' precision from m to ms	Jin.wang Willem.zhang	2017-10-18 2017-11-21
V1.5	Add BT power on/off status and cmd	Ruitian hou Willem zhang	2017-11-03 2018-01-14
V1.6	Add BT mode cmd Bluetooth Mode_Switch Bluetooth Stop Inquiry Bluetooth Unpair Get BT Inquiry list Bluetooth Connect Add some status cmd Bluetooth status change	Ruitian hou Willem zhang	2018-01-14
V1.7	Add Get BT paired list Bluetooth Unpair add mac as param	Ruitian hou	2018-01-29

Table of Contents

IPC introduction

The Third party programs communicate with appmainprog by unix domain socket, socket type is UDP. On the socket, encapsulated data by cJSON.

1 Process Communication Protocol Command

The Third party programs send message to appmainprog，it is no need response to third programs，and type is notification.

1.1 Play the specified content

Type	notification

Example	{
	"command": "/playback/play",
	"params": {
	"uri": <string>,
	“audioId”:<string>,
	“audioUid”:<string>,
	“audioSource”:<string>
	“audioName”:<string>
	“audioAnchor”:<string>
	“audioAlbum”:<string>
	“progress”:<int>,
	“audioExt”:<string>,
	}
	}

parameter description	uri: streaming media address, need to support： 1. HTTP address; 2. local file protocol address, like file:// ; 3. local file:begin with a slash ; audio_name:music title audio_anchor: Artist name audio_album: source progress:playback progress audioUid: equipment side send，need return status , BT playback set to "" audioExt：equipment side send，need return status , BT playback set to ""
Behavior description	After receiving the command, player automatic loading URI content and playback, update the player status; Information display to the screen, display style assign by the PRD.

1.2 Play voice prompt

Type	notification

Example	{
	"command": " /playback/play_voice_prompt ",
	"params": {
	"uri": <string>,
	“volume”:<int>,
	“type”:<string>,
	“feedback”:<boolean>
	}
	}

parameter description	uri: streaming media address, need to support： 1. HTTP address; 2. local file protocol address, like file:// ; 3. local file:begin with a slash ; feedback: play done whether need return play_done to third program； volume: playback prompt volume type: normal/alarm/reset
Behavior description	play voice prompt

1.3 Play TTS

Type	notification

Example	{
	"command": " /playback/play_tts ",
	"params": {
	“tts_id”:<int>,
	“speech_enable”:<boolean>,
	“need_mix”:<boolean>
	}
	}

parameter description	tts_id:tts streaming ID speech_enable: Whether or not support continuously receive audio need_mix：Whether or not support mix audio
Behavior description	Playback TTS

1.4 Playback pause

Type	notification

Example	{
	"command": "/playback/pause"
	}

parameter description	NULL
Behavior description	Pause playback, if current status is pause then ignore it; update player status information

1.5 Playback stop

Type	notification

Example	{
	"command": "/playback/stop"
	}

parameter description	NULL
Behavior description	Pause playback, can't resume

1.6 Playback resume

Type	notification

Example	{
	"command": "/playback/resume"
	}

parameter description	NULL
Behavior description	Recovery playback, if in a state of play, ignore it ; Update player status information

1.7 Set volume

Type	notification

Example	{
	"command": "/system/set_volume",
	"params": {
	"volume": <int>

	} }
parameter description	volume: integer, range [0, 100], 0 mute, 100 largest volume
Behavior description	according to the percentage update volume

1.8 Adjust progress

Type	notification

Example	{
	"command": "/playback/adjust_progress",
	"params": {
	"progress": <int>
	}
	}

parameter description	progress: adjust audio playback progress, accuracy: seconds
Behavior description	Specifies the progress of audio playback

1.9 Set system status

Type	notification

Example	{
	"command": "/system/set_system_status",
	"params": {
	"status": <string>
	}
	}

parameter description	Set status of the speaker: normal/standby
Behavior description	Set normal/standby

1.10 Play previous audio

Type	Notification

Example	{
	"command": "/bluetooth/play_prev_audio"
	}

parameter description	NULL
Behavior description	Play the previous audio（use for Bluetooth）

1.11 Play next audio

Type	Notification

Example	{
	"command": "/bluetooth/play_next_audio"
	}

parameter description	NULL
Behavior description	Play the next audio（use for Bluetooth）

1.12 Set Bluetooth name

Type	Notification

Example	{
	"command": "/bluetooth/set_bt_name",
	"params":{
	"name": <string>
	}
	}

parameter description	name: Bluetooth SSID name
Behavior description	Set Bluetooth name

1.13 Start Bluetooth pair

Type	Notification

Example	{
	"command": "/bluetooth/start_bt_pair"
	}

parameter description	NULL
Behavior	Turn on Bluetooth pairing mode

description	

1.14 PowerOn Bluetooth

Type	Notification

Example	{
	"command": "/bluetooth/power_on"
	}

parameter description	NULL
Behavior description	Open Bluetooth

1.15 PowerOff Bluetooth

Type	Notification

Example	{
	"command": "/bluetooth/power_off"
	}

parameter description	NULL
Behavior description	Close Bluetooth

1.16 Delete Bluetooth pair

Type	Notification

Example	{
	"command": "/bluetooth/del_bt_paired"
	
	}

parameter description	NULL
Behavior description	Delete Bluetooth pair info

1.17 Play Bluetooth music

Type	Notification

Example	{
	"command": "/bluetooth/play_bt_music"
	}

parameter description	NULL
Behavior description	Play Bluetooth music

1.18 Disconnected Bluetooth

Type	Notification

Example	{
	"command": "/bluetooth/bt_disconnect"
	}

parameter description	NULL
Behavior description	Disconnect the Bluetooth connection

1.19 Bluetooth Mode_Switch

Type	Notification

Example	{
	"command": "/bluetooth/mode_switch"
	}

parameter description	NULL
Behavior description	bluetooth/mode_switch

1.20 Bluetooth Stop Inquiry

Type	Notification

Example	{
	"command": "/bluetooth/stop_inquiry"
	}

parameter description	NULL
Behavior description	Bluetooth stop_inquiry

1.21 Bluetooth Unpair

Type	Notification

Example	{
	"command": "/bluetooth/unpair"
	"params": {
	" mac ": <string>
	}

parameter description	NULL
Behavior description	Bluetooth unpair

1.22 Bluetooth Connect

Type	Notification

Example	{
	"command": "/ bluetooth / connect ",
	"params": {
	" mac ": <string>
	},
	"id": <int>
	}

parameter description	NULL
Behavior description	Bluetooth connect

1.23 Speech start

Type	Notification

Request	{
	"command": "/speech/start"
	}

parameter description	NULL
Behavior description	Voice wake up, start speech recognition

1.24 Speech process

Type	Notification

Request	{
	"command": "/speech/process"
	}

parameter description	
Behavior description	Voice wake up, start speech process

1.25 Speech feedback

Type	Notification

Request	{
	"command": "/speech/feedback"
	}

parameter description	
Behavior description	Voice wake up, start voice feedback

1.26 Speech finish

Type	Notification

Request	{
	"command": "/speech/finish"
	}

parameter description	
Behavior description	Voice wake up, end speech recognition

1.27 Wifi setup result

Type	Notification

Request	{
	"command": "/wifi_setting/wifi_setup_result",
	“params”: {
	“result”:<int>

	} }
parameter description	NULL
Behavior description	Notify the results of the system network config 0：success 1：fail

1.28 OTA upgrade

Type	Notification

Request	{
	"command": "/system/ota_upgrade"
	“params”: {
	“ota_url”:<string>
	}
	}

parameter description	NULL
Behavior description	OTA upgrade, according to ota_url download image to upgrade

1.29 Factory reset

Type	Notification

Request	{
	"command": "/system/factory_reset_result",
	“params”: {
	“result”:<int>
	}
	}

parameter description	NULL
Behavior description	Notify the results of the system factory reset 0：success 1：fail

2 The Third Program Interaction With Appmainprog

The third-party program gets the appmainprog state by RPC which is called "request -response".The third party program sends a request JSON to appmainprog firstly, According to the command field, Appmainprog recieve the request to return the content, and then organize a response JSON string which will be returned to the third party program. Note that the id field in the response needs to be same as in the request and is used to indicate the correspondence between the response and the request.

2.1 Get speaker status

Type	rpc

Request	{
	"command": "/system/get_speaker_status",
	"id": <int>
	}

Response	{
	
	"result": {
	"command": "/system/get_speaker_status",
	"system": {
	"status": <string>,
	"wifi_mac ": <string>,
	“bt_mac”:<string>,
	“device_sn”:<string>,
	},
	"player": {
	“volume”:<int>,
	"status": <string>,
	“source”:<string>,
	“audioId”:<string>
	“audioSource”:<string>
	“audioName”:<string>
	“audioAnchor”:<string>
	“audioAlbum”:<string>
	“progress”:<int>
	},
	"network": {

	"quantity": <int>,
	"status": <string>,
	“ssid”:<string>
	"bssid": <string>
	},
	"bluetooth": {
	"status": <string>,
	"name": <string>,
	“bt_paired_name”:<string>
	},
	…….
	]
	},
	"id": <int>
	}

parameter description	mac_address: The mac address used in the current device, formatted like AA: CC: CC: CA: CA: CA, case insensitive
Behavior description	Get the current system status

2.2 Get AP list

Type	rpc

Request	{
	"command": "/wifi /get_ap_list",
	"id": <int>
	}

Response	{ "command": "/wifi /get_ap_list", "id": <int> "finish": <int> "list": [{ "ssid": <string>, "bssid": <string>, "auth_mode: <int>, "level": <int>, "frequency": <int>, }, ...

	] }
parameter description	id： id of this message finish： upload scan result is finished auth_mode: kind of authorize modes -1： unknow, need to verify by scan 0： no encryption 1： wep 2： wpa/wpa2
Behavior description	Third party program get SSID and password, notify appmainprog to connect to the network, and appmainprog response network status to the third party program.

2.3 Get BT Inquiry list

Type	rpc

Request	{
	"command": "/ bluetooth /start_inquiry",
	"id": <int>
	}

Response	{
	"command": "/ bluetooth /start_inquiry",
	"id": <int>
	"finish": <int>
	"list": [
	{
	" mac": <string>,
	" name": <string>,
	" type": <int>,
	},
	...
	]
	}

parameter description	id： id of this message finish： upload inquiry result is finished
Behavior description	Get BT inquiry result

2.4 Get BT paired list

Type	rpc

Request	{
	"command": "/ bluetooth /paired_list_update",
	}

Response	{
	"command": "/ bluetooth / paired_list_update ",
	"list": [
	{
	" mac": <string>,
	" name": <string>,
	" role": <int>,
	},
	}

parameter description	role: (paired devices) 0: BT_A2DP_ROLE_SRC, 1: BT_A2DP_ROLE_SINK, 2: BT_A2DP_ROLE_MAX
Behavior description	Get BT paired list result

2.5 Wifi connect

Type	rpc

Request	{
	"command": "/wifi_setting/wifi_connect",
	"params": {
	"ssid": <string>,
	"bssid": <string>,
	“password”:<string>,
	“bssid”:<string>
	},
	"id": <int>
	}

Response	{ "result": {

	"wifi_status": <int> }, "id": <int> }
parameter description	auth_mode: kind of authorize modes -1： unknow, need to verify by scan 0： no encryption 1： wep 2： wpa/wpa2 wifi_status： 0： connection success 1： error password 2： ap is not exist 3： connection timed out 4： Failed to obtain IP 10： unknow error
Behavior description	Third party program get SSID and password, notify appmainprog to connect to the network, and appmainprog response network status to the third party program.

3 Appmainprog Status Notify To The Third Party Programs

Appmainprog system status changes are notified to the third party programs via notification, and no need response.

2.5 Current audio play done

Type	notification

Example	{
	"command": "/playback/play_done",
	"params": {
	"uri": <string>,
	"status": <int>,
	"error": {
	"code": <int>
	}
	}
	}

parameter description	status: 0. Normal completion 1. Abnormal completion error code（status=1）: HTTP request failed, corresponding code for HTTP return status code, such as 404,403; Timeout: -1
Behavior description	When the resource is played, notify the event to a third party program. Note that if the uri provided in the previous play method is invalid uri cause the play to fail, it should also send this message. After the third party receives the event, resend the play method.

2.6 TTS play done

Type	notification

Example	{
	"command": "/playback/play_tts_done",
	"params": {
	"tts_id": <int>,
	“status”:<int>
	}

	}
parameter description	status: 0. Normal completion 1. Abnormal completion
Behavior description	Notify the third party program when the TTS voice stream play done

2.7 System status change

Type	notification

Example	{
	"command": "/system/system_status_change",
	"params": {
	"status": <string>
	}
	}

parameter description	status: standby/normal
Behavior description	enter or exit standby

2.8 Player status change

Type	notification

Example	{
	"command": "/playback/player_status_change",
	"params": {
	“volume”:<int>,
	"status": <string>,
	“source”:<string>,
	“audioId”:<string>,
	“audioUid”:<string>,
	“audioSource”:<string>
	“audioName”:<string>
	“audioAnchor”:<string>
	“audioAlbum”:<string>
	“progress”:<int>,
	“audioExt”:<string>
	}

	}
parameter description	1、 volume:0~10 2、 state play/pause/stop 3、 source bluetooth sdk Device-side local SDK music cloud Server cloud resources none not playing 4、 audioId 5、 audioSource:xiami/ximalay 6、 progress: the current playback progress, in millisecond 7、 audioUid: Equipment terminal issued, need to return to the state, if it is bt play, set to "" 8、 audioExt: Equipment terminal issued, need to return to the state, if it is bt play, set to ""
Behavior description	Player play / pause / stop switch, music source switch

2.9 Network status change

Type	notification

Example	{
	"command": "/system/network_status_change",
	"params": {
	"quantity": <int>,
	“status”: <string>,
	“ssid”:<string>
	}
	}

parameter description	1、quantity: 0~100 Percentage of network signal strength 2、status: connect/disconnect 3、wired SSID is connected
Behavior description	network status changes need to be notified to the third party program

2.10 Bluetooth status change

Type	notification

Example	{
	"command": "/system/bluetooth_status_change",
	"params": {
	"status": <string>,
	“name”: <string>,
	“bt_paired_name”:<string>
	“bt_paired_mac”:<string>
	"role": <int>,
	}
	}

parameter description	1.status: power_on, power_off, connect, disconnect, pair_success, pair_failed, connect_timeout, inquiry _start, inquiry _end, 2.name: Bluetooth name 3.bt_paired_name：the name of the other device to which the device is paired 4. role: 0: BT_A2DP_ROLE_SRC, 1: BT_A2DP_ROLE_SINK, 2: BT_A2DP_ROLE_MAX,(invaild)
	Bluetooth status changes need to be notified to the third party program

4 The Key Event Notificate To The Third-party Program

3.5 Volume Increase

Type	notification

Example	{
	"command": "button",
	"params": {
	"name": "volume_inc"
	}
	}

parameter description	name: volume increase
Behavior description	increase the current speakers volume

3.6 Volume decrease

Type	notification

Example	{
	"command": "button",
	"params": {
	"name": "volume_dec"
	}
	}

parameter description	name: volume decrease
Behavior description	decrease the current speakers volume

3.7 Network config key

Type	notification

Example	{
	"command": "button",
	"params": {
	"name": "wifi_setup"
	}

	}
parameter description	name: network key ,use for wifi config
Behavior description	NULL

3.8 Factory reset key

Type	notification

Example	{
	"command": "button",
	"params": {
	"name": "factory_reset"
	}
	}

parameter description	name: factory reset
Behavior description	Long press to restore factory settings

3.9 Microphone mute key

Type	notification

Example	{
	"command": "button",
	"params": {
	"name": "mic_mute"
	}
	}

parameter description	name: microphone mute key
Behavior description	Press the microphone mute key , device do not accept voice input

3.10 Bluetooth reconnect key

Type	notification
Example	{ "command": "button",

	"params": {
	"name": "bt_reconnect"
	}
	}

parameter description	name: Bluetooth repair key
Behavior description	After a long press, the current pairing of Bluetooth is disconnected and reassembled

3.11 Self-test program start key

Type	notification

Example	{
	"command": "button",
	"params": {
	"name": "self_test"
	}
	}

parameter description	name: start self test program
Behavior description	Press the combination key, start the self-test program

3.12 Start the factory burner program

Type	notification

Example	{
	"command": "button",
	"params": {
	"name": "factory_ota"
	}
	}

parameter description	name: start factory ota
Behavior description	Press the combination key, start the factory OTA