Grandstream Networks, Inc.

GXP2200

GMI Web Service Guide V1.0

Grandstream Networks, Inc.

www.grandstream.com

Index

OVERVIEW 3

GMI WEB SERVICE INTERFACE 4

1. webServiceLogin(ip, username, password, callbackFunction) 4
2. getUptime(callbackFunction) 5
3. getPN(callbackFunction) 5
4. getHardware(callbackFunction) 6
5. getVendor(callbackFunction) 7
6. getProduct(callbackFunction) 7
7. getProductInfo(callbackFunction) 8
8. originateCall(account, isvideo isdialplan, destnum, headerstring, callbackFunction) 8
9. getNetworkInfo(callbackFunction) 9
10. getAccountInfo(callbackFunction) 10
11. getGroup(gpID, callbackFunction) 10
12. getContact(ctID, gpID, ctName, callbackFunction) 11
13. getGroupCount(callbackFunction).. 12
14. getContactCount(callbackFunction) 13
15. setGroup(phbkGroup, callbackFunction).. 13
16. setContact(phbkContact, callbackFunction) 14
17. removeContact(ctID, callbackFunction) 16
18. clearGroup(gpID, callbackFunction) 16
19. removeGroup(gpID, callbackFunction).. 17
20. moveToDefault(ctID, callbackFunction) 18
21. downloadPhonebook(phbkConfigure, flag, callbackFunction) 19
22. setPhonebook(phbkConfigure, flag, callbackFunction)................................ 21
23. getMessage(id, callFunction) 23
24. setNewMessage(num, account, text, flag, callbackFunction) 24
25. sendDraftMessage(id, callbackFunction) 24
26. removeMessage(id, flag, callbackFunction) 25
27. saveMessage(callbackFunction) 26
28. getLastCall(type, callbackFunction) 26
29. removeCall(id, flag, callbackFunction) 27
30. saveCallHistory(callbackFunction) 28
31. setUpgrade(upgradeConf, reboot, callbackFunction) 28
32. setParameter(confItem[ ], callbackFunction).. 31
33. getParameter(confItem[ ], callbackFunction).. 32

34. launchService(name, arg, callbackFunction) 33
35. getGMIVersion 33
36. getPhoneStatus 34
37. getPhoneMem 35
38. getLineStatus 35

GMIService EXAMPLE 37

USING JAVASCRIPT 37
USING MFC 38

OVERVIEW

GMI (Grandstream Manager Interface) is a management API developed by Grandstream Networks, Inc. Designed for our IP Multimedia phones, it allows partners to develop customized applications on the phone. GMI supports standard HTML/CSS/Javascript, users can use these dynamic web page development languages for the customized applications and GMI will display the application on the phone based on the web pages generated.

The number of GMI web service interfaces is growing with increasing customer demands. Some interfaces in this guide are newly introduced after certain GMI version. To avoid compatibility issue, we added **getGMIVersion** interface [35. getGMIVersion] for users to retrieve current GMI interface version. For interfaces that require specific GMI version, please use **getGMIVersion** to match the returned version number. For interfaces that do not have GMI version requirement, they exist in all GMI versions and there is no need to identify GMI version.

Note:

Before starting the GMI Web Service development, please upgrade your GXP2200 to the latest firmware version. The firmware release information can be found in the following link:

http://www.grandstream.com/support/firmware

GMI WEB SERVICE INTERFACE

GXP2200 GMI Web Service provides basic API to facilitate users calling the existing application on the GXP2200 or to retrieve the phone's status. Usually JavaScript and Windows MFC can be used for development. However, no matter what programming language is used, the functions in the interface have the same name and parameters, which are included in **GMIService** class. Users could define the interface according to the parameters and HTTP request. This section describes details for each interface.

Note:

- Parameter **account** in each interface represents the account index. This is the index number retrieved from **getAccountInfo** [10. getAccountInfo(callbackFunction)] interface. The valid index number is from 0 to 5 for account 1 to account 6.

- For the HTTP requests in each interface introduced below, if the action data is empty or incorrect, the following message will be returned:
  "{"res": "error", "msg": "command not found"}"

- For the interfaces introduced below, if the HTTP requests' interval is more than 900s, the request will be disconnected. The following message will be returned and users will need login again.
  "{"res": "error", "msg": "authentication required"}"

1. webServiceLogin(ip, username, password, callbackFunction)

Description	Login GXP2200 web GUI with IP address, username and password. This interface has to be executed first before using other interfaces (except **getVendor**, **getProduct** and **getProductInfo** interfaces)
Parameters	• ip: Phone's IP address • username: Web GUI login username • password: Web GUI login password • callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "login" • username: "admin" • secret: "123" • format: "json"

	• jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=login&username=admin&secret=123&format=json&jsoncallback=?
Return	1. HTTP request return: • Login successful "{"res": "success", "msg" : "authentication accepted"}" • Login failed if username or password is missing or incorrect "{"res": "error", "msg" : "authentication failed"}" 2. callbackFunction(data) will process the data returned from HTTP request

2. getUptime(callbackFunction)

Description	Retrieve current system up time
Parameters	callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "uptime" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=uptime&format=json&jsoncallback=?
Return	1. HTTP request return: • Return successful "{"res": "success", "day": "1", "hour" : "12", "min" : "23", "sec" : "45"}" • Return failed "{"res": "error", "msg": "can't get uptime"}" 2. callbackFunction(data) will process the data returned from HTTP request

3. getPN(callbackFunction)

Description	Retrieve GXP2200 P/N number
Parameters	callbackFunction: Call back function

HTTP Request	• url: "http://"+ ip + "/manager?" • action: "pn" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=pn&format=json&jsoncallback=?
Return	1. HTTP request return: • Return successful "{"res": "success", "pn": "9620001902A"}" • Return failed "{"res": "error", "msg" : "can't get pn version"}" 2. callbackFunction(data) will process the data returned from HTTP request

4. getHardware(callbackFunction)

Description	Retrieve GXP2200 Hardware version
Parameters	callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "hardware" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=hardware&format=json&jsoncallback=?
Return	1. HTTP request return: • Return successful "{"res": "success", "hardware": "V2.0A"}" • Return failed "{"res": "error", "msg" : "can't get hardware version"}" 2. callbackFunction(data) will process the data returned from HTTP request

5. getVendor(callbackFunction)

Description	Retrieve GXP2200 vendor information (login is not required)
Parameters	callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "vendor" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=vendor&format=json&jsoncallback=?
Return	1. HTTP request return: • Return successful "{"res": "success", "vendor": "Grandstream Networks, Inc."}" • Return failed "{"res": "error", "msg" : "can't get vendor"}" 2. callbackFunction(data) will process the data returned from HTTP request

6. getProduct(callbackFunction)

Description	Retrieve GXP2200 product information (login is not required)
Parameters	callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "vendor" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=product&format=json&jsoncallback=?
Return	1. HTTP request return: • Return successful "{"res": "success", "product": "GXP2200"}" • Return failed "{"res": "error", "msg" : "can't get product model"}" 2. callbackFunction(data) will process the data returned from HTTP request

7. getProductInfo(callbackFunction)

Description	Retrieve GXP2200 vendor and product information (login is not required)
Parameters	callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "productinfo" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=productinfo&format=json&jsoncallback=?
Return	1. HTTP request return: • Return successful "{"res": "success", "product": "GXP2200","vendor": "Grandstream Networks, Inc."}" • Return failed "{"res": "error", "msg" : "can't get product information"}" 2. callbackFunction(data) will process the data returned from HTTP request

8. originateCall(account, isvideo isdialplan, destnum, headerstring, callbackFunction)

Description	Make calls
Parameters	• account: account index (int type, the index value can be retrieved from **getAccountInfo** interface) • isvideo: audio call or video call. 0 - audio; 1 - video • isdialplan: 0 or 1 • destnum: the number to be dialed • headerstring: SIP request header • callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "originatecall" • destnum: "8108819" • account: 0 • isvideo: 0 • isdialplan: 0 • headerstring: "" • format: "json"

	• jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=originatecall&destnum=8108819&account=0&isvideo=0&isdialplan=0&headerstring=&format=json&jsoncallback=?
Return	HTTP request return: • Call originated "{"res": "success", "msg": "call originated"}" • Destination number is empty "{"res": "error", "msg" : "destnum can't be null"}" • Timeout "{"res": "error", "msg" : "timeout"}" • callbackFunction(data) will process the data returned from HTTP request

9. getNetworkInfo(callbackFunction)

Description	Retrieve GXP2200 network status
Parameters	callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "network" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=network&format=json&jsoncallback=?
Return	1. HTTP request return: "{"res" : "success", "mac" : "00-0b-82-29-5c-a7", "ip" : "192.168.121.230", "mask" : "255.255.255.0", "gateway" : "192.168.121.1", "dns" : "192.168.121.253", "type" : "dhcp"}" 2. callbackFunction(data) will process the data returned from HTTP request

10. getAccountInfo(callbackFunction)

Description	Retrieve GXP2200 account information
Parameters	callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "accountStatus" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=status&format=json&jsoncallback=?
Return	1. HTTP request return: {"Response":"Success","Data":[{"Index":"1","Server":"192.168.120.254","Number":"3882","Status":"1","Activate":"1"}, {"Index":"0","Server":"sip.ipvideotalk.com:48879","Number":"8118263","Status":"1","Activate":"1"}, {"Index":"2","Server":"192.168.121.22","Number":"1001","Status":"1","Activate":"1"}]} 2. String definition: • Index: The account index. From 0 to 5 for account 1 to account 6. This is the account index used in **originateCall** function. • Server: The SIP server address for the account. • Number: The account number. • Status: account status. 0 - unregistered; 1 and 2 - registered; 3 and 5 - dialing; 4 - ringing; 6 - talking • Activate: account activated/deactivated. 1 - activated; 0 - deactivated

11. getGroup(gpID, callbackFunction)

Description	Retrieve phonebook group information for group gpID. If gpID is empty, return information for all groups
Parameters	• gpID: phonebook group ID • callbackFunction: Call back function

HTTP Request	• url: "http://"+ ip + "/manager?" • action: "getgroup" • groupID: "" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=getgroup&groupID=&format=json&jsoncallback=?
Return	1. HTTP request return: • Return successful "{"res": "success", "msg": [{"groupID":"0","groupName":"default"}, {"groupID":"100","groupName":"blacklist"}]}" • No groups "{"res": "error", "msg" : "0"}" • Return failed "{"res": "error", "msg" : "can't get groups information"}" • Parameters in wrong format (GMI version 9 or newer) "{"res": "error", "msg" : "wrong format"}" 2. callbackFunction(data) will process the data returned from HTTP request

12. getContact(ctID, gpID, ctName, callbackFunction)

Description	1. Retrieve contact information • If ctID is specified but ctName is empty, return the contact ctID • If ctName is specified, return the contacts which has string ctName in its name field • If ctID and ctName are empty but gpID is specified, return all the contacts in group gpID • If ctID, ctName and gpID are empty, return all contacts 2. Before using **getContact** interface, users could get the number of contacts by **getContactCount** interface first. If the number of contacts is more than 20, it is recommended to use **setPhonebook** interface instead of **getContact** interface
Parameters	callbackFunction: Call back function

HTTP Request	• url: "http://"+ ip + "/manager?" • action: "getcontact" • contactID: "" • groupID: "" • contactName: "" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=getcontact&contactID=&groupID=&contactName=&format=json&jsoncallback=?
Return	1. HTTP request return: • Return successful "{"res": "success", "msg": [{"contactID":"0","contactName":"xxx","groupID":"0","work":"8819","home":"", "mobile":"","fax":"","photo":"","email":""}, {“contactID":"1","contactName":"grandstream","groupID":"0","work":"3587","home":"" , "mobile":"","fax":"","photo":"","email":""}]}" • No contacts "{"res": "error", "msg" : "0"}" • Return failed "{"res": "error", "msg" : "can't get contacts information"}" • Parameters in wrong format (GMI version 9 or newer) "{"res": "error", "msg" : "wrong format"}" 2. callbackFunction(data) will process the data returned from HTTP request

13. getGroupCount(callbackFunction)

Description	Retrieve the number of groups in phonebook
Parameters	callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "getgroupcount" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=getgroupcount&format=json&jsoncallback=?

Return	1. HTTP request return: - Return successful "{"res": "success", "msg": "3"}" - Return failed "{"res": "error", "msg" : "can't get group count"}" 2. callbackFunction(data) will process the data returned from HTTP request

14. getContactCount(callbackFunction)

Description	Retrieve the number of contacts in phonebook
Parameters	callbackFunction: Call back function
HTTP Request	- url: "http://"+ ip + "/manager?" - action: "getcontactcount" - format: "json" - jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) - For example: http://192.168.121.230/manager?action=getcontactcount&format=json&jsoncallback=?
Return	1. HTTP request return: - Return successful "{"res": "success", "msg": "9"}" - Return failed "{"res": "error", "msg" : "can't get contact count"}" 2. callbackFunction(data) will process the data returned from HTTP request

15. setGroup(phbkGroup, callbackFunction)

Description	- Configure phonebook group information (edit group and add group)
	- Phonebook group structure
	PhbkGroup
	{
	gpID; //Group ID, string
	gpName; //Group name, string
	}
	- If gpID is empty, the group with gpID will be added. If gpID is existed, group gpID will be edited.

Parameters	• PhbkGroup: Phonebook group information • callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "setgroup" • groupInfo: gpID::gpName • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=setgroup&groupInfo=1::grandstream&format=json&jsoncallback=?
Return	1. HTTP request return: • Return successful "{"res": "success", "msg": "1"}" //gpID is "1" • Return failed "{"res": "error", "msg" : "can't set group"}" • groupInfo is empty "{"res" : "error", "msg": "group information is empty" }" • Group setup failed "{"res" : "error", "msg": "set group failed" }" • groupInfo wrong format "{"res" : "error", "msg": "wrong format" }" • Database operation error (GMI version 9 or newer) "{"res" : "error", "msg": "0" }" 2. callbackFunction(data) will process the data returned from HTTP request

16. setContact(phbkContact, callbackFunction)

Description	• Configure phonebook contact information (edit contact and add contact)
	• Phonebook contact structure
	PhbkContact
	{
	ctID; //Contact ID
	ctName; //Contact name
	gpID; //Group ID
	work; //Work number
	home; //Home number

	mobile; //Mobile number
	fax; //Fax number
	mail; //Email address
	}
	• If ctID is empty, contact will be added with a ctID in the returned info. If ctID is existed, contact ctID will be edited
	• If contacts need to be added in batch, it is recommended to use **setPhonebook** interface to import the contacts

Parameters	• PhbkContact: Contact information • callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "setcontact" • contactInfo: ctID::ctName::gpID::work::home::mobile::fax::mail • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=setcontact&contactInfo=1::grandstream::0::5211::2512::1234::0120::::&format=json&jsoncallback=?
Return	1. HTTP request return: • Return successful "{"res": "success", "msg": "1"}" //ctID is "1" • Return failed "{"res": "error", "msg" : "can't set contact"}" • "contactInfo" is empty "{"res" : "error", "msg": "contact information is empty" }" • Contact setup failed "{"res" : "error", "msg": "set contact failed" }" • "contactInfo" wrong format "{"res" : "error", "msg": "wrong format" }" • Database operation error (GMI version 9 or newer) "{"res" : "error", "msg": "0" }" 2. callbackFunction(data) will process the data returned from HTTP request

17. removeContact(ctID, callbackFunction)

Description	Remove contact ctID from phonebook. If ctID is empty, remove all contacts
Parameters	• ctID: Contact ID • callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "removecontact" • contactID: "1" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=removecontact&contactID=1&format=json &jsoncallback=?"
Return	1. HTTP request return: • Remove successful "{"res": "success"}" • Remove failed "{"res" : "error", "msg": "remove contact failed" }" • Return data failed "{"res": "error", "msg" : "can't remove contact"}" • Parameters in wrong format (GMI version 9 or newer) "{"res" : "error", "msg": "wrong format" }" • Database operation error (GMI version 9 or newer) "{"res" : "error", "msg": "0" }" 2. callbackFunction(data) will process the data returned from HTTP request

18. clearGroup(gpID, callbackFunction)

Description	Clear contacts in the group. All the cleared contacts in the group will be moved to the default group
Parameters	• gpID: Group ID • callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "cleargroup" • groupID: gpID • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in

	AJAX and JQuery) - For example: http://192.168.121.230/manager?action=cleargroup&groupID=1&format=json&jsoncallback=?
Return	1. HTTP request return: - Clear successful "{"res": "success"}" - Clear failed "{"res" : "error", "msg": "clear group failed" }" - Return data failed "{"res": "error", "msg" : "can't clear group"}" - "groupID" is empty "{"res" : "error", "msg": "ID is empty" }" 2. callbackFunction(data) will process the data returned from HTTP request
Note	The default group cannot be cleared. The ctID of the default group is 0

19. removeGroup(gpID, callbackFunction)

Description	Remove phonebook group according to group ID. If gpID is empty, delete all the groups.
Parameters	- gpID: Group ID - callbackFunction: Call back function
HTTP Request	- url: "http://"+ ip + "/manager?" - action: "removegroup" - groupID: gpID - format: "json" - jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) - For example: http://192.168.121.230/manager?action=removegroup&groupID=1&format=json&jsoncallback=?

Return	1. HTTP request return: - Remove successful "{"res": "success"}" - Remove failed "{"res" : "error", "msg": "remove group failed" }" - Return data failed "{"res": "error", "msg" : "can't remove group"}" - Parameters in wrong format (GMI version 9 or newer) "{"res" : "error", "msg": "wrong format" }" - Database operation error (GMI version 9 or newer) "{"res" : "error", "msg": "0" }" 2. callbackFunction(data) will process the data returned from HTTP request
Note	The default group and blacklist cannot be removed. The gpID of the default group is 0; the gpID of the blacklist is 100.

20. moveToDefault(ctID, callbackFunction)

Description	Move the contact to default phonebook group
Parameters	- ctID: Contact ID - callbackFunction: Call back function
HTTP Request	- url: "http://"+ ip + "/manager?" - action: "movetodefault" - contactID: ctID - format: "json" - jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) - For example: http://192.168.121.230/manager?action=movetodefault&contactID=1&format=json&jsoncallback=?
Return	1. HTTP request return: - Move successful "{"res": "success"}" - Move failed "{"res" : "error", "msg": "move contact to default failed" }" - Return data failed "{"res": "error", "msg" : "can't move contact to default"}" //dbus returns null - "contactID" is empty

	"{"res" : "error", "msg": "ID is empty" }" - Parameters in wrong format (GMI version 9 or newer) "{"res" : "error", "msg": "wrong format" }" - Database operation error (GMI version 9 or newer) "{"res" : "error", "msg": "0" }" 2. callbackFunction(data) will process the data returned from HTTP request
Note	If the ctID is the default group ID, it won't be processed.

21. downloadPhonebook(phbkConfigure, flag, callbackFunction)

Description	- Set up phonebook download configurations and download phonebook.xml
	- Phonebook download configuration structure
	phbkConfigure
	{
	mode; //Download mode. 0 - OFF; 1 - TFTP; 2 - HTTP
	url; //Phonebook download URL
	interval; //Automatic download interval
	redup; //replace duplicate items. 0 - No; 1 - Yes
	clearold; //Clear old list. 0 - No; 1 - Yes
	filetype; //Phonebook file format. 0 - XML; 2 - CSV
	encode; //Phonebook file encoding format. UTF-8 or GBK
	}

Parameters	- phbkConfigure: Phonebook download configuration - confItem: nvram configuration - flag(int): 0 - save phonebook configuration; 1 - save phonebook configuration and download phonebook - callbackFunction: Call back function
HTTP Request	- url: "http://"+ ip + "/manager?" - action: "putdownphbk" - flag: "1" (Save the configuration and download phonebook) - downMode: "2" (Download mode. 0: OFF, 1: TFTP, 2: HTTP) - downUrl: Download phonebook URL - downInterval: "0" (0: no automatic download) - downReplace: "0" (replace duplicate items, 0: No, 1: Yes) - downClear: "1" (clear old list, 0: No, 1: Yes) - downEncode: "UTF-8" (phonebook file encoding format) - format: "json"

	• jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=putdownphbk&flag=1downMode=2&downUrl=http://192.168.121.29/WebService/Phonebook.xml&downInterval=0&downReplace=0&downClear=1&downEncode=UTF-8&downtype=0&format=json&jsoncallback=?
Return	1. HTTP request return: • Phonebook configuration saved "{"res": "success", "flag":"1"}" • Phonebook configuration cannot be saved "{"res" : "error", "msg" : "configuration file not found"}" • GXP2200 is rebooting "{"res": "error", "msg" : "phone rebooting"}" 2. callbackFunction(data) will process the data returned from HTTP request
Note	• In the HTTP request, if flag=1, when the phone is downloading and importing phonebook, users could send HTTP request to check if the downloading/importing is done • For example: http://192.168.121.230/manager?action=phbkresponse&format=json • Return: "{"res": "success", "phbkresponse":"0"}" • phbkresponse value: 0: Download successful (for downloading and importing) 1: Downloading 2: Contacts full (for downloading and importing) 3: Parse failed (For downloading and importing) 4: Downloading failed 5: Downloading mode is OFF 6: Download address invalid 8: File encoding format wrong 9: Importing (For importing) 10: File format invalid (for importing)

22. setPhonebook(phbkConfigure, flag, callbackFunction)

Description	• Phonebook import/export
	• Phonebook configuration structure
	phbkConfigure
	{
	mode; //Download mode. 0 - OFF; 1 - TFTP; 2 - HTTP
	url; //Phonebook download URL
	interval; //Automatic download interval
	redup; //Replace duplicate items. 0 - No; 1 - Yes
	clearold; //Clear old list. 0 - No; 1 - Yes
	filetype; //Phonebook file format. 0 - XML; 2 - CSV
	encode; //Phonebook file encoding format. UTF-8 or GBK
	}

Parameters	• phbkConfigure: Phonebook download configuration • flag: For import/export • callbackFunction: Call back function
HTTP Request	1. Export phonebook (1) Save configuration • url: "http://"+ ip + "/manager?" • action: "putportphbk" • flag: "0" • opmode: "1" • portEncode: "UTF-8" • portType: "0" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=putportphbk&opmode=1&flag=0&portType=0&portEncode=UTF-8&format=json&jsoncallback=? (2) Save phonebook.xml • url: "http://"+ ip + "/manager?" • action: "savephbk" • format: "json" • jsoncallback ——"?" • For example, http://192.168.121.230/manager?action=savephbk&format=json&jsoncallback=? (3) When "savephbk" succeed, the phoneook.xml will be exported to the phone.then you can download the phonebook.xml from link as follow:

	http://192.168.121.230/phonebook/phonebook.xml 2. Import phonebook (1) Upload the local phonebook.xml to the phone first. Then use the HTTP POST method to send the phonebook content. HTTP request header: http://192.168.121.230/upload?type=phonebook&format=json (2) Import phonebook • url: "http://" + ip +"/manager?" • action: "putportphbk" • flag: "1" • opmode: "0" • portEncode: "UTF-8" • portType: "0" • portReplace: "0" • portClear: "1" • format: "json" • jsoncallback: "?" • For example: http://192.168.121.230/manager?action=putportphbk&opmode=0&flag=1&portType=0&portEncode=UTF-8&portReplace=0&portClear=1&format=json&jsoncallback=?
Return	1. HTTP request return: • "putportphbk": Import successful "{"res": "success", "flag":"1"}" • "putportphbk": Import failed "{"res" : "error", "msg" : "configuration file not found"}" • "savephbk": Save successful "{"res": "success"}" 2. callbackFunction(data) will process the data returned from HTTP request
Note	1. If the HTTP request returns "savephbk" successfully, users could send HTTP requests to check whether the phonebook.xml saving process is done. For example: http://192.168.121.230/manager?action=portphbkresponse&format=json Return: • Saving done: "{"res": "success", "phbkresponse":"0"}" • Saving in process: "{"res": "success", "phbkresponse":"1"}" 2. If the HTTP request returns "putportphbk" successfully, users could send HTTP requests to check whether the phonebook.xml import process is done. For example:

	http://192.168.121.230/manager?action=phbkresponse&format=json Return: • Import done: "{"res": "success", "phbkresponse":"0"}" • Importing in process: "{"res": "success", "phbkresponse":"1"}" • Contacts full "{"res": "success", "phbkresponse":"2"}" • Parse failed "{"res": "success", "phbkresponse":"3"}" • Import failed "{"res": "success", "phbkresponse":"4"}" • Phone is processing "{"res": "success", "phbkresponse":"9"} • Invalid file "{"res": "success", "phbkresponse":"10"}"

23. getMessage(id, callFunction)

Description	Get message according to message id. When id=-1, get the last message
Parameters	• id: Message ID • callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "getmessage" • id: "-1" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=getmessage&id=-1&format=json&jsoncallback=?
Return	1. HTTP request return: • Successful "{"res": "success", "msg" : [{"id" : "15", "number" : "8107314", "account" : "0", "state", "1", "message" : "hello"}]}" • Failed "{"res" : "error", "msg" : "get message failed"}" • Parameters in wrong format (GMI version 9 or newer) "{"res" : "error", "msg" : "wrong format"}"

24. setNewMessage(num, account, text, flag, callbackFunction)

Description	Send message; save message
Parameters	• num: The number to send the message to • account: Account index • text: Message content • flag: Save or send message. 0 - save message to Draft box; 1 - send message • callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "setnewmessage" • number: "8107314" • account: "0" • content: "hello" • flag: "1" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=setnewmessage&number=8107314&account=0&content=hello&flag=1&format=json&jsoncallback=?
Return	1. HTTP request return: • Successful "{"res": "success"}" • Failed "{"res" : "error", "msg" : "set new message failed"}" • Parameters in wrong format (GMI version 9 or newer) "{"res" : "error", "msg" : "wrong format"}" • Database operation error (GMI version 9 or newer) "{"res" : "error", "msg" : "0"}"

25. sendDraftMessage(id, callbackFunction)

Description	Send message from Draft box. After the message is sent, it will be moved from Draft box to Outbox
Parameters	• id: Message ID • callbackFunction: Call back function

HTTP Request	• url: "http://"+ ip + "/manager?" • action: "senddraftmessage" • messageID: "12" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=senddraftmessage&messageID=12&format=json&jsoncallback=?
Return	HTTP request return: • Send successful "{"res": "success"}" • Send failed "{"res" : "error", "msg" : "send new message failed"}"

26. removeMessage(id, flag, callbackFunction)

Description	Remove message
Parameters	• flag: Remove type. 1 - remove all messages from Inbox, Outbox or Draft box; 2 - remove one message • id: Message identifier. flag=1, id=0 - remove all messages in Inbox flag=1, id=1 - remove all messages in Outbox flag=1, id=2 - remove all messages in Draftbox flag=0, id is the message ID - remove message with message ID = id • callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "removemessage" • ID: "1" • flag: "1" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=removemessage&ID=1&flag=1&format=json&jsoncallback=?
Return	1. HTTP request return: • Remove successful "{"res": "success"}"

	• Remove failed "{"res" : "error", "msg" : "remove message failed"}" • Parameters in wrong format (GMI version 9 or newer) "{"res" : "error", "msg" : "wrong format"}"

27. saveMessage(callbackFunction)

Description	Save message.xml
Parameters	callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "savemessage" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=savemessage&format=json&jsoncallback=?
Return	HTTP request return: • Save successful "{"res": "success"}"
Note	After HTTP request returns saving successful, users could download message.xml to the phone using the following link: http://192.168.121.230/message/message.xml

28. getLastCall(type, callbackFunction)

Description	Get the last call log (answered calls, dialed calls or missed calls) entry information
Parameters	• type: Call log type. 0 - answered calls; 1 - dialed calls; 2 - missed calls • callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "getlastcall" • type: "1" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery)

	- For example: http://192.168.121.230/manager?action=getlastcall&type=1&format=json&jsoncallback=?
Return	HTTP request return: - Successful "{"res": "success"， "msg" : [{"id" : "15", "number" : "8107314", "account" : "0", "isVideo", "0"}]}" - Failed "{"res" : "error", "msg" : "get last call failed"}" - "type" wrong format "{"res" : "error", "msg" : "wrong type format"}"

29. removeCall(id, flag, callbackFunction)

Description	Remove call log
Parameters	- flag: Remove type. 1 - remove all the logs in answered calls, dialed calls or missed calls; 2 - remove one call - id: Call log identifier flag=1, id=0 - remove all answered call log flag=1, id=1 - remove all dialed call log flag=1, id=2 - remove all missed call log flag=0, id is the log ID - remove the log with id =ID - callbackFunction: Call back function
HTTP Request	- url: "http://"+ ip + "/manager?" - action: "removeall" - ID: "1" - flag: "1" - format: "json" - jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) - For example: http://192.168.121.230/manager?action=removecall&ID=1&flag=1&format=json&jsoncallback=?
Return	1. HTTP request return: - Remove successful "{"res": "success"}" - Remove failed "{"res" : "error", "msg" : "remove call failed"}" - Parameters in wrong format (GMI version 9 or newer)

	"{"res" : "error", "msg" : "wrong format"}"

30. saveCallHistory(callbackFunction)

Description	Save callhistory.xml
Parameters	callbackFunction: Call back function
HTTP Request	• url: "http://"+ ip + "/manager?" • action: "savecallhistory" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=savecallhistory&format=json&jsoncallback=?
Return	HTTP request return: • Save successful "{"res": "success"}"
Note	After HTTP request returns saving successful, users could download callhistory.xml to the phone with the following link: http://192.168.121.230/callhistory/callhistory.xml

31. setUpgrade(upgradeConf, reboot, callbackFunction)

Description	• Set up upgrading configuration
	• Configuration item structure. Each configuration item contains two fields with the following format:
	ConfItem
	{
	value; //P value
	key; //nvram address (P parameter)
	}
	• The ConfItem list is as follows:
	UpgradeConf
	{
	ConfItem updaterule //nvram address: 238
	//0 - Always check for new firmware; 1 - check
	//new firmware when FW prefix/suffix changes;

```
                                //2 - Always skip firmware check
    ConfItem updatevia          //nvram address: 212
                                //Upgrade via. 0 - TFTP; 1 - HTTP; 2 - HTTPS
    ConfItem firpath            //nvram address: 192
                                //Firmware server path
    ConfItem confpath           //nvram address: 237
                                //Config server path
    ConfItem firpre             //nvram address: 232
                                //Firmware prefix
    ConfItem firpost            //nvram address: 233
                                //Firmware suffix
    ConfItem confpre            //nvram address: 234
                                //Config file prefix
    ConfItem confpost           //nvram address: 235
                                //Config file suffix
    ConfItem dhcp66             //nvram address: 145
                                //Use Option 66. 0 - No; 1 - Yes
    ConfItem autoup             //nvram address: 194
                                //Automatic upgrade. 0 - No;
                                //1 - Check every day; 2 - Check every week;
                                //3 - Check every specific period
    ConfItem keypadup           //nvram address: 88
                                //Lock keypad update. 0 - No; 1 - Yes
    ConfItem authconffile       //nvram address: 240
                                //Authenticate config file. 0 - No; 1 - Yes
    ConfItem hourofday          //nvram address: 285
                                //Hour of the day
    ConfItem dayofweek          //nvram address: 286
                                //Day of the week
    ConfItem peroid             //nvram address: 193
                                //Automatic upgrade period (in minutes)
    ConfItem xmlpass            //nvram address: 1359
                                //XML Config file password
    ConfItem httpuser           //nvram address: 1360
                                //HTTP/HTTPS username
    ConfItem httppass           //nvram address: 1361
                               //HTTP/HTTPS password
   }
```

Parameters	• UpgradeConf: Upgrade configuration • Reboot: 0 - Save configuration without reboot; 1 - Yes, reboot

	• callbackFunction: Call back function
HTTP Request	1. Save configuration • url: http://" + ip +"/manager?" • action: "put" • var-0000: 238 //Key. P238 is for automatic upgrade val-0000: 0 //P238 Value var-0001: 212 //Key. P212 is for upgrade method val-0001: 1 //P212 value .... var-xxxx val-xxxx • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=put&var-0000=238&val-0000=0&var-0001=212&val-0001=1&format=json&jsoncallback=? 2. Upgrade • url: http://" + ip +"/manager?" • action: "upgrade" • var-0000: 238 //Key. P238 is for automatic upgrade val-0000: 0 //P238 Value var-0001: 212 //Key. P212 is for upgrade method val-0001: 1 //P212 value .... var-xxxx val-xxxx • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=upgrade&var-0000=238&val-0000=0&var-0001=212&val-0001=1&format=json&jsoncallback=?

Return	1. HTTP request return: • Upgrade setup successful "{"res": "success", "flag":"1"}" • Upgrade setup failed "{"res": "error", "msg":"phone rebooting"}" 2. callbackFunction(data) will process the data returned from HTTP request

32. setParameter(confItem[ ], callbackFunction)

Description	• Phone's configuration (set up video, call and other parameters)
	• Configuration item structure. Each configuration item contains two fields with the following format:
	ConfItem
	{
	value; //P value
	key; //nvram address (P parameter)
	}

Parameters	• ConfItem: Phone's configuration (array) • callbackFunction: Call back function
HTTP Request	• url: http://" + ip +"/manager?" • action: "put" • var-0000: val1 //Key val-0000: var1 //Value var-0001: val2 //Key val-0001: var2 //Value .... var-xxxx val-xxxx • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=put&var-0000=7044&val-0000=0&var-0001=904&val-0001=15&format=json&jsoncallback=?

Return	1. HTTP request return: - Configure successful "{"res": "success", "flag":"1"}" - Configure failed "{"res": "error", "msg":"phone rebooting"}" 2. callbackFunction(data) will process the data returned from HTTP request

33. getParameter(confItem[ ], callbackFunction)

Description	- Get phone's configuration (set up video, call and other parameters)
	- Configuration item structure. Each configuration item contains two fields with the following format:
	ConfItem
	{
	value; //P value
	key; //nvram address (P parameter)
	}

Parameters	- ConfItem: Phone's configuration (array) - callbackFunction: Call back function
HTTP Request	- url: http://" + ip +"/manager?" - action: "get" - var-0000: val1 //Key var-0001: val2 //Key .... var-xxxx - format: "json" - jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) - For example: http://192.168.121.230/manager?action=get&var-0000=7044&var-0001=904&format=json&jsoncallback=?
Return	1. HTTP request return: - Successful "{"res": "success", "7044":"1", "904":"15"}" - Failed "{"res": "error", "msg":"new memory failed or no value"}" 2. callbackFunction(data) will process the data returned from HTTP request

34. launchService(name, arg, callbackFunction)

Description	• Open an application. • GXP2200 applications: AlarmClock Audio Browser Message Video
Parameters	• name: Application name (case insensitive) • args: Input parameters • callbackFunction: Call back function
HTTP Request	• url: http://" + ip +"/manager?" • action: "launchservice" • name: name • args: http://192.168.121.29/a.html • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=launchservice&name=Browser&arg=http://192.168.121.29/a.html&format=json&jsoncallback=?
Return	1. HTTP request return: • Launch successful "{"res": "success"}" • Server cannot get data. Open failed "{"res": "error", "msg" : "open application failed"}" //dbus returns null • "name" is empty "{"res" : "error", "msg": "name is empty" }" • Run application failed "{"res" : "error", "msg": "run application failed" }" 2. callbackFunction(data) will process the data returned from HTTP request

35. getGMIVersion

Description	Get phone's GMI version number
Parameters	N/A

HTTP Request	• url: http://" + ip +"/manager?" • action: "gmiVersion" • For example: http://192.168.121.230/manager?action=gmiVersion
Return	GMI Version number. For example, 6. The version number increases by 1 on each new GMI version.

36. getPhoneStatus

Description	Get phone's process status (GMI version 6 or newer)
Parameters	• funcname: Process name
HTTP Request	• url: http://" + ip +"/manager?" • action: "phoneStatus" • funcname: process name • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=phoneStatus&funcname=gs_gui http://192.168.121.230/manager?action=phoneStatus&funcname=gs_gui&format=json&jsoncallback=?
Return	• If the return data is retrieved in json format and there is jsoncallback, the return data is as follows: **jsoncallback(S)** where **jsoncallback** is the callback function name • Otherwise, the process status will be returned directly, for example, S. The status are listed as below: R: running S: sleeping in an interruptible wait D: waiting in uninterruptible disk sleep Z: zombie T: traced or stopped (on a signal) W: paging Unknown: unknown status or cannot get status return data

37. getPhoneMem

Description	Get phone's memory status (GMI version 6 or newer)
Parameters	• funcname: Process name
HTTP Request	• url: http://" + ip +"/manager?" • action: "phoneMem" • funcname: process name • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=phoneStatus&funcname=gs_gui http://192.168.121.230/manager?action=phoneStatus&funcname=gs_phone&format=json&jsoncallback=?
Return	• If the return data is retrieved in json format and there is jsoncallback, the return data is as follows: **jsoncallback(S)** where **jsoncallback** is the callback function name • Otherwise, the physical memory of the process will be returned. For example, 3176 kB • If the process memory is unknown or cannot get memory, "Unknown" will be returned.

38. getLineStatus

Description	Get phone's line status (GMI version 8 or newer)
Parameters	• line: Line index number. From 0 to 2 for line 1 to line 3
HTTP Request	• url: http://" + ip +"/manager?" • action: "lineStatus" • format: "json" • jsoncallback: "?" (This is the HTTP request for the call back function, usually used in AJAX and JQuery) • For example: http://192.168.121.230/manager?action=lineStatus&line=0
Return	1. If the return data is retrieved in json format and there is jsoncallback, the return data is as follows: **jsoncallback(S)** where **jsoncallback** is the callback function name

	2. Otherwise, the line status value will be returned. • -1: Retrieve line status failed • 0: CALL_IDLE. The line is in idle and users can make calls • 1: CALL_DIALING. Dialing • 2: CALL_RINGING. Ringing • 3: CALL_CALLING. Calling • 4: CALL_CONNECTED. Connected • 5: CALL_ONHOLD. Call on hold • 6: CALL_TRANSFERED. Call is already transferred • 7: CALL_ENDING. Call is being ended • 8: CALL_FAILED. Call cannot be established • 9: CALL_TRANSFER. Call is being transferred • 10: CALL_CONFERENCE. In conference • 11: CALL_PAGING • 12: CALL_RINGBACK. Ringback • 13: CALL_IPCALL. Direct IP Call 3. Other return data • "line" cannot be empty {"res": "error", "msg": "line can't be null"} • "line" value is not valid {"res": "error", "msg": "line invalid"} • Cannot get line status {"res": "error", "msg": "can't get line status"}

GMIService EXAMPLE

USING JAVASCRIPT

Add the following code in the file *webservice.html*.

```
<script language="javascript" src="js/WebService.js"></script>
<script type="text/javascript">
function showLogin(data)
{
    if(data.res  == "success")
    {
        $("#connectresult").html("<font color='red'>Connect success</font>");
    }
}
function login()
{
    ip = $("#ip").val();
    username = $("#username").val();
    password = $("#password").val();
    webService.webServiceLogin(ip, username, password, showLogin);
}
</script>
...
<div id="connectinfo">
Phone IP: <input id="ip" value="Input Phone IP here">
UserName: <input id="username" value="admin">
Password: <input id="password" value="admin">
<button onclick="login()">Connect</button>
</div>
<br>
<h4>2. Connect info</h4>
<div id="connectresult">
not connected
</div>
...
```

USING MFC

Include header "GMIService.h" first. Then load the lib file.

- Static load:

```
#pragma comment(lib,"GMI.lib")
```

- Dynamic load:

```
HINSTANCE hInst;
hInst=LoadLibrary("GMI.dll");                    //load dll
typedef void  (*ADDPROC)( CallBackFun fun);
ADDPROC Add=(ADDPROC)GetProcAddress(hInst,"getAllGrouops");
Add(userFun);
```

Note:

If callback function **userFun** is used in the class, it can be declared as static function or global function.