Configuration RESTful Service

General URL structure

Common prefix:

```
https://host.example.com/sipxconfig/rest
```

Testing the services

Here is an example of how to print the content of a phonebook named sales to stdout in CSV format

```
curl --insecure --basic -u superadmin [https://]{host}/sipxconfig/rest/phonebook/sales
```

To test the call placing service with curl use:

```
curl --insecure -X PUT [https://]{user}:{password}@{host}/sipxconfig/rest/call/{number}
```

HTTP PUT to service URL will cause sipXconfig to place call to {number}.
The calls is placed using authorized user credentials. It works in the same way as click-to-call available on the user portal. User's phone rings first and, once someone answers, it places the call to {number}.

To test the call forwarding service with curl use:

```
curl --insecure -X PUT [https://]{user}:{password}@{host}/sipxconfig/rest/my/forward/ --data-binary @{file.xml}
```

HTTP PUT to service URL will cause sipXconfig to place a callforward to a number with specs in the xml file.
A xml file should look like:

```
<call-sequence>
<rings>
<ring>
<expiration>20</expiration>
<type>If no response</type>
<enabled>true</enabled>
<number>100</number>
</ring>
</rings>
<expiration>23</expiration>
</call-sequence>
```

HTTP GET to service URL will return the saved callforwarding scheme. In addition, a new field is added: <withVoicemail>
The value of this field is true if the user voicemail permission is set to true, false if voicemail permission is set to false

```
<call-sequence>
<rings>
<ring>
<expiration>20</expiration>
<type>If no response</type>
<enabled>true</enabled>
<number>100</number>
</ring>
</rings>
<expiration>23</expiration>
<withVoicemail>true</withVoicemail>
</call-sequence>
```

Admin services

Accessible for users with admin privileges:

URI	Methods	Formats
/phonebook	GET	Returns a list with all the phonebooks. XML: {{<?xml version="1.0" encoding="UTF-8"?> <phonebooks><phonebook name="phonebook1"/><phonebook name="phonebook2"/></phonebooks>}}

/phonebook/{name}	GET	Returns a list with {name} phonebook entries. CSV `"First name","Last name","Number"` XML
/phone	POST	Creates a phone. XML
/auto-attendant	GET	Retrieves the list of auto-attendants configured. XML, JSON
/auto-attendant/specialmode	GET PUT DELETE	GET retrieves the use the special auto attendant status(true/false). PUT will set it to true, DELETE will set it to false. XML, JSON
/auto-attendant/{attendant} /special	PUT DELETE	PUT - Use the attendant as special attendant; DELETE - TBD
/activecdrs/{user}	GET	GET - Retrieve active calls for given user, sample output <cdrs> <cdr> <from>2012</from> <from-aor><sip:2012@test.com></from-aor> <to>32020</to> <to-aor><sip:32020@test.com></to-aor> <direction>INCOMING</direction> <recipient>32020</recipient> <internal>false</internal> <type>Unknown</type> <start-time>1361226016000</start-time> <duration>28926</duration> </cdr> </cdrs>

User services

Accessible for all users:

URI	Methods	Formats
/my/call/{to} /call/{to}	PUT	Initiates the call from the user to {to} address.PUT method requires non empty body which is ignored.Supported as GET for clients that do not handle PUT.
/my/voicemail/pin/{pin}	PUT	changes user voicemail PIN
/my/forward	GET PUT	retrieves (GET) or changes (PUT) user call forwardingXML,JSON
/my/feed/voicemail/{folder}	GET	voicemail folder presented as RSS feed
/my/phonebook	GET	JSON, XMLphonebook representation
/my/phonebook/entry/{entryId}	GET PUT DELETE	retrieves (GET), changes (PUT) and deletes (DELETE) entries in private phonebookXMLJSON
/my/contact-information	GET PUT	retrieve and change contact info for the userXML,JSON
/my/search/phonebook?query= {search-term}	GET	searching user phonebookXML
/my/mailbox/{user}/preferences /activegreeting /my/mailbox/{user}/preferences /activegreeting/{greeting}	GET PUT	retrieves and sets active greeting setting for a specific user GET: XML, plain text (one of none, standard, outofoffice, extendedabsence) PUT: plain text (one of none, standard, outofoffice, extendedabsence); an error 500 will be returned if the greeting is not one of the 4 strings
/my/conferences	GET	returns a list with all conferences for a specific user (enabled, name, description, extension) XML, JSON
/my/activecdrs	GET	returns a list with all active calls (ongoing) for a specific user in XML or JSON format
/my/logindetails	GET	returns username and im ID for a specific user in XML or JSON format. To be used when alias is provided in authentication details

Sample php Click to Call Code:

```
<?php
        $to="101";                //Number to dial
        $from="5001";        //userid in sipx
        $pass="1234";        //sipx pin (NOT SIP password)

        //replace sipx.gcgov.local with your sipx server
        $url = "http://sipx.gcgov.local:6667/callcontroller/".$from."/".$to."?isForwardingAllowed=true";
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_DIGEST);
        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_USERPWD, $from.":".$pass);
        $result = curl_exec($ch);
        curl_close($ch);
?>
```

Future Services

User ('my') services (those are services needed to implement functionality available through current user portal)

- account - pin, voicemail e-mail,
- voicemail - list, remove, delete, marked as saved
- call list
- speed dial
- personal phonebook (initially read only - configured by administrator, later also should allow adding/syncing from other phonebook sources)
- device/phones - monitoring registered devices
- conference - monitoring, muting, isolating, inviting, initiating
- personal attendant

Admin services:

- users adding/removing/listing
- phones adding/removing/configuring
- lines (users-phones) associations - adding/removing/configuring

New developer services:

- New or rewrite existing services implemented on TestPage.java
- Phonebook for end user (would require changing acegi security configuration)

If you are thinking about implementing an external application interacting with sipXecs and you need a new service ask on the sipx-dev list.

Adding new Services

See: Adding New Services in sipXconfig