Gigaset Redirect server

Redirection server

When the VoIP phone contacts the Gigaset server, in order to get all the necessary configuration data, the redirection server supplies the URL of the provisioning server which is responsible for providing the VoIP phone with the provider data (SIP account).

To enable auto-provisioning (i.e. the end-user does not need to select the provider manually) the provisioner must add the redirection information for the VoIP phones to the redirection database.

Access is secured via HTTP digest 12 digits username and MAC-ID.

Setting up redirection information using the web-interface

To add the redirection data to the redirection database, Gigaset provides a web user interface for provisioners.

You need a user account (user name and password) which has to be provided by Gigaset Communications.

1. Open the web user interface:

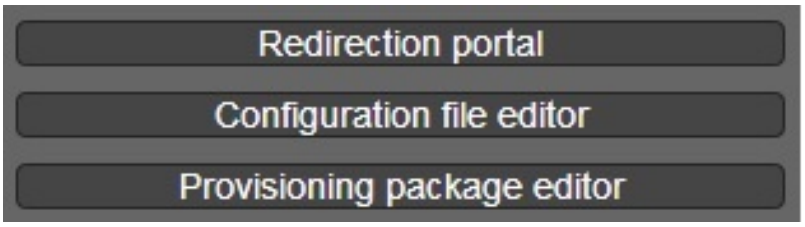
http://prov.gigaset.net or https://prov.gigaset.net/Editor/#loginPage

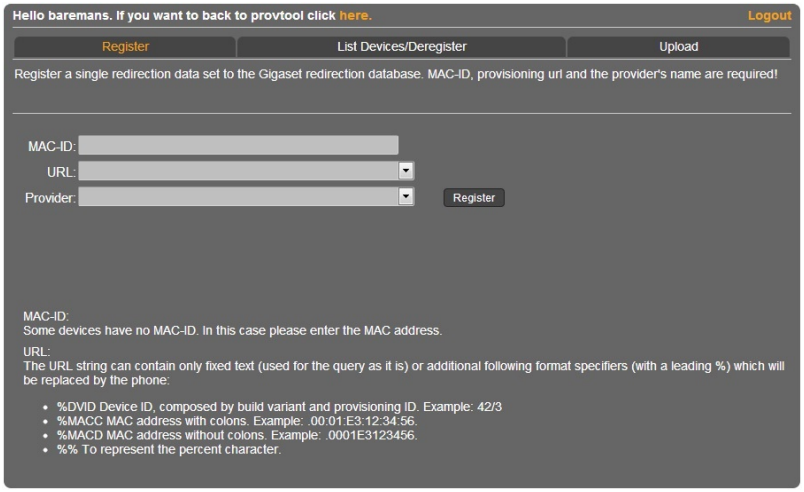


2. Login using the user name and password provided by Gigaset.

3. Select the option Redirect portal.
The main menu is opened. The following functions are available:

- Registration, control and deregistration of single devices
- Display of devices list
- Upload of prepared XML files





Registering VoIP phones

- To register a Gigaset VoIP phone, enter the MAC-ID (XXXXXXXXXXXX-XXXX) of the device, the URL of the provisioning server and the Provider for the device configuration.

  URL and **Provider** can be entered manually or selected from a list of known provisioner URLs and providers.

  Example: http://provider.com/xml/%MACD.xml

- Click on the Register button when to save the entry.

  The corresponding parameters are checked and – if approved – saved in the Gigaset redirection database. The provisioner is informed accordingly.

When using Maxwell 10, connection to the server and loading the XML configuration data takes longer then with other Gigaset devices. The reason is that other Android processes also need to be started after boot.

Connection to Redirect server 50 seconds after boot.

Activate new xml configuration 2 minutes after boot.

Maxwell B/2/3/4 have an option implemented that you can add the HTTP digest username and password in the URL.

Example: http://provider.com/xml/%MACD.xml

You want to add an HTTP digest username and password for your own server.

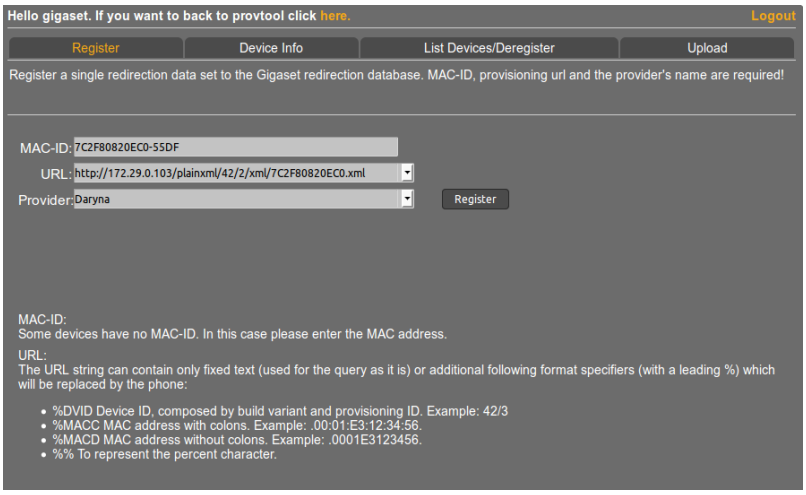
New example: http(s)://<username>:<password>@provider.com/xml/%MACD.xml

Example of usage (Based on XML-RPC technique)

Registration of the single device

To register a single device you may follow two different scenarios:

- Manually, by using "Register" tab, as shown.
- Using XML file of the next structure:

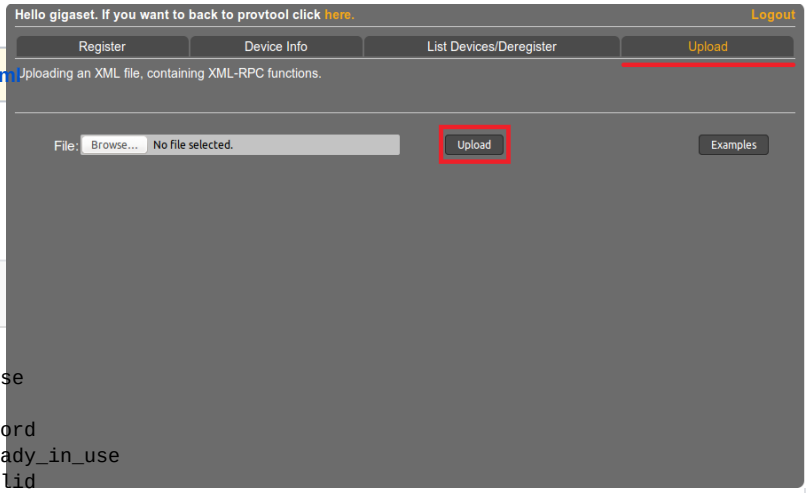


Registration of the single device

```
<?xml version='1.0'?>
<methodCall>
  <methodName>autoprov.registerDevice</methodName>
    <params>
     <param>
         <value>
           <string>7C2F80820EC0-55DF</string>
         </value>
         </param>
         <param>
         <value>
                <string>http://172.29.0.103/plainxml/42/2/xml/7C2F80820EC0.xml</string>
         </value>
         </param>
         <param>
         <value>
                <string>Daryna</string>
         </value>
         </param>
        </params>
</methodCall>
```

XML template can be find here:

Download XML file_ Registration of the single device.xml



You need to create valid XML – file in accordance with the provided example, then “Browse” it and “Upload”.
Next server's answers are possible:

Server's answers

```
Return value (1) (Boolean) 1 | 0
                            1 = true, 0 = false
Return value (2) (String)
                           if true: OK: password
                           if false: mac_already_in_use
                                     mac_invalid
                                     url_invalid
                                     name_invalid
```

After successful uploading your XML – file, you will see server response:

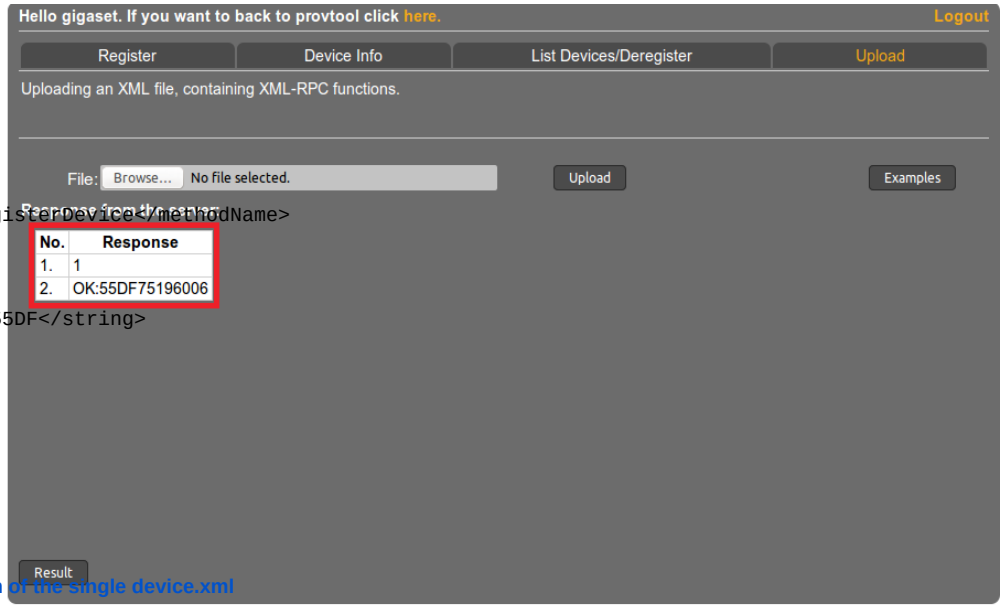
Deregistration of a single device

You can deregister device with a MAC - ID – address e.g. “7C2F80820EC0” manually, by pressing, “Deregister” button on the left corner of the window:

Usage of XML file instead is also possible:

Deregistration of the single device

```
<?xml version='1.0'?>
<methodCall>
  <methodName>autoprov.deregisterDevice</methodName>
   <params>
    <param>
    <value>
      <string>7C2F80820EC0-55DF</string>
    </value>
    </param>
   </params>
</methodCall>
```



XML template for the deregistration can be downloaded here:

Download XML file_ Deregistration of the single device.xml

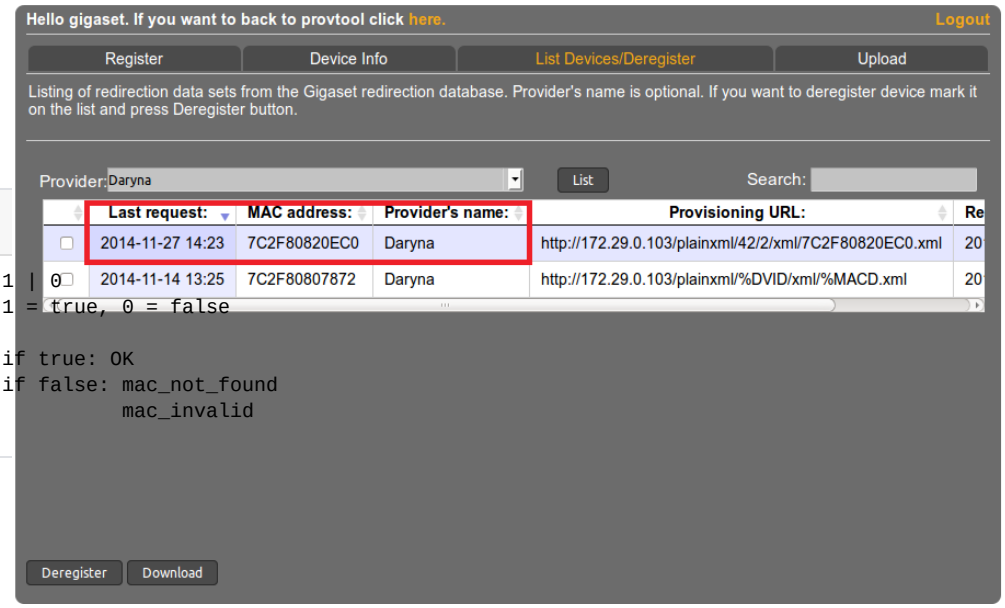
After XML file was constructed, it is necessary to upload it, in the same way as it was described previously. Different
server's answers are possible:

Server's answers

```
Return value (1) (Boolean) 1 | 0
1 = true, 0 = false

Return value (2) (String)

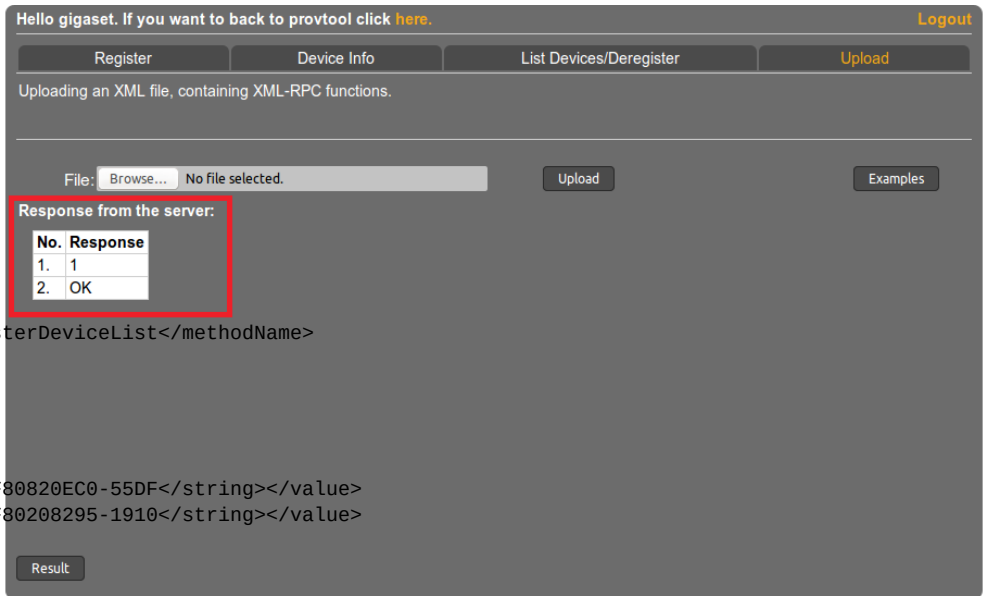
if true: OK
if false: mac_not_found
          mac_invalid
```



In our case XML file within MAC-ID were valid and we got next response:

Register device list
Let's assume that at once, you may need to register not a single device, but a list containing several of them,

XML structure for such a case will look like this:



Deregistration of the single device

```
<?xml version='1.0'?>
<methodCall>
  <methodName>autoprov.registerDeviceList</methodName>
   <params>
    <param>
    <value>
      <array>
      <data>
         <value><string>7C2F80820EC0-55DF</string></value>
         <value><string>7C2F80208295-1910</string></value>
     </data>
     </array>
    </value>
    </param>
    <param>
        <value><string>http://172.29.0.103/plainxml/%DVID/xml/%MACD.xml</string></value>
   </param>
    <param>
       <value><string>Daryna</string></value>
   </param>
  </params>
</methodCall>
```

XML template can be downloaded here:

Download XMl file_ Registration list of devices.xml

As previously, several server's answers are possible:

Server's answers

```
Return value (1) (Boolean) 1 | 0
                           1 = true, 0 = false
Return value (2-n) (String)
                           if true: OK: Password
                           if false: mac_invalid
                                     url_invalid
                                     name_invalid
                                     mac_already_in_use
                                     mac_not_exist
```

The next picture shows, 'True' server's answer:

An example of real server response after usage of the XML – file with incorrect data (inappropriate MAC-ID was used):

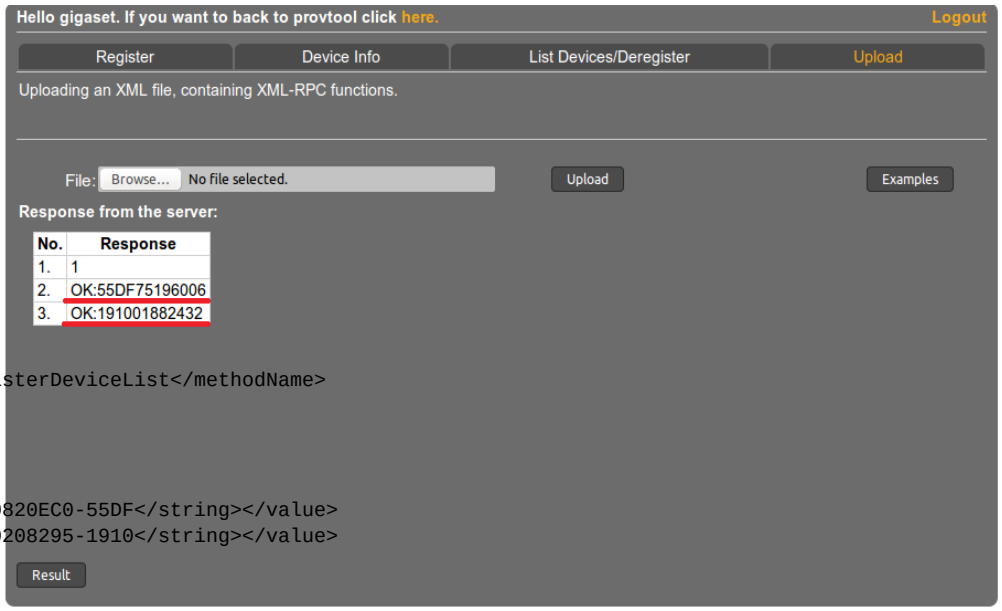
Deregistration of the list of devices

Let's assume that at once, you may need to deregister not a single device, but a list containing several of them.

XML structure for such a case will look like this:

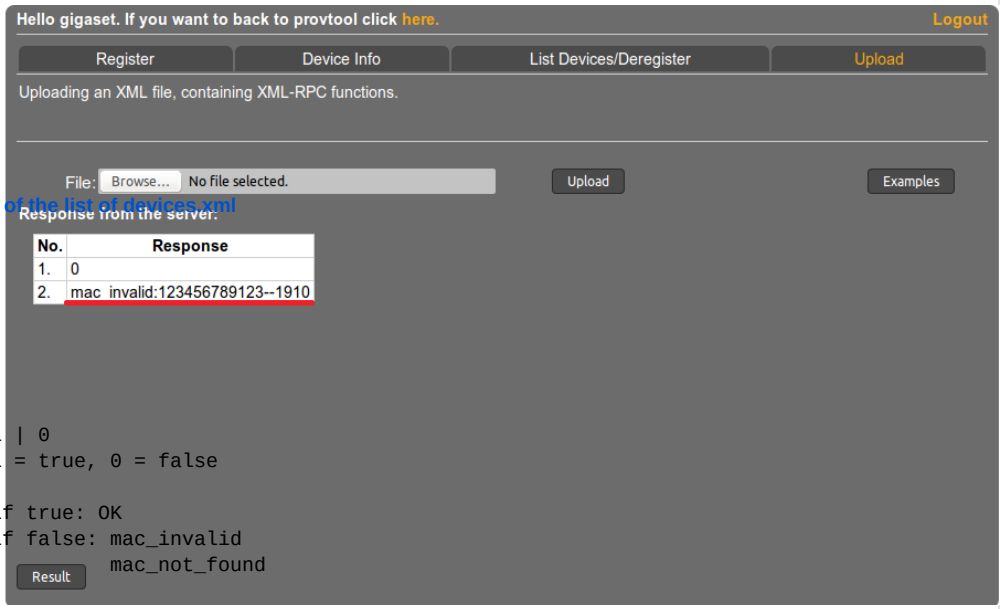
Deregistration of the list of devices

```
<?xml version='1.0'?>
<methodCall>
 <methodName>autoprov.deregisterDeviceList</methodName>
   <params>
   <param>
     <value><array>
       <data>
       <value><string>7C2F80820EC0-55DF</string></value>
       <value><string>7C2F80208295-1910</string></value>
       </data>
     </array></value>
   </param>
   </params>
</methodCall>
```



XML template can be downloaded here:

Download XML file- Deregistration of the list of devices.xml



Next server responds are possible:

Server's answers

```
Return value (1) (Boolean) 1 | 0
                           1 = true, 0 = false
Return value (2-n) (String)
                           if true: OK
                           if false: mac_invalid
                                     mac_not_found
```

An example of real server respond on the correct XML – file:

An example of real server responds, while only one of the provided MAC – ID was invalid:

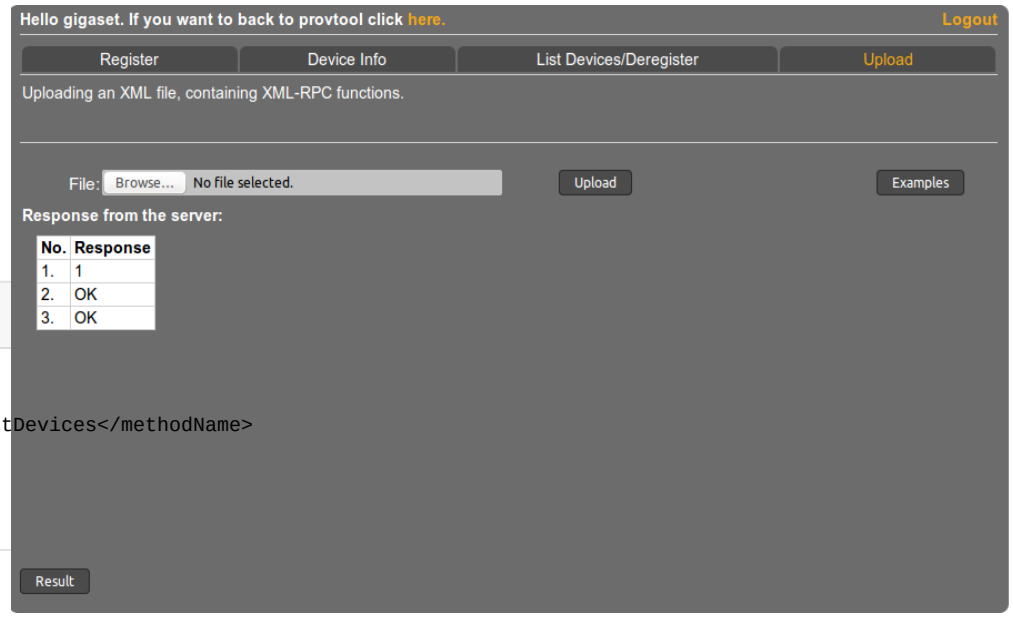
List devices
The purpose of this request is to get:

- MAC – address(es) of the registered device(s);
- Provider name(s);
- URL(s);
- Date(s) of the registration;

General structure of the XML – file for such a case looks like this:

Devices listeng

```
<?xml version='1.0'?>
 <methodCall>
   <methodName>autoprov.listDevices</methodName>
     <params>
     </params>
</methodCall>
```



XML template can be downloaded here:

Download XML file- List devices.xml

As in the previous cases, several server's responses are possible:

Server's answers

```
Return value (0-n)
             (Object[])
             [MAC,NAME,URL,DATE]
```

On the
picture below, you can see the result of the real server respond after valid XML – file uploading:

Here, by pressing the “Result” button marked by red square, you can get XML – file containing the same information as presented on the picture, in other words, XML – file, containing – MAC – address, provider name, URL and date of registration.

5.1 List devices name

This is an a sub-option of the method presented above. Basically, you will get the same information (after uploading your XML – file), but for a specific provider.
General structure of the XML – file:

Devices listeng

```
<?xml version='1.0'?>
<methodCall>
  <methodName>autoprov.listDevices</methodName>
    <params>
      <param>
      <value><string>Daryna</string></value>
      </param>
    </params>
</methodCall>
```

- *you can specify as many providers as you wish*

General structure of the XML – file:

Download XMl file- List devices name.xml

Server response description:

Server's answers

```
Return value (0-n)
             (Object[])
             [MAC,NAME,URL,DATE]
```

The result of usage valid XML -file
after successful uploading (only one provider was specified):

Check device(s)
This option allow its requester to check either the device is registered or not. Along with information that device is registered, requester will get a provider name, URL and date of registration.
General structure of an XML file will look like this:

Devices listeng

```
<?xml version='1.0'?>
<methodCall>
<methodName>autoprov.checkDevice</methodName>
<params>
<param>
<value><string>7C2F80820EC0-55DF</string></value>
</param>
</params>
</methodCall>
```

XML template can be downloaded here:

Download XML file- Check devices.xml

Several server's responses are possible:

Server's answers

```
Return value (1) (Boolean) 1 | 0
                           1 = true, 0 = false
Return value (2) (String)
                         if true: [MAC]
                         if false: mac_not_found
                                   mac_invalid
Return value (3)
                         if true: [NAME]
Return value (4)
                         if true: [DATE]
Return value (5)
                         if true: [URL]
```

Below, you
can see the real server respond based on the valid MAC – ID:

The real
server respond based on the invalid MAC – ID:
Usage of XML-RPC script

Registration of the single device using ap-client.py script

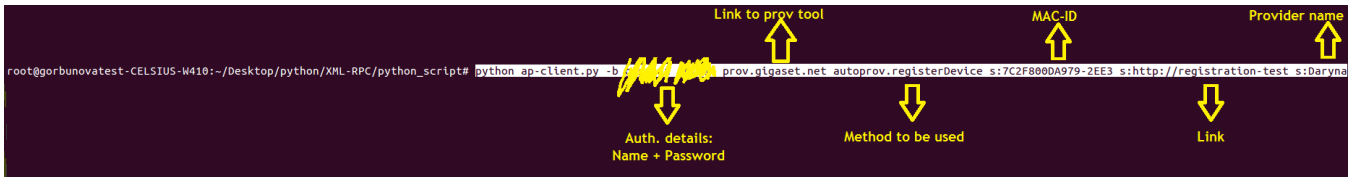
First thing which has to be done (for this steps and all the steps below):

- Make sure Python is installed on your OS. If not, please do install.
- Under console change your working to the directory in which ap-client.py file is saved.
- Run necessary command.

After all the described steps above were done, please run 1st command for the device registration:

Device registration

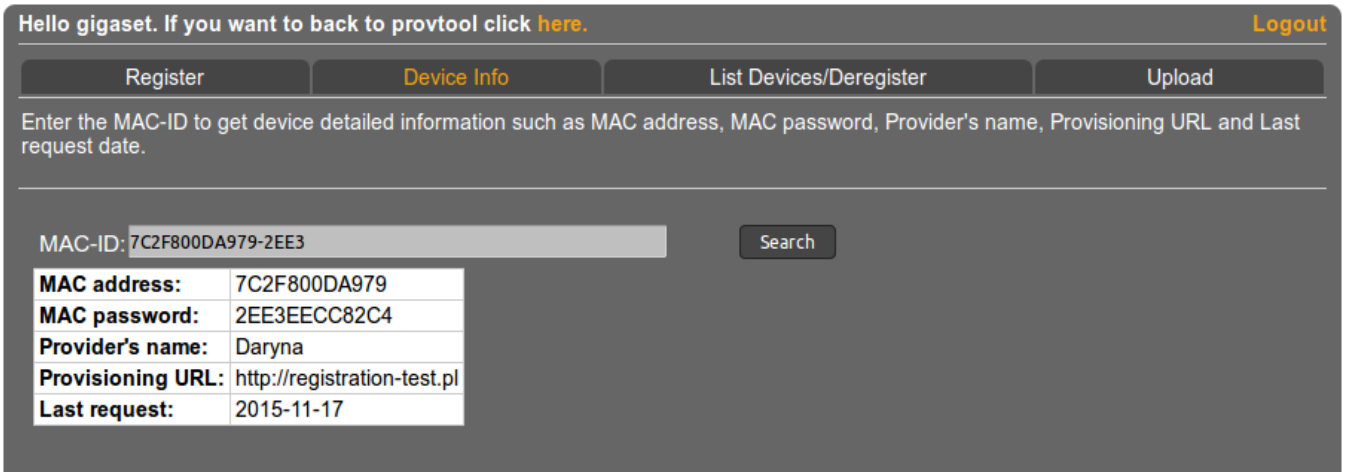
```
python ap-client.py Auth-Name Auth-Pass prov.gigaset.net autoprov.registerDevice s:MAC-ID s:URL s:Provider-
Name
```



In case of the successful registration, server will answer:



And in prov.gigaset.net your device will be recognized as the registered one:

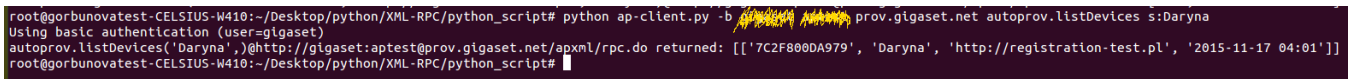


Check - status

In order to check status of the device, you have to run next command:

Check-status
`python ap-client.py Auth-Name Auth-Pass prov.gigaset.net autoprov.listDevices s:Provider-Name`

On the picture below, command in use within server response can be seen:

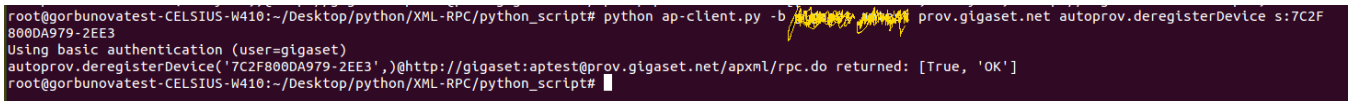


Device deregistration

In order to deregister device, you have to run next command:

```
python ap-client.py Auth-Name Auth-Pass prov.gigaset.net autoprov.deregisterDevice s:MAC-ID
```

On the picture below, command in use within server response can be seen:



Deregistration of the list of devices

In order to deregister list of the devices, you have to run next command (specify here all MAC-ID which you want to deregister):

```
python ap-client.py Auth-Name Auth-Pass prov.gigaset.net autoprov.deregisterDeviceList sl:MAC-ID1,MAC-ID2,MAC-IDN
```

On the picture below, command in use within server response can be seen:

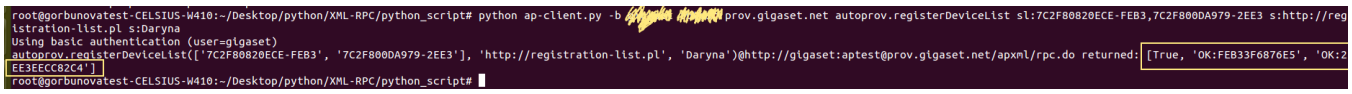


Registration of the list of devices

In order to register list of the devices, you have to run next command (specify here all MAC-ID which you want to register):

```
python ap-client.py Auth-Name Auth-Pass prov.gigaset.net autoprov.registerDeviceList sl:MAC-ID1,MAC-ID2,MAC-IDN s:URL, s:Provider-Name
```

On the picture below, command in use within server response can be seen:



DX800 enabling Redirect Server

Dependent on the DX800 variant, it can be that the redirect server option is disabled. This can only be enabled via special configuration, for this you need a patch. The patch is based on the variant and contains the following settings to enable redirect server.

DX800 enable redirect

BS_IP_Data1.ucB_HAS_PROVISIONING_CODE=1
BS_IP_Data1.ucB_SHOW_PROVISIONING_CODE_IN_HS=1
BS_IP_Data1.ucB_AUTOPROVISIONING=0
BS_IP_Data1.ucI_AUTOPROVISIONING_STYLE=0
BS_IP_Data1.ucB_AUTO_UPDATE_PROFILE=1
BS_IP_Data3.ucI_ONESHOT_PROVISIONING_MODE_1=1
BS_IP_Data1.aucS_AUTOPROVISIONING_CODE=0x34,0x34,0x34

Via the restore option, you can activate the patch below. If you variant is not listed you need to request this from Gigaset.

- Patch for DX800 Dutch variant DX800_Provisioning_Norm_patch.cfg
- Patch for DX800 GER variant DX800_Provisioning_Norm_patch_de.cfg