Sympa SOAP server

Introduction

SOAP [http://www.w3.org/2002/ws/] is a protocol (generally over HTTP) that can be used to provide **web services**. Sympa SOAP server allows to access a Sympa service from within another program, written in any programming language and on any computer. SOAP encapsulates procedure calls, input parameters and resulting data in an XML data structure. The Sympa SOAP server's API is published in a **WSDL** document, retrieved through Sympa's web interface.

The SOAP server provides a limited set of high level functions, see supported_functions. Other functions might be implemented in the future. One of the important implementation constraints is to provide services for proxy applications with a correct authorization evaluation process where authentication may differ from classic web methods. The following cases can be used to access the service:

- The client sends credentials and then requests a service providing a cookie with an id `sympa-user`.
- The client authenticates the end user providing the `sympa-user` HTTP cookie. This can be used in order to share an authenticated session between Sympa and other applications running on the same server as *WWSympa*. The SOAP method used is `getUserEmailByCookieRequest`.
- The client provides a user email and password and requests a service in a single SOAP access using the `authenticateAndRun` SOAP service.
- The client is trusted by Sympa as a proxy application and is authorized to set some variables that will be used by Sympa during the authorization scenario evaluation. Trusted applications have their own password, and the variables they can set are listed in a configuration file named `trusted_applications.conf`. See Trust remote applications.

In any case, scenario authorization is used with the same rules as a mail interface or a normal web interface.

The SOAP server uses the SOAP::Lite [http://www.soaplite.com/] Perl library. The server is running as a daemon (thanks to FastCGI), receiving the client SOAP requests via a web server (Apache for example).

Supported functions

Note that all functions accessible through the SOAP interface apply the appropriate access control rules, given the user's privileges.

The following functions are currently available through the Sympa SOAP server :

- login : user email and passwords are checked against Sympa user DB, or another backend.
- casLogin : this function will verify CAS proxy tickets against the CAS server
- authenticateAndRun : useful for SOAP clients that can't set an HTTP cookie ; they can provide both the Sympa session cookie and the requested command in a single call
- authenticateRemoteAppAndRun : equivalent of the previous command used in a trusted context (see trust_remote_applications)
- lists : provides a list of available lists (authorization scenarios are applied)
- complexLists : same as the previous feature, but provides a complex structure for each list
- info : provides description informations about a given list
- which : gets the list of subscription of a given user
- complexWhich : same as previous command, but provides a complex structure for each list
- amI : tells if a given user is member of a given list
- review : lists the members of a given list
- subscribe : subscribes the current user to a given list
- signoff : current user is removed from a given list
- add : used to add a given user to a given list (admin feature)
- del : removes a given user from a given list (admin feature)
- createList : creates a new mailing list (requires appropriate privileges)
- closeList : closes a given mailing list (admin feature)

Note that when a list parameter is required for a function, you can either provide the list name or the list address. However the domain part of the address will be ignored.

Check the_wsdl_service_description for detailed API informations.

Web server setup

Starting Sympa 5.4, the sympa_soap_server is wrapped in small C script, sympa_soap_server-wrapper.fcgi, in order to avoid to use the -unsecure and no longer maintained - setuid perl mode.

You **need to install FastCGI** for the SOAP server to work properly, because it will run as a daemon.

Until version 5.3

Here is a sample piece of your Apache `httpd.conf` with a SOAP server configured:

```
FastCgiServer /home/sympa/bin/sympa_soap_server.fcgi -processes 1
ScriptAlias /sympasoap /home/sympa/bin/sympa_soap_server.fcgi

<Location /sympasoap>
     SetHandler fastcgi-script
</Location>
```

Version 5.4 and higher

Here is a sample piece of your Apache `httpd.conf` with a SOAP server configured and using the C wrapper:

```
     FastCgiServer /home/sympa/bin/sympa_soap_server-wrapper.fcgi -processes 1
     ScriptAlias /sympasoap /home/sympa/bin/sympa_soap_server-wrapper.fcgi

     <Location /sympasoap>
          SetHandler fastcgi-script
     </Location>
```

Sympa setup

The only mandatory parameter you need to set in the `sympa.conf`/`robot.conf` files is the `soap_url`, that defines the URL of the SOAP service corresponding to the ScriptAlias you have previously set up in the Apache configuration.

This parameter is used to publish the SOAP service URL in the WSDL file (defining the API), but also for the SOAP server to deduce what Virtual Host is concerned by the current SOAP request (a single SOAP server will serve all Sympa virtual hosts).

Trust remote applications

The SOAP service `authenticateRemoteAppAndRun` is used in order to allow some remote applications such as a web portal to request the Sympa service as a proxy for the end user. In such cases, Sympa will not authenticate the end user itself, but instead it will trust a particular application to act as a proxy.

This configuration file `trusted_applications.conf` can be created in the robot `etc/` subdirectory or in the `/home/sympa/etc` directory depending on the scope you want for it (the source package include a sample of file `trusted_applications.conf` in the `soap` directory). This file is made of paragraphs separated by empty lines and stating with keyword `trusted_application`. A sample `trusted_applications.conf` file is provided with Sympa sources. Each paragraph defines a remote trusted application with keyword/value pairs:

- `name`: the name of the application. Used with password for authentication; the `remote_application_name` variable is set for use in authorization scenarios;
- `md5password`: the MD5 digest of the application password. You can compute the digest as follows: `sympa.pl -md5_digest=<the password>`.
- `proxy_for_variables`: a comma separated list of variables that can be set by the remote application and that will be used by the Sympa SOAP server when evaluating an authorization scenario. If you list `USER_EMAIL` in this parameter, then the remote application can act as a user. Any other variable such as `remote_host` can be listed.

You can test your SOAP service using the `sympa_soap_client.pl` sample script as follows:

```
  /home/sympa/bin/sympa_soap_client.pl --soap_url=http://my.server/sympasoap --service=createList --trusted_application=myTestApp --trusted_application_password=myTest

  /home/sympa/bin/sympa_soap_client.pl --soap_url=http://myserver/sympasoap --service=add --trusted_application=myTestApp --trusted_application_password=myTestAppPwd
```

Below is a sample Perl code that does a SOAP procedure call (for a SUBSCRIBE sympa command) using the trusted_application feature :

```
my $soap = new SOAP::Lite();
$soap->uri('urn:sympasoap');
$soap->proxy('http://myserver/sympasoap');

my $response = $soap->authenticateRemoteAppAndRun('myTestApp', 'myTestAppPwd', 'USER_EMAIL=userProxy@my.server', 'subscribe', ['myList@dom']);
```

S. Santoro [mailto:dereckson@espace-win.org] wrote its own PHP Trusted Application library for Sympa.

The WSDL service description

Here is what the WSDL file looks like before it is parsed by *WWSympa*:

```
  <?xml version=''1.0''?>
  <definitions name=''Sympa''
      xmlns:xsd=''http://www.w3.org/2001/XMLSchema''
      xmlns:soap=''http://schemas.xmlsoap.org/wsdl/soap/''
      targetNamespace="[% conf.wwsympa_url %]/wsdl"
      xmlns:tns="[% conf.wwsympa_url %]/wsdl"
      xmlns=''http://schemas.xmlsoap.org/wsdl/''
      xmlns:xsdl="[% conf.soap_url %]/wsdl">

  <!-- types part -->

  <types>
  <schema targetNamespace="[% conf.wwsympa_url %]/wsdl"
      xmlns:SOAP-ENC=''http://schemas.xmlsoap.org/soap/encoding/''
      xmlns:wsdl=''http://schemas.xmlsoap.org/wsdl/''
      xmlns=''http://www.w3.org/2001/XMLSchema''>

      <complexType name=''ArrayOfLists''>
          <complexContent>
              <restriction base=''SOAP-ENC:Array''>
                  <attribute ref=''SOAP-ENC:arrayType'' wsdl:arrayType=''tns:listType[]''/>
              </restriction>
          </complexContent>
      </complexType>

      <complexType name=''ArrayOfString''>
          <complexContent>
              <restriction base=''SOAP-ENC:Array''>
                  <attribute ref=''SOAP-ENC:arrayType'' wsdl:arrayType=''string[]''/>
              </restriction>
          </complexContent>
      </complexType>

      <complexType name=''listType''>
             <all>
          <element name=''listAddress'' minOccurs=''1'' type=''string''/>
          <element name=''homepage'' minOccurs=''0'' type=''string''/>
          <element name=''isSubscriber'' minOccurs=''0'' type=''boolean''/>
          <element name=''isOwner'' minOccurs=''0'' type=''boolean''/>
          <element name=''isEditor'' minOccurs=''0'' type=''boolean''/>
          <element name=''subject'' minOccurs=''0'' type=''string''/>
```

```
           </all>
    </complexType>
</schema>
</types>

<!-- message part -->

<message name=''infoRequest''>
        <part name=''listName'' type=''xsd:string''/>
</message>

<message name=''infoResponse''>
    <part name=''return'' type=''tns:listType''/>
</message>

<message name=''complexWhichRequest''>
</message>

<message name=''complexWhichResponse''>
    <part name=''return'' type=''tns:ArrayOfLists''/>
</message>

<message name=''whichRequest''>
</message>

<message name=''whichResponse''>
    <part name=''return'' type=''tns:ArrayOfString''/>
</message>

<message name=''amIRequest''>
    <part name=''list'' type=''xsd:string''/>
    <part name=''function'' type=''xsd:string''/>
    <part name=''user'' type=''xsd:string''/>
</message>

<message name=''amIResponse''>
    <part name=''return'' type=''xsd:boolean''/>
</message>

<message name=''reviewRequest''>
    <part name=''list'' type=''xsd:string''/>
</message>

<message name=''reviewResponse''>
    <part name=''return'' type=''tns:ArrayOfString''/>
</message>

<message name=''signoffRequest''>
    <part name=''list'' type=''xsd:string''/>
    <part name=''email'' type=''xsd:string'' xsd:minOccurs=''0''/>
</message>

<message name=''signoffResponse''>
    <part name=''return'' type=''xsd:boolean''/>
</message>

<message name=''subscribeRequest''>
    <part name=''list'' type=''xsd:string''/>
    <part name=''gecos'' type=''xsd:string'' xsd:minOccurs=''0''/>
</message>

<message name=''addRequest''>
    <part name=''list'' type=''xsd:string''/>
    <part name=''email'' type=''xsd:string''/>
    <part name=''gecos'' type=''xsd:string''  xsd:minOccurs=''0''/>
    <part name=''quiet'' type=''xsd:boolean''  xsd:minOccurs=''0''/>
</message>

<message name=''addResponse''>
    <part name=''return'' type=''xsd:boolean''/>
</message>

<message name=''delRequest''>
    <part name=''list'' type=''xsd:string''/>
    <part name=''email'' type=''xsd:string''/>
    <part name=''quiet'' type=''xsd:boolean''  xsd:minOccurs=''0''/>
</message>

<message name=''delResponse''>
    <part name=''return'' type=''xsd:boolean''/>
</message>

<message name=''createListRequest''>
    <part name=''list'' type=''xsd:string''/>
    <part name=''subject'' type=''xsd:string''/>
    <part name=''template'' type=''xsd:string''/>
    <part name=''description'' type=''xsd:string''/>
    <part name=''topics'' type=''xsd:string''/>
</message>

<message name=''createListResponse''>
    <part name=''return'' type=''xsd:boolean''/>
</message>

<message name=''closeListRequest''>
    <part name=''list'' type=''xsd:string''/>
</message>

<message name=''closeListResponse''>
    <part name=''return'' type=''xsd:boolean''/>
</message>

<message name=''subscribeResponse''>
    <part name=''return'' type=''xsd:boolean''/>
</message>

<message name=''loginRequest''>
    <part name=''email'' type=''xsd:string''/>
    <part name=''password'' type=''xsd:string''/>
</message>
```

```
<message name=''loginResponse''>
    <part name=''return'' type=''xsd:string''/>
</message>

<message name=''getUserEmailByCookieRequest''>
    <part name=''cookie'' type=''xsd:string''/>
</message>

<message name=''getUserEmailByCookieResponse''>
    <part name=''return'' type=''xsd:string''/>
</message>

<message name=''authenticateAndRunRequest''>
    <part name=''email'' type=''xsd:string''/>
    <part name=''cookie'' type=''xsd:string''/>
    <part name=''service'' type=''xsd:string''/>
    <part name=''parameters'' type=''tns:ArrayOfString'' xsd:minOccurs=''0''/>
</message>

<message name=''authenticateAndRunResponse''>
    <part name=''return'' type=''tns:ArrayOfString'' xsd:minOccurs=''0''/>
</message>

<message name=''authenticateRemoteAppAndRunRequest''>
    <part name=''appname'' type=''xsd:string''/>
    <part name=''apppassword'' type=''xsd:string''/>
    <part name=''vars'' type=''xsd:string''/>
    <part name=''service'' type=''xsd:string''/>
    <part name=''parameters'' type=''tns:ArrayOfString'' xsd:minOccurs=''0''/>
</message>

<message name=''authenticateRemoteAppAndRunResponse''>
    <part name=''return'' type=''tns:ArrayOfString'' xsd:minOccurs=''0''/>
</message>

<message name=''casLoginRequest''>
    <part name=''proxyTicket'' type=''xsd:string''/>
</message>

<message name=''casLoginResponse''>
    <part name=''return'' type=''xsd:string''/>
</message>

<message name=''listsRequest''>
    <part name=''topic'' type=''xsd:string'' xsd:minOccurs=''0''/>
    <part name=''subtopic'' type=''xsd:string'' xsd:minOccurs=''0''/>
</message>

<message name=''listsResponse''>
    <part name=''listInfo'' type=''xsd:string''/>
</message>

<message name=''complexListsRequest''>
</message>

<message name=''complexListsResponse''>
    <part name=''return'' type=''tns:ArrayOfLists''/>
</message>

<message name=''checkCookieRequest''>
</message>

<message name=''checkCookieResponse''>
    <part name=''email'' type=''xsd:string''/>
</message>

<!-- portType part -->

<portType name=''SympaPort''>
    <operation name=''info''>
        <input message=''tns:infoRequest'' />
        <output message=''tns:infoResponse'' />
    </operation>
    <operation name=''complexWhich''>
        <input message=''tns:complexWhichRequest'' />
        <output message=''tns:complexWhichResponse'' />
    </operation>
    <operation name=''which''>
        <input message=''tns:whichRequest'' />
        <output message=''tns:whichResponse'' />
    </operation>
    <operation name=''amI''>
        <input message=''tns:amIRequest'' />
        <output message=''tns:amIResponse'' />
    </operation>
    <operation name=''add''>
        <input message=''tns:addRequest'' />
        <output message=''tns:addResponse'' />
    </operation>
    <operation name=''del''>
        <input message=''tns:delRequest'' />
        <output message=''tns:delResponse'' />
    </operation>
    <operation name=''createList''>
        <input message=''tns:createListRequest'' />
        <output message=''tns:createListResponse'' />
    </operation>
    <operation name=''closeList''>
        <input message=''tns:closeListRequest'' />
        <output message=''tns:closeListResponse'' />
    </operation>
    <operation name=''review''>
        <input message=''tns:reviewRequest'' />
        <output message=''tns:reviewResponse'' />
    </operation>
    <operation name=''subscribe''>
        <input message=''tns:subscribeRequest'' />
        <output message=''tns:subscribeResponse'' />
    </operation>
    <operation name=''signoff''>
        <input message=''tns:signoffRequest'' />
        <output message=''tns:signoffResponse'' />
    </operation>
    <operation name=''login''>
        <input message=''tns:loginRequest'' />
        <output message=''tns:loginResponse'' />
    </operation>
    <operation name=''casLogin''>
        <input message=''tns:casLoginRequest'' />
        <output message=''tns:casLoginResponse'' />
    </operation>
    <operation name=''getUserEmailByCookie''>
        <input message=''tns:getUserEmailByCookieRequest'' />
        <output message=''tns:getUserEmailByCookieResponse'' />
    </operation>
    <operation name=''authenticateAndRun''>
        <input message=''tns:authenticateAndRunRequest'' />
        <output message=''tns:authenticateAndRunResponse'' />
    </operation>
    <operation name=''authenticateRemoteAppAndRun''>
        <input message=''tns:authenticateRemoteAppAndRunRequest'' />
        <output message=''tns:authenticateRemoteAppAndRunResponse'' />
    </operation>
    <operation name=''lists''>
        <input message=''tns:listsRequest'' />
        <output message=''tns:listsResponse'' />
    </operation>
    <operation name=''complexLists''>
        <input message=''tns:complexListsRequest'' />
        <output message=''tns:complexListsResponse'' />
    </operation>
    <operation name=''checkCookie''>
        <input message=''tns:checkCookieRequest'' />
        <output message=''tns:checkCookieResponse'' />
    </operation>
</portType>

<!-- Binding part -->

<binding name=''SOAP'' type=''tns:SympaPort''>
<soap:binding style=''rpc'' transport=''http://schemas.xmlsoap.org/soap/http''/>
    <operation name=''info''>
        <soap:operation soapAction=''urn:sympasoap#info''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''complexWhich''>
        <soap:operation soapAction=''urn:sympasoap#complexWhich''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''which''>
        <soap:operation soapAction=''urn:sympasoap#which''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''amI''>
        <soap:operation soapAction=''urn:sympasoap#amI''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''createList''>
        <soap:operation soapAction=''urn:sympasoap#createList''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''review''>
        <soap:operation soapAction=''urn:sympasoap#review''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''subscribe''>
        <soap:operation soapAction=''urn:sympasoap#subscribe''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''signoff''>
        <soap:operation soapAction=''urn:sympasoap#signoff''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''login''>
        <soap:operation soapAction=''urn:sympasoap#login''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''casLogin''>
        <soap:operation soapAction=''urn:sympasoap#casLogin''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''getUserEmailByCookie''>
        <soap:operation soapAction=''urn:sympasoap#getUserEmailByCookie''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''authenticateAndRun''>
        <soap:operation soapAction=''urn:sympasoap#authenticateAndRun''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''authenticateRemoteAppAndRun''>
        <soap:operation soapAction=''urn:sympasoap#authenticateRemoteAppAndRun''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </output>
    </operation>
    <operation name=''lists''>
        <soap:operation soapAction=''urn:sympasoap#lists''/>
            <input>
                <soap:body use=''encoded''
                    namespace=''urn:sympasoap''
                    encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
            </input>
            <output>
                <soap:body use=''encoded''
```

```
                        namespace=''urn:sympasoap''
                        encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
                </output>
        </operation>
        <operation name=''complexLists''>
            <soap:operation soapAction=''urn:sympasoap#complexLists''/>
                <input>
                    <soap:body use=''encoded''
                        namespace=''urn:sympasoap''
                        encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
                </input>
                <output>
                    <soap:body use=''encoded''
                        namespace=''urn:sympasoap''
                        encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
                </output>
        </operation>
        <operation name=''checkCookie''>
            <soap:operation soapAction=''urn:sympasoap#checkCookie''/>
                <input>
                    <soap:body use=''encoded''
                        namespace=''urn:sympasoap''
                        encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
                </input>
                <output>
                    <soap:body use=''encoded''
                        namespace=''urn:sympasoap''
                        encodingStyle=''http://schemas.xmlsoap.org/soap/encoding/''/>
                </output>
        </operation>
    </binding>

    <!-- service part -->

    <service name=''SympaSOAP''>
        <port name=''SympaPort'' binding=''tns:SOAP''>
            <soap:address location="[% conf.soap_url %]"/>
        </port>
    </service>

    </definitions>
```

Client-side programming

Sympa is distributed with 2 sample clients written in Perl and in PHP. The Sympa SOAP server has also been successfully tested with a UPortal Channel as a Java client (using Axis). The sample PHP SOAP client has been installed on our demo server: http://demo.sympa.org/sampleClient.php [http://demo.sympa.org/sampleClient.php].

Depending on your programming language and the SOAP library you are using, you will either directly contact the SOAP service (as with the Perl SOAP::Lite library), or first load the WSDL description of the service (as with PHP nusoap or Java Axis). Axis is able to create a stub from the WSDL document.

The WSDL document describing the service should be fetched through *WWSympa*'s dedicated URL: http://your.server/sympa/wsdl.

Note: the `login()` function maintains a login session using HTTP cookies. If you are not able to maintain this session by analyzing and sending appropriate cookies under SOAP, then you should use the `authenticateAndRun()` function that does not require cookies to authenticate.

Writing a Java client with Axis

First, download jakarta-axis (http://ws.apache.org/axis [http://ws.apache.org/axis]).

You must add the libraries provided with jakarta axis (v >1.1) to you CLASSPATH. These libraries are:

- axis.jar;
- saaj.jar;
- commons-discovery.jar;
- commons-logging.jar;
- xercesImpl.jar;
- jaxrpc.jar;
- xml-apis.jar;
- jaas.jar;
- wsdl4j.jar;
- soap.jar.

Next, you have to generate client Java class files from the sympa WSDL URL. Use the following command:

```
java org.apache.axis.wsdl.WSDL2Java -av WSDL_URL
```

For example:

```
java org.apache.axis.wsdl.WSDL2Java -av  http://demo.sympa.org/sympa/wsdl
```

Exemple of screen output during generation of Java files:

```
  Parsing XML file:  http://demo.sympa.org/sympa/wsdl
  Generating org/sympa/demo/sympa/msdl/ListType.java
  Generating org/sympa/demo/sympa/msdl/SympaPort.java
  Generating org/sympa/demo/sympa/msdl/SOAPStub.java
  Generating org/sympa/demo/sympa/msdl/SympaSOAP.java
  Generating org/sympa/demo/sympa/msdl/SympaSOAPLocator.java
```

If you need more information or more generated classes (to have the server-side classes or junit testcase classes for example), you can get a list of switches:

```
java org.apache.axis.wsdl.WSDL2Java -h
```

The reference page is: http://ws.apache.org/axis/java/reference.html [http://ws.apache.org/axis/java/reference.html].

Take care of Test classes generated by axis, there are not useable as are. You have to stay connected between each test. To use junit testcases, before each SOAP operation tested, you must call the authenticated connexion to Sympa instance.

Here is a simple Java code that invokes the generated stub to perform a `casLogin()` and a `which()` on the remote Sympa SOAP server:

```
  SympaSOAP loc = new SympaSOAPLocator();
  ((SympaSOAPLocator)loc).setMaintainSession(true);
  SympaPort tmp = loc.getSympaPort();
  String _value = tmp.casLogin(_ticket);
  String _cookie = tmp.checkCookie();
  String[] _abonnements = tmp.which();
```

The test command line SOAP client

Sympa distribution includes a simple command line application that allows you to test SOAP request towards your Sympa SOAP server. This script is named sympa_soap_client.pl and is located in the Sympa bin directory.

The four methods available through the Sympa SOAP server can be tested using this tool. There is no explicit option to tell what acces methos is used. It is inferred based on what options are provided to the script.

Getting the email associated to a session id

You must use the id of a session actually used at the time you launch the command. It is the value of the “sympa_session” cookie set when accessing to the Sympa web interface.

Command line

```
# /home/sympa/bin/sympa_soap_client.pl
 --soap_url=<SOAP server URL>
 --cookie=<cookie identifier>
```

- --soap_url: the URL to your Sympa SOAP server
- --cookie: the value of the “sympa_session” cookie set when accessing to the Sympa web interface.

Expected output

```
error : get_email_cookie
cookie : 65354224256806


getEmailUserByCookie....
0
        'mail@cru.fr'
```

Using the Sympa SOAP functions with the command line tool

It is done by calling the script and providing two kind of arguments :

- the argument required by the service usage : SOAP URL, service name and service parameters,
- the arguments allowing to authenticate the user requesting the service.

Authentication using an HTTP session cookie

Actually, providing the HTTP cookie to a command line sums up in providing a session id, i.e. a simple number. You must use the value of a session cookie actually used at the time you launch the command. It is the “sympa_session” cookie set when accessing to the Sympa web interface.

```
# /home/sympa/bin/sympa_soap_client.pl --soap_url=<SOAP server URL>
                                       --service=<a sympa service>
                                       --service_parameters=<value1,value2,value3>
                                       --session_id=<cookie identifier>
```

The options used are:

- --soap_url: the URL to your Sympa SOAP server;
- --service: the requested SOAP service. See below;
- --service_parameters: the parameters needed to use the service. They must be provided as a comma separated list, without spaces. See below;
- --session_id: the value of the “sympa_session” cookie set when accessing to the Sympa web interface.

Authentication using a user name and password

```
# /home/sympa/bin/sympa_soap_client.pl --soap_url=<SOAP server URL>
                                       --service=<a sympa service>
                                       --service_parameters=<value1,value2,value3>
                                       --user_email=<email>
                                       --user_password=<password>
```

The options used are:

- --soap_url: the URL to your Sympa SOAP server;
- --service: the requested SOAP service. See below;
- --service_parameters: the parameters needed to use the service. They must be provided as a comma separated list, without spaces. See below;
- --user_email: the email of the user requesting the service;
- --user_password: the password of this user.

Access through a trusted application

```
# /home/sympa/bin/sympa_soap_client.pl --soap_url=<SOAP server URL>
                                       --service=<a sympa service>
                                       --service_parameters=<value1,value2,value3>
                                       --cookie=<cookie identifier>
                                       --trusted_application=<app name>
                                       --trusted_application_password=<password>
                                       --proxy_vars=<id=value,id2=value2>
```

The options used are:

- --soap_url: the URL to your Sympa SOAP server;
- --service: the requested SOAP service. See below;
- --service_parameters: the parameters needed to use the service. They must be provided as a comma separated list, without spaces. See below;
- --cookie: the value of the "sympa_session" cookie set when accessing to the Sympa web interface;
- --trusted_application: the trusted application name as defined in `trusted_applications.conf`;
- --trusted_application_password: the password of the trusted application as defined in `trusted_applications.conf`;
- --proxy_vars: the proxy vars of the trusted application as defined in `trusted_applications.conf`.

Sympa SOAP services and the command line tool

This is a description of how to use the Sympa SOAP services using the command line tool. The parameters are given in the same order they must be found in the command tool option `service_parameters`. They must be provided as a comma separated list, without spaces. Don't forget to escape characters that would break the command line, such as spaces, exclamation marks and so on.

Parameters example

If the list of parameters is:

1. list name
2. user email

Then the `service_parameters` option will look like:

--service_parameters=mylist,mail@cru.fr

login

No object here: this is the service used to log when the command tool uses a username and password.

casLogin

No object here.

authenticateAndRun

No object here: this the service used by the command line tool to call the other services, when authentication is done through session id or user name + password.

authenticateRemoteAppAndRun

No object here: this the service used by the command line tool to call the other services, when testing trusted applications.

lists

The parameters are optional.

Parameters:

1. topic: the topic of the lists to return
2. subtopic: the subtopic of this topic

Output example:

```
lists....
0
        'homepage=http://domain.tld/sympa/info/amietestdv01;subject=Amical;listAddress=amietestdv01@domain.tld'
1
        'homepage=http://domain.tld/sympa/info/archeologie;subject=Liste sur l'archéologie;listAddress=archeologie@domain.tld'
2
        'homepage=http://domain.tld/sympa/info/blackmambo;subject=A black mambo;listAddress=blackmambo@domain.tld'
3
        'homepage=http://domain.tld/sympa/info/bluemambo;subject=Another mambo. This one is blue.;listAddress=bluemambo@domain.tld'
```

complexLists

The parameters are optional.

Parameters:

1. topic: the topic of the lists to return
2. subtopic: the subtopic of this topic

Output example:

```
AuthenticateAndRun complexLists....
0
        _homepage_
                'http://domain.tld/sympa-dv/info/amietestdv01'
        _listAddress_
                'amietestdv01@domain.tld'
        _subject_
                'Amical'
1
        _homepage_
                'http://domain.tld/sympa-dv/info/archeologie'
        _listAddress_
                'archeologie@domain.tld'
        _subject_
                'List sur l'archéologie'
2
        _homepage_
                'http://domain.tld/sympa-dv/info/blackmambo'
        _listAddress_
                'blackmambo@domain.tld'
        _subject_
                'A black mambo'
3
        _homepage_
                'http://domain.tld/sympa-dv/info/bluemambo'
        _listAddress_
                'bluemambo@domain.tld'
        _subject_
                'Another mambo. This one is blue.'
```

info

Parameters:

1. listname (mandatory): the name of the list for which info are requested

Output example:

```
```

which

All arguments are mandatory (at least with an empty value).

Parameters:

- no parameters

Output example:

```
which....
0
        'isOwner=1;homepage=http://domain.tld/sympa/info/amietestdv01;subject=Amical;listAddress=amietestdv01@domain.tld;isEditor=0;isSubscriber=0'
1
        'isOwner=1;homepage=http://domain.tld/sympa/info/archeologie;subject=Liste sur l'archéologie;listAddress=archeologie@domain.tld;isEditor=0;isSubscriber=0'
2
        'isOwner=1;homepage=http://domain.tld/sympa/info/blackmambo;subject=A black mambo;listAddress=blackmambo@domain.tld;isEditor=0;isSubscriber=0'
3
        'isOwner=1;homepage=http://domain.tld/sympa/info/bluemambo;subject=Another mambo. This one is blue.;listAddress=bluemambo@domain.tld;isEditor=0;isSubscriber=0'
```

complexWhich

All arguments are mandatory (at least with an empty value).

Parameters:

- no parameters

Output example:

```
complexWhich....
0
        _homepage_
                'http://dev-sympa.cru.fr/sympa-dv/info/redmambo'
        _isEditor_
                '0'
        _isOwner_
                '1'
        _isSubscriber_
                '0'
        _listAddress_
                'redmambo@dev-sympa.cru.fr'
        _subject_
                'Amical'
1
        _homepage_
                'http://dev-sympa.cru.fr/sympa-dv/info/bluemambo'
        _isEditor_
                '1'
        _isOwner_
                '1'
        _isSubscriber_
                '0'
        _listAddress_
                'bluemambo@dev-sympa.cru.fr'
        _subject_
                'Another mambo. This one is blue.'
2
```

```
        _homepage_
                'http://dev-sympa.cru.fr/sympa-dv/info/archeologie'
        _isEditor_
                '1'
        _isOwner_
                '1'
        _isSubscriber_
                '0'
        _listAddress_
                'archeologie@dev-sympa.cru.fr'
        _subject_
                'Liste sur l'archéologie'
3
        _homepage_
                'http://dev-sympa.cru.fr/sympa-dv/info/blackmambo'
        _isEditor_
                '0'
        _isOwner_
                '1'
        _isSubscriber_
                '0'
        _listAddress_
                'blackmambo@dev-sympa.cru.fr'
        _subject_
                'A black mambo'
```

amI

Parameters:

1. list name (mandatory): the name of the list for which the function is tested;
2. function (mandatory): the function the existence of which we will test. The allowed values are: `subscriber`, `owner` and `editor`;
3. user (mandatory): the email address of the user for whom we want to know if she has the function indicated in the target list.

Output example:

```
param: blackmambo
param: owner
param: david.verdin@cru.fr
Using Session_id 48339436597794


AuthenticateAndRun amI....
0
        '1'
```

review

Parameters:

1. the name of the list for which we want the subscribers list (mandatory).

Output example:

```
review....
0
        'mail1@cru.fr'
1
        'mail2@cru.fr'
2
        'mail3@cru.fr'
```

subscribe

Parameters:

1. list name (mandatory)

Output example:

```
subscribe....
0
        '1'
```

signoff

Parameters:

1. list name (mandatory)

Output example:

```
signoff....
0
        '1'
```

add

Parameters:

1. listname (mandatory): the name of the list we want to subscribe the mail address to;
2. email (mandatory): the email to subscribe to the list;
3. gecos: the name under which this email will be subscribed (for example: “John Doe”);

4. quiet: if set to '0', the user doesn't receive a subscription notification

Output example:

```
add....
0
        ''
```

del

Parameters:

1. listname (mandatory): the name of the list we want to unsubscribe the mail address from;
2. email (mandatory): the email of the user to unsubscribe;
3. quiet: if set to '0', the user doesn't receive an unsubscription notification

Output example:

```
del....
0
        '1'
```

createList

Parameters:

1. the list name (mandatory);
2. the subject of the list (mandatory);
3. the template to use (mandatory) (the name of a template found in the `create_list_templates` directory for this Sympa robot;
4. the description of the list (mandatory);
5. the topic of the list (mandatory) (one among the different options existing in topics.conf).

Output example:

```
param: orangemambo
param: Dude !
param: hotline
param: La liste verte
param: computing
Using Session_id 4860001445687


AuthenticateAndRun createList....
0
        '1'
```

closeList

Parameters:

1. the name of the list to close (mandatory).

Output example:

```
param: orangemambo
Using Session_id 4860001445687


AuthenticateAndRun closeList....
0
        '1'
```

manual/soap.txt · Last modified: 2009/10/22 12:10 by david.verdin@cru.fr