phpGroupWare XML-RPC/SOAP Methodology

1. System level requests

1.1. Login and authentication

Authentication for user logins is handled internally no differently than for the typical phpGroupWare login via web browser. Server logins, added for XML-RPC and SOAP, are only slightly different. For either protocol, user and server login and authentication and subsequent requests are handled by their respective server apps, xmlrpc.php and soap.php. A server is identified by a custom HTTP header, without which a normal user login will be undertaken.

A client or server sends the appropriate XML-RPC or SOAP packet containing host, user, and password information to the phpgw server. The server then assigns a sessionid and key, which is returned to the client in the appropriate format.

Our current method for authenticating requests after successful login is via the Authorization: Basic HTTP header to be sent by the client or requesting server. The format of this header is a base64 encoding of the assigned sessionid and kp3 variables, seperated by a ':'.

Further security may be obtained by using SSL on the client and server. In the future, we may encrypt/descrypt the data on either end, or at least provide this as an option. The sessionid and key variables will make this possible, and relatively secure.

1.1.1. system.login

The first request a client will make is the system.login method. Here is a sample of a server login packet in XML-RPC:

```
<?xml version="1.0"?>
<methodCall>
<methodName>system.login</methodName>
<params>
<param>
<value><struct>
<member><name>server_name</name>
<value><string>my.host.name</string></value>
```

```
</member>
<member><name>username</name>
<value><string>bubba</string></value>
</member>
<member><name>password</name>
<value><string>gump</string></value>
</member> </struct></value>
</param>
</params>
</methodCall>
```

And the same in SOAP:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:si="http://soapinterop.org/xsd"
xmlns:ns6="http://soapinterop.org" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body> <ns6:system_login>
<server_name xsi:type=":string">my.host.name</server_name>
<username xsi:type=":string">bubba</username>
<password xsi:type=":string">gump</password>
</ns6:system_login>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The same style of packet would be required for a user/client login. A successful login should yield the following reply:

```
<methodResponse>
<params>
<param>
<value><struct>
<member><name>sessionid</name>
<value><string>cf5c5534307562fc57915608377db007</string></value>
</member>
<member><name>kp3</name>
<value><string>2fe54daa11c8d52116788aa3f93cb70e</string></value>
</member>
</struct></value>
</param>
</params>
</methodResponse>
```

And a failed login:

```
<methodResponse>
<params>
<param>
<value><struct>
<member><name>GOAWAY</name>
<value><string>XOXO</string></value>
</member>
</struct></value>
</param>
</params>
</methodResponse>
```

eqweqw

1.1.2. system.logout

Logout:

```
<?xml version="1.0"?>
<methodCall>
<methodName>system.logout</methodName>
<params> <param>
<value><struct>
<member><name>sessionid</name>
<value><string>ea35cac53d2c12bd05caecd97304478a</string></value>
</member>
<member><name>kp3</name>
<value><string>4f2b256e0da4e7cbbebaac9f1fc8ca4a</string></value>
</member>
</struct></value>
</param>
</params>
</methodCall>
```

Logout worked:

```
<methodResponse>
<params>
```

```
<param>
<value><struct>
<member><name>GOODBYE</name>
<value><string>XOXO</string></value>
</member>
</struct></value>
</param>
</params>
</methodResponse>
```

2. Business layer requests

Once a successful login return packet has been received and sessionid/kp3 have been extracted, every subsequent packet sent to the phpgroupware server must be preceded by an Authorization header. Here is a sample header:

```
POST /phpgroupware/xmlrpc.php HTTP/1.0
User-Agent: PHP XMLRPC 1.0
Host: my.local.host
Authorization: Basic ZDgxNDIyZDRkYjg5NDEyNGNiMzZlMDhhZTdlYzAxZmY6NTU3YzkyYjBmNGE4ZDVlOTUzMzI2YmU2OTQyNjM3YjQ=
Content-Type: text/xml
Content-Length: 875
```

The longish string is a base64 encoding of the $sessionid . ':' . $kp3. For now this is our only supported authentication method. Additional methods would probably also affect the methodCalls. This is certainly open to discussion. Following is a typical request for some contact data:

```
<?xml version="1.0"?>
<methodCall>
<methodName>addressbook.boaddressbook.read_entries</methodName>
<params>
<param>
<value><struct>
<member><name>start</name>
<value><string>1</string></value>
</member>
<member><name>limit</name>
<value><string>5</string></value>
```

```
</member>
<member><name>fields</name>
<value><struct>
<member><name>n_given</name>
<value><string>n_given</string></value>
</member>
<member><name>n_family</name>
<value><string>n_family</string></value>
</member>
</struct></value>
</member>
<member><name>query</name>
<value><string></string></value>
</member>
<member><name>filter</name>
<value><string></string></value>
</member>
<member><name>sort</name>
<value><string></string></value>
</member>
<member><name>order</name>
<value><string></string></value>
</member>
</struct></value>
</param>
</params>
</methodCall>
```

Successful response:

```
<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value><struct>
<member><name>0</name>
<value><struct>
<member><name>id</name>
<value><string>1</string></value>
</member>
<member><name>lid</name>
<value><string></string></value>
</member>
<member><name>tid</name>
```

```
<value><string>n</string></value>
</member>
<member><name>owner</name>
<value><string>500</string></value>
</member>
<member><name>access</name>
<value><string>private</string></value>
</member>
<member><name>cat_id</name>
<value><string>1</string></value>
</member>
<member><name>n_given</name>
<value><string>Alan</string></value>
</member>
</struct></value>
</member>
<member><name>1</name>
<value><struct>
<member><name>id</name>
<value><string>2</string></value>
</member>
<member><name>lid</name>
<value><string></string></value>
</member>
<member><name>tid</name>
<value><string>n</string></value>
</member>
<member><name>owner</name>
<value><string>500</string></value>
</member>
<member><name>access</name>
<value><string>private</string></value>
</member>
<member><name>cat_id</name>
<value><string>1</string></value>
</member>
<member><name>n_given</name>
<value><string>Andy</string></value>
</member>
</struct></value>
</member>
...
```

Unauthorized access attempt returns:

```
<methodResponse>
<params>
<param>
<value><string>UNAUTHORIZED</string></value>
</param>
</params>
</methodResponse>
```

3. More to come...

Documenting every single call will be difficult, but should be done. In leiu of this, please see the class.bo{APPNAME}.inc.php files in each application/inc directory in the phpgroupware cvs. In this file will be a list_methods() function, which returns the information to the server about input/output structure for each call. If the file does not have this function, then it is not yet workable via this interface. As for the actual functions, they are also in this file. Generally, they will all accept associative array input and return same, but not always. This code is in flux, have fun.