XGEN Plus SOAP Api Documentation

What is **XgenPlus** ?

XgenPlus is the most advanced mail server and web mail client which provides fast, secure and reliable emailing along with unified mailing service. It's a total email management and relationship solution that has been designed especially to cater the needs of a company. XgenPlus adds new dimensions to internal communication and brings a whole new meaning to responding to customers by sharing information and pooling resources.

About SOAP:

SOAP, originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It relies on extensible Markup Language (XML) as its message format, and usually relies on other Application Layer protocols (most notably Remote Procedure Call (RPC) and HTTP) for message negotiation and transmission.

Our messaging system support SOAP 1.1 or 1.2, it also support Soap with attachment and WSDL 1.1 & 2.0. Currently we have implemented some soap services which can be used by Soap protocol.

Available XgenPlus Soap services are:

1) Service Name: urn:XgenSoapContactApi

This service is used for ***XgenPlus user authentication***, some basic ***manipulation in user's contact book***. List of available functions under this service:

- **getSoapAuthKey**:

  Purpose: This method is used for validating the user and getting its ***soap authentication key***. This soap auth key is unique and it will used, when ever user use any methods of XgenPlus soap services.

- **getContactCount:**

  Purpose: This method is used for getting total number of contacts in users address book.

- **getGroupList:**

  Purpose: This method is used for getting list of groups and total number of contacts associated in each group of users address book.

- **getContactGroupFinder:**

  Purpose: This method is used for finding contact, in which group(s) it exist.

- **addRecord:**

  Purpose: This function is used for adding a record in users address book.

- **deleteRecord:**

Purpose: This method is used for deleting a specific record from users address book.

- **updateRecord:**

  Purpose: This method is used for updating the contact detail of a specific contact in users address book.

- **getUserKey:**

  Purpose: Only server administration and domain administrator can use this method. It is basically use for getting any of user's soap authentication key.

2) **Service Name: urn:XgenSoapMessageApi**

This service is used for messaging purpose, by using, user can ***send SMS, Mail, Mail with Attachment.*** List of available functions under this service:

- **sendSMS**:

  Purpose: This method is used for sending sms message to a specific mobile number.

- **sendMAIL:**

  Purpose: This method is used for sending simple ***TEXT/HTML*** e-mail; also user can send e-mail to ***CC & BCC.***

- **SendMailWithAttachment:**

  Purpose: This method is used for sending ***TEXT/HTML*** e-mail with Attachment; also user can send e-mail to ***CC & BCC.*** Currently we are supporting one attachment with on e-mail.

3) **Service Name: urn:XgenSoapCalendarApi**

This service is used for maintaining XgenPlus calendar events, tasks, appointments purpose, by using, user can ***view today's events, add or delete events.*** List of available methods under this service:

- **getTodayEventsList**:

  Purpose: This method is used for getting today's event list, so that user can manage his/her schedule for the day.

- **addCalendarEventRecord:**

  Purpose: This method is used for adding an event, task, appointments etc to his/her XgenPlus calendar.

- **deleteCalendarEventRecord:**

  Purpose: This method is used for deleting an event, appointment or any task from his/her XgenPlus calendar.

Sample client side code for using XgenPlus Soap Api services:

```
package xgen.soap.client;

import java.util.*;
import org.w3c.dom.*;
import javax.xml.parsers.*;

import org.apache.axis.AxisFault;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.encoding.ser.JAFDataHandlerDeserializerFactory;
import org.apache.axis.encoding.ser.JAFDataHandlerSerializerFactory;
import org.apache.axis.transport.http.HTTPConstants;
import org.apache.axis.utils.Options;
import org.apache.log4j.Logger;

import javax.activation.DataHandler;
import javax.activation.FileDataSource;
import javax.xml.namespace.QName;
import javax.xml.rpc.ParameterMode;
import javax.xml.soap.AttachmentPart;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPBodyElement;
import javax.xml.soap.SOAPConnection;
import javax.xml.soap.SOAPConnectionFactory;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import java.io.File;
import java.net.URL;

public class xgenSoapClientRequester{

        public xgenSoapClientRequester(){
                //
        }

        public static void main(String[] args){
                if (args == null || args.length == 0  || args.length < 2) {
                        System.err.println("Need email and password argument.");
                        System.exit(8);
                }
                String value = validateUser(args[0],args[1]);
                System.out.println("User key is "+value);
        }

        /* This method uses two parameter
                - email-id
                - password :- password should be encrypted
        */

        public static String validateUser(String nameToLookup, String pwdToLookup){
                String strUV = "";
```

```
String BASEURL = "http://power.dil.in/services/"; //This url specifies the soap server connect with the services.
try{
    /* The information required to invoke the service is --
    First, the URI for the service, as specified in the deployment descriptor, is used as the target URI. Second, one of the method names in the deployment descriptor must be specified. Finally, the URL of the SOAP endpoint must be specified. */

    /* XgenSoapContactApi is nothing but the deployment descriptor file which contains the information like class name, methods, etc */

    URL URLOBJ = new URL(BASEURL + "XgenSoapContactApi"); // XgenSoapContactApi

    /* getSoapAuthKey is the method name which we want to call through apache soap. With this method we have to specify the descriptor file name that works as an id.*/

    QName OBJQNAME = new QName("urn:XgenSoapContactApi", "getSoapAuthKey");

    // Process the arguments.

    Vector vPARAMOBJ = new Vector();
    Vector vPARAMOBJInner = new Vector();

    vPARAMOBJInner.add("nameToLookup"); vPARAMOBJInner.add(XMLType.XSD_STRING);
    vPARAMOBJInner.add(ParameterMode.IN); vPARAMOBJInner.add(nameToLookup);

    vPARAMOBJ.add(vPARAMOBJInner);

    vPARAMOBJInner = new Vector();
    vPARAMOBJInner.add("pwdToLookup"); vPARAMOBJInner.add(XMLType.XSD_STRING);
    vPARAMOBJInner.add(ParameterMode.IN); vPARAMOBJInner.add(pwdToLookup);

    vPARAMOBJ.add(vPARAMOBJInner);

    /* Calling the CreateCall method */

    strUV = CreateCall(URLOBJ, OBJQNAME, vPARAMOBJ, XMLType.XSD_STRING);
}catch(Exception ex){
    System.out.println(" Execption in validating user:"+ex.getMessage());
}
return strUV;
}

/* This method is used to invoke the call. If the call is successfully completed it returns the soap response*/

public static String CreateCall(URL __URLOBJECT, QName __QOBJECT, Vector __VPARAMS, QName __RESPONSETYPE){
String strRSLT = "";
try{
    Object[] oDataSource = null;
    if(__VPARAMS.size()>0)
        oDataSource = new Object[__VPARAMS.size()];
    else
        oDataSource = new Object[1];

    Service service = new Service();
```

```
                        // Build the call.

                        Call call = (Call) service.createCall();
                        call.setTargetEndpointAddress(__URLOBJECT);

                        call.setOperationName(__QOBJECT);
                        /**** setting params ***/
                        if(__VPARAMS.size()>0){
                                Vector vTmp =new Vector();
                                for(int iC=0;iC<__VPARAMS.size();iC++){
                                        vTmp =(Vector) __VPARAMS.get(iC);
                                        call.addParameter((String)vTmp.get(0),                (QName)vTmp.get(1),
(ParameterMode)vTmp.get(2));
                                        oDataSource[iC]=(String)vTmp.get(3);
                                }
                        }

                        // Invoke the call.
                        call.setReturnType(__RESPONSETYPE);
                        strRSLT = (String)call.invoke(oDataSource);
                }catch(Exception ex){
                        System.out.println(" Exception in CreateCall ");
                        strRSLT = "Error while processing";
                }
                return strRSLT;
        }
}
```