

SAWSession Service

```
// This source file is generated by Oracle tools and is subject to change
// It is a utility client for invoking the operations of the Web service port.
// For reporting problems, use the following
// Version = Oracle WebServices (10.1.3.0.0, build 060119.1546.05277)
```

```
package soaptest;
```

```
import oracle.webservices.transport.ClientTransport;
import oracle.webservices.OracleStub;
import javax.xml.rpc.ServiceFactory;
import javax.xml.rpc.Stub;
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.applet.*;
```

```
public class SAWSessionServiceSoapClient extends Applet{
    private soaptest.SAWSessionServiceSoap _port;
```

```
    public SAWSessionServiceSoapClient() throws Exception {
        ServiceFactory factory = ServiceFactory.newInstance();
        _port =
        ((soaptest.SAWSessionService)factory.loadService(soaptest.SAWSessionService.class)).
        getSAWSessionServiceSoap();
```

```
    }
```

```
/**
```

```
 * @param args
```

```
 */
```

```
public static void main(String[] args) {
    try {
        soaptest.SAWSessionServiceSoapClient myPort = new
soaptest.SAWSessionServiceSoapClient();
        soaptest.HtmlViewServiceClient htmlClient = new
soaptest.HtmlViewServiceClient();
        soaptest.ReportEditingServiceSoapClient reportEdit = new
soaptest.ReportEditingServiceSoapClient();
        soaptest.XmlViewServiceSoapClient xmlView = new
soaptest.XmlViewServiceSoapClient();
        StartPageParams newPage = new StartPageParams();
        ReportRef newreportRef = new ReportRef();
        ReportParams newreportParams = new ReportParams();
```

```
ReportHTMLOptions newreportHTMLOptions = new ReportHTMLOptions();
```

```
newPage.dontUseHttpCookies = false;  
newPage.idsPrefix = "1";
```

```
String sessionID = new String();  
String curUser = new String();  
String pageID = new String();  
String reportID = new String();  
String reportPath = new String();  
String reportXML = new String();  
String htmlOutput = new String();  
String bodyHTML = new String();  
String sqlResult = new String();  
String xmlResult = new String();
```

```
sessionID = myPort.logon("Administrator","Administrator");  
curUser = myPort.getCurUser(sessionID);  
pageID = htmlClient.startPage(newPage,sessionID);  
reportID = "Report1";
```

```
reportPath = "/users/administrator/GEC_DW/Regional Sales/Profit per Category  
Pie Chart";
```

```
reportXML = "<saw:report xmlns:saw=\"com.siebel.analytics.web/report/v1\"  
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"  
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"  
xmlns:sawx=\"com.siebel.analytics.web/expression/v1\">\n" +  
" <saw:criteria subjectArea=\"GEC_DW\">\n" +  
" <saw:columns>\n" +  
" <saw:column formula=\"Products.\"Product Category\"\"  
columnID=\"c1\"/>\n" +  
" <saw:column formula=\"Sales.Profit\" columnID=\"c4\">\n" +  
" <saw:displayFormat suppress=\"default\" interaction=\"default\"/>\n" +  
" <saw:columnHeading>\n" +  
" <saw:displayFormat interaction=\"default\"/>\n" +  
" <saw:caption>\n" +  
" <saw:text>Profit per Product  
Category</saw:text></saw:caption></saw:columnHeading>\n" +  
" <saw:tableHeading>\n" +  
" <saw:caption>\n" +  
" <saw:text>Sales</saw:text></saw:caption></saw:tableHeading></saw:column></saw:columns>\n" +  
" <saw:columnOrder/></saw:criteria>\n" +  
" <saw:views>\n" +  
" <saw:view xsi:type=\"saw:compoundView\" name=\"compoundView!1\"  
rptViewVers=\"200510010\">\n" +  
" <saw:cvTable>\n" +
```

```

"      <saw:cvRow>\n" +
"      <saw:cvCell
viewName= \"staticchart!1\"/></saw:cvRow></saw:cvTable></saw:view>\n" +
"      <saw:view xsi:type= \"saw:titleView\" name= \"titleView!1\"
rptViewVers= \"200510010\" includeName= \"true\" startedDisplay= \"none\"/>\n" +
"      <saw:title>\n" +
"      <saw:caption>\n" +
"      <saw:text>Rank Profit By Product
Category</saw:text></saw:caption></saw:title></saw:view>\n" +
"      <saw:view xsi:type= \"saw:tableView\" name= \"tableView!1\"
rptViewVers= \"200510010\"/>\n" +
"      <saw:view xsi:type= \"saw:staticchart\" name= \"staticchart!1\"
rptViewVers= \"200510010\"/>\n" +
"      <saw:template tid= \"charts/pie.cxml\"/>\n" +
"      <saw:canvasFormat bgColor= \"#CCFFCC\"/>\n" +
"      <saw:axesFormats>\n" +
"      <saw:axisFormat axis= \"pievalues\"/>\n" +
"      <saw:scale/></saw:axisFormat></saw:axesFormats>\n" +
"      <saw:legendFormat/>\n" +
"      <saw:variantSelectors>\n" +
"      <saw:variantSelector name= \"subtype\"
value= \"3d\"/></saw:variantSelectors>\n" +
"      <saw:selections>\n" +
"      <saw:categories>\n" +
"      <saw:category position= \"0\"/>\n" +
"      <saw:constant value= \"1\"/></saw:category></saw:categories>\n" +
"      <saw:measures>\n" +
"      <saw:column columnID= \"c4\" position= \"0\"/></saw:measures>\n" +
"      <saw:seriesGenerators>\n" +
"      <saw:column columnID= \"c1\"/>\n" +
"
<saw:measureLabels/></saw:seriesGenerators></saw:selections></saw:view>\n" +
"      <saw:view xsi:type= \"saw:tableView\" name= \"tableView!2\"
rptViewVers= \"200510010\" showHeading= \"false\"/>\n" +
"      <saw:view xsi:type= \"saw:pivotTableView\" name= \"pivotTableView!1\"
rptViewVers= \"200510010\"/>\n" +
"      <saw:edge axis= \"page\"/>\n" +
"      <saw:edge axis= \"section\"/>\n" +
"      <saw:edge axis= \"row\"/>\n" +
"      <saw:edgeLayer type= \"column\" columnID= \"c5\"/></saw:edge>\n" +
"      <saw:edge axis= \"column\"/>\n" +
"      <saw:edgeLayer type= \"column\" columnID= \"c4\"/>\n" +
"      <saw:edgeLayer type= \"labels\"/></saw:edge>\n" +
"      <saw:edge axis= \"measure\"/>\n" +
"      <saw:edgeLayer type= \"column\" columnID= \"c6\"/>\n" +
"      <saw:edgeLayer type= \"column\"
columnID= \"c7\"/></saw:edge></saw:view></saw:views></saw:report>";

```

```

newreportRef.reportPath = reportPath;
    newreportRef.reportXml = reportXML;

    htmlClient.addReportToPage(pageID, reportID, newreportRef, null, null, null,
sessionID);
    htmlOutput = htmlClient.getHtmlForReport(pageID, reportID, sessionID);
    bodyHTML = htmlClient.getCommonBodyHtml(pageID,sessionID);
    sqlResult = reportEdit.generateReportSQL(newreportRef,null,sessionID);
    //xmlResult = xmlView.getResults(newreportRef,"XML",null,null,sessionID);

    System.out.println("User:" + curUser);
    System.out.println("PageID:" + pageID);
    System.out.println(htmlOutput);
    //System.out.println(bodyHTML);
    System.out.println(sqlResult);
    System.out.println(xmlResult);

    int startIndex;
    int endIndex;
    String url = new String();

    startIndex = htmlOutput.indexOf("IFRAME SRC=");
    endIndex = htmlOutput.indexOf("% 3aReport1",startIndex);

    startIndex = startIndex + 12;
    endIndex = endIndex + 10;
    url = "<HTML><BODY><IFRAME SRC=\"\" +
htmlOutput.substring(startIndex,endIndex) + "\"></BODY></HTML>";

    System.out.println(startIndex);
    System.out.println(endIndex);
    System.out.println(url);

    // Printing out the HTML Output

    FileOutputStream htmlFile, htmlFile1;

    try {
        htmlFile = new FileOutputStream ("myfile.html");
        htmlFile1 = new FileOutputStream ("myfile1.html");
        new PrintStream(htmlFile).println (htmlOutput);
        new PrintStream(htmlFile1).println (url);
        htmlFile.close();
        htmlFile1.close();
    }
    // Catches any error conditions

```

```

catch (IOException e)
    {
        System.err.println ("Unable to write to file");
        System.exit(-1);
    }

    Runtime.getRuntime().exec( "\"C:/Program Files/Mozilla
Firefox/firefox.EXE\" \"file:///D:/Oracle
Software/JDeveloper/jdev/mywork/SoapTest/SoapTest/myfile1.html\"" );
    curUser = myPort.getCurUser(sessionID);
    pageID = htmlClient.startPage(newPage,sessionID);
    System.out.println("User:" + curUser);
    System.out.println("PageID:" + pageID);

//
//      JTextPane tp = new JTextPane();
//      JScrollPane js = new JScrollPane();
//      js.getViewPort().add(tp);
//      JFrame jf = new JFrame();
//      jf.getContentPane().add(js);
//      jf.pack();
//      jf.setSize(400,500);
//      jf.setVisible(true);
//
//      try {
//
//          tp.setPage(url);
//          }
//      catch (Exception e) {
//          e.printStackTrace();
//      }

//
//      Thread.currentThread().sleep(10000);
//  }
//  catch (Exception e) {
//      e.printStackTrace();
//  }

//      System.out.println("calling " + myPort.getEndpoint());
//  Add your own code here

catch (Exception ex) {
    ex.printStackTrace();
}

```

```

}

/**
 * delegate all operations to the underlying implementation class.
 */

public String logon(String name, String password) throws java.rmi.RemoteException {
    return _port.logon(name, password);
}

public AuthResult logonex(String name, String password, SAWSessionParameters
sessionparams) throws java.rmi.RemoteException {
    return _port.logonex(name, password, sessionparams);
}

public void logoff(String sessionID) throws java.rmi.RemoteException {
    _port.logoff(sessionID);
}

public void keepAlive(String[] sessionID) throws java.rmi.RemoteException {
    _port.keepAlive(sessionID);
}

public String getCurUser(String sessionID) throws java.rmi.RemoteException {
    return _port.getCurUser(sessionID);
}

public SessionEnvironment getSessionEnvironment(String sessionID) throws
java.rmi.RemoteException {
    return _port.getSessionEnvironment(sessionID);
}

public String impersonate(String name, String password, String impersonateID) throws
java.rmi.RemoteException {
    return _port.impersonate(name, password, impersonateID);
}

public AuthResult impersonateex(String name, String password, String impersonateID,
SAWSessionParameters sessionparams) throws java.rmi.RemoteException {
    return _port.impersonateex(name, password, impersonateID, sessionparams);
}

/**
 * used to access the JAX-RPC level APIs
 * returns the interface of the port instance
 */
public soaptest.SAWSessionServiceSoap getPort() {

```

```

return _port;
    }

    public String getEndpoint() {
        return (String) ((Stub)
_port)._getProperty(Stub.ENDPOINT_ADDRESS_PROPERTY);
    }

    public void setEndpoint(String endpoint) {
        ((Stub) _port)._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, endpoint);
    }

    public String getPassword() {
        return (String) ((Stub) _port)._getProperty(Stub.PASSWORD_PROPERTY);
    }

    public void setPassword(String password) {
        ((Stub) _port)._setProperty(Stub.PASSWORD_PROPERTY, password);
    }

    public String getUsername() {
        return (String) ((Stub) _port)._getProperty(Stub.USERNAME_PROPERTY);
    }

    public void setUsername(String username) {
        ((Stub) _port)._setProperty(Stub.USERNAME_PROPERTY, username);
    }

    public void setMaintainSession(boolean maintainSession) {
        ((Stub) _port)._setProperty(Stub.SESSION_MAINTAIN_PROPERTY,
Boolean.valueOf(maintainSession));
    }

    public boolean getMaintainSession() {
        return ((Boolean) ((Stub)
_port)._getProperty(Stub.SESSION_MAINTAIN_PROPERTY)).booleanValue();
    }

    /**
     * returns the transport context
     */
    public ClientTransport getClientTransport() {
        return ((OracleStub) _port).getClientTransport();
    }
}

```

HTMLView Service

```
// This source file is generated by Oracle tools and is subject to change
// It is a utility client for invoking the operations of the Web service port.
// For reporting problems, use the following
// Version = Oracle WebServices (10.1.3.0.0, build 060119.1546.05277)
```

```
package soaptest;
```

```
import oracle.webservices.transport.ClientTransport;
import oracle.webservices.OracleStub;
import javax.xml.rpc.ServiceFactory;
import javax.xml.rpc.Stub;
```

```
public class HtmlViewServiceClient {
    private soaptest.HtmlViewServiceSoap _port;

    public HtmlViewServiceClient() throws Exception {
        ServiceFactory factory = ServiceFactory.newInstance();
        _port =
((soaptest.HtmlViewService)factory.loadService(soaptest.HtmlViewService.class)).getHtmlViewService();
    }
}
```

```
/**
```

```
 * @param args
```

```
 */
```

```
public static void main(String[] args) {
    try {
        soaptest.HtmlViewServiceClient myPort = new
soaptest.HtmlViewServiceClient();
        System.out.println("calling " + myPort.getEndpoint());
        // Add your own code here

    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

```
/**
```

```
 * delegate all operations to the underlying implementation class.
```

```
 */
```

```
public String startPage(StartPageParams options, String sessionID) throws
java.rmi.RemoteException {
    return _port.startPage(options, sessionID);
}
```

```
public void endPage(String pageID, String sessionID) throws
```

```

java.rmi.RemoteException {
    _port.endPage(pageID, sessionID);
}

    public void addReportToPage(String pageID, String reportID, ReportRef report, String
reportViewName, ReportParams reportParams, ReportHTMLOptions options, String
sessionID) throws java.rmi.RemoteException {
        _port.addReportToPage(pageID, reportID, report, reportViewName, reportParams,
options, sessionID);
    }

    public String getHeadersHtml(String pageID, String sessionID) throws
java.rmi.RemoteException {
        return _port.getHeadersHtml(pageID, sessionID);
    }

    public String getCommonBodyHtml(String pageID, String sessionID) throws
java.rmi.RemoteException {
        return _port.getCommonBodyHtml(pageID, sessionID);
    }

    public String getHtmlForReport(String pageID, String pageReportID, String
sessionID) throws java.rmi.RemoteException {
        return _port.getHtmlForReport(pageID, pageReportID, sessionID);
    }

    public void setBridge(String bridge, String sessionID) throws
java.rmi.RemoteException {
        _port.setBridge(bridge, sessionID);
    }

/**
 * used to access the JAX-RPC level APIs
 * returns the interface of the port instance
 */
    public soaptest.HtmlViewServiceSoap getPort() {
        return _port;
    }

    public String getEndpoint() {
        return (String) ((Stub)
_port)._getProperty(Stub.ENDPOINT_ADDRESS_PROPERTY);
    }

    public void setEndpoint(String endpoint) {
        ((Stub) _port)._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, endpoint);
    }

```

```

public String getPassword() {
    return (String) ((Stub) _port)._getProperty(Stub.PASSWORD_PROPERTY);
}

public void setPassword(String password) {
    ((Stub) _port)._setProperty(Stub.PASSWORD_PROPERTY, password);
}

public String getUsername() {
    return (String) ((Stub) _port)._getProperty(Stub.USERNAME_PROPERTY);
}

public void setUsername(String username) {
    ((Stub) _port)._setProperty(Stub.USERNAME_PROPERTY, username);
}

public void setMaintainSession(boolean maintainSession) {
    ((Stub) _port)._setProperty(Stub.SESSION_MAINTAIN_PROPERTY,
Boolean.valueOf(maintainSession));
}

public boolean getMaintainSession() {
    return ((Boolean) ((Stub)
_port)._getProperty(Stub.SESSION_MAINTAIN_PROPERTY)).booleanValue();
}

/**
 * returns the transport context
 */
public ClientTransport getClientTransport() {
    return ((OracleStub) _port).getClientTransport();
}
}

```

XMLView Service

```

// This source file is generated by Oracle tools and is subject to change
// It is a utility client for invoking the operations of the Web service port.
// For reporting problems, use the following
// Version = Oracle WebServices (10.1.3.0.0, build 060119.1546.05277)

```

```

package soaptest;

```

```

import oracle.webservices.transport.ClientTransport;
import oracle.webservices.OracleStub;
import javax.xml.rpc.ServiceFactory;
import javax.xml.rpc.Stub;

```

```

public class XmlViewServiceSoapClient {
    private soaptest.XmlViewServiceSoap _port;

    public XmlViewServiceSoapClient() throws Exception {
        ServiceFactory factory = ServiceFactory.newInstance();
        _port =
((soaptest.XmlViewService)factory.loadService(soaptest.XmlViewService.class)).getXml
ViewServiceSoap();
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        try {
            soaptest.XmlViewServiceSoapClient myPort = new
soaptest.XmlViewServiceSoapClient();
            System.out.println("calling " + myPort.getEndpoint());
            // Add your own code here

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    /**
     * delegate all operations to the underlying implementation class.
     */

    public Object getResults(ReportRef report, String outputFormat, boolean
encodeInString, ReportParams reportParams, String sessionID) throws
java.rmi.RemoteException {
        return _port.getResults(report, outputFormat, encodeInString, reportParams,
sessionID);
    }

    public QueryResults executeXMLQuery(ReportRef report, XMLQueryOutputFormat
outputFormat, XMLQueryExecutionOptions executionOptions, ReportParams
reportParams, String sessionID) throws java.rmi.RemoteException {
        return _port.executeXMLQuery(report, outputFormat, executionOptions,
reportParams, sessionID);
    }

    public QueryResults executeSQLQuery(String sql, XMLQueryOutputFormat
outputFormat, XMLQueryExecutionOptions executionOptions, String sessionID) throws
java.rmi.RemoteException {
        return _port.executeSQLQuery(sql, outputFormat, executionOptions, sessionID);
    }

```

```

}

    public QueryResults fetchNext(String queryID, String sessionID) throws
java.rmi.RemoteException {
        return _port.fetchNext(queryID, sessionID);
    }

    public void cancelQuery(String queryID, String sessionID) throws
java.rmi.RemoteException {
        _port.cancelQuery(queryID, sessionID);
    }

    /**
     * used to access the JAX-RPC level APIs
     * returns the interface of the port instance
     */
    public soaptest.XmlViewServiceSoap getPort() {
        return _port;
    }

    public String getEndpoint() {
        return (String) ((Stub)
_port)._getProperty(Stub.ENDPOINT_ADDRESS_PROPERTY);
    }

    public void setEndpoint(String endpoint) {
        ((Stub) _port)._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, endpoint);
    }

    public String getPassword() {
        return (String) ((Stub) _port)._getProperty(Stub.PASSWORD_PROPERTY);
    }

    public void setPassword(String password) {
        ((Stub) _port)._setProperty(Stub.PASSWORD_PROPERTY, password);
    }

    public String getUsername() {
        return (String) ((Stub) _port)._getProperty(Stub.USERNAME_PROPERTY);
    }

    public void setUsername(String username) {
        ((Stub) _port)._setProperty(Stub.USERNAME_PROPERTY, username);
    }

    public void setMaintainSession(boolean maintainSession) {
        ((Stub) _port)._setProperty(Stub.SESSION_MAINTAIN_PROPERTY,

```

```

Boolean.valueOf(maintainSession));
    }

    public boolean getMaintainSession() {
        return ((Boolean) ((Stub)
_port)._getProperty(Stub.SESSION_MAINTAIN_PROPERTY)).booleanValue();
    }

    /**
     * returns the transport context
     */
    public ClientTransport getClientTransport() {
        return ((OracleStub) _port).getClientTransport();
    }
}

```

ReportEditing Service

```

// This source file is generated by Oracle tools and is subject to change
// It is a utility client for invoking the operations of the Web service port.
// For reporting problems, use the following
// Version = Oracle WebServices (10.1.3.0.0, build 060119.1546.05277)

```

```
package soaptest;
```

```

import oracle.webservices.transport.ClientTransport;
import oracle.webservices.OracleStub;
import javax.xml.rpc.ServiceFactory;
import javax.xml.rpc.Stub;

```

```

public class ReportEditingServiceSoapClient {
    private soaptest.ReportEditingServiceSoap _port;

    public ReportEditingServiceSoapClient() throws Exception {
        ServiceFactory factory = ServiceFactory.newInstance();
        _port =
((soaptest.ReportEditingService)factory.loadService(soaptest.ReportEditingService.class)
).getReportEditingServiceSoap();
    }
}

```

```

/**
 * @param args
 */
public static void main(String[] args) {
    try {
        soaptest.ReportEditingServiceSoapClient myPort = new
soaptest.ReportEditingServiceSoapClient();
    }
}

```

```

System.out.println("calling " + myPort.getEndpoint());
    // Add your own code here

    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

/**
 * delegate all operations to the underlying implementation class.
 */

public Object applyReportParams(ReportRef reportRef, ReportParams reportParams,
boolean encodeInString, String sessionID) throws java.rmi.RemoteException {
    return _port.applyReportParams(reportRef, reportParams, encodeInString,
sessionID);
}

public String generateReportSQL(ReportRef reportRef, ReportParams reportParams,
String sessionID) throws java.rmi.RemoteException {
    return _port.generateReportSQL(reportRef, reportParams, sessionID);
}

/**
 * used to access the JAX-RPC level APIs
 * returns the interface of the port instance
 */
public soaptest.ReportEditingServiceSoap getPort() {
    return _port;
}

public String getEndpoint() {
    return (String) ((Stub)
_port)._getProperty(Stub.ENDPOINT_ADDRESS_PROPERTY);
}

public void setEndpoint(String endpoint) {
    ((Stub) _port)._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, endpoint);
}

public String getPassword() {
    return (String) ((Stub) _port)._getProperty(Stub.PASSWORD_PROPERTY);
}

public void setPassword(String password) {
    ((Stub) _port)._setProperty(Stub.PASSWORD_PROPERTY, password);
}

```

```
public String getUsername() {
    return (String) ((Stub) _port)._getProperty(Stub.USERNAME_PROPERTY);
}

public void setUsername(String username) {
    ((Stub) _port)._setProperty(Stub.USERNAME_PROPERTY, username);
}

public void setMaintainSession(boolean maintainSession) {
    ((Stub) _port)._setProperty(Stub.SESSION_MAINTAIN_PROPERTY,
Boolean.valueOf(maintainSession));
}

public boolean getMaintainSession() {
    return ((Boolean) ((Stub)
_port)._getProperty(Stub.SESSION_MAINTAIN_PROPERTY)).booleanValue();
}

/**
 * returns the transport context
 */
public ClientTransport getClientTransport() {
    return ((OracleStub) _port).getClientTransport();
}
}
```