Sun Secure Global Desktop 4.4 Web Services Developer Guide

Sun Microsystems, Inc.
www.sun.com

Part No. 820-4291-01
Feb 2008, Revision 01

Submit comments about this document at: `http://docs.sun.com/app/docs/form/comments`

Contents

CHAPTER 1

Introducing SGD Web Services

Pre-release draft. This chapter is not available.

About SGD Web Services

CHAPTER 2

Developing Applications Using WSDL Files

This chapter describes how to use a Web Services Description Language (WSDL) file to create a simple Java™ technology-based client application to access SGD web services.

This chapter includes the following topics:

Generating the Client Stub Classes

A WSDL file is an Extensible Markup Language (XML) file that describes how you can communicate with a web service. The WSDL file defines a web service in terms of the messages you can exchange, and the operations you can perform.

You can use a SOAP framework, such as Apache Axis or Microsoft.NET, to generate client stub classes from a WSDL file. You can then use the generated stub classes to access web services.

A stub class is a local proxy class that represents a remote object on a server. By using a reference to the local stub class, a client can invoke a method on the remote object.

This section describes how to use Apache Axis to generate Java client stub classes using the `webtopsession.wsdl` file. The `webtopsession.wsdl` file defines the `ITarantellaWebtopSession` web service that is used to create and manage SGD user sessions.

Note – This example uses the 1.x version of the Apache Axis SOAP toolkit. See `http://www.apache.org` for more details about the Axis project.

The Apache Axis toolkit has a command line tool, `WSDL2Java`, that you can use to generate Java stub classes from a WSDL file. `WSDL2Java` is included in the `axis.jar` file that is shipped with the Apache Axis distribution.

Other SOAP toolkits provide similar command line tools for generating client stub classes from a WSDL file.

▼ How to Generate Client Stub Classes Using `WSDL2Java`

- **Generate the client stub classes.**

```
$ java org.apache.axis.wsdl.WSDL2Java \
http://server.example.com/axis/services/document/webtopsession?wsdl
```

where *server.example.com* is the name of an SGD server. `WSDL2Java` generates the client stub class files in the current directory.

- Alternatively, you can download a copy of the WSDL file to your hard disk and run `WSDL2Java` on the local file.

Tip – The Apache Axis documentation includes useful information about modifying your CLASSPATH variable in order to run Axis commands. More information on using `WSDL2Java` can be found at `http://ws.apache.org/axis/java/user-guide.html#UsingWSDLWithAxis`.

Overview of the Generated Client Stub Classes

This section discusses the client stub classes that `WSDL2Java` generates.

When you generate stub classes, the namespaces specified in the WSDL file map to Java package names. For example, an entry in the `webtopsession.wsdl` file of
`<schema targetNamespace=`
`"http://sgd.sun.com/webservices/document/webtopsession">`
results in a Java package name of
`com.sun.sgd.webservices.document.webtopsession`.

The client stub classes are created in the following sub-folders that correspond to Java package names:

- `com/sun/sgd/webservices/param/`
- `com/sun/sgd/webservices/document/webtopsession/`

`WSDL2Java` generates the following classes:

- **Item Class.** A key-value utility class, used to store and manipulate objects.
- **Service Endpoint Interface (SEI).** The SEI defines the methods for a particular web service. For the SEI, `WSDL2Java` generates an interface declaration, `ITarantellaWebtopSession`, and an implementation of the interface, `WebtopsessionSoapBindingStub`.
- **Service Interface.** The Service Interface defines a stub that points to the *endpoint URL*. The endpoint URL is the web address of the web service. For the Service Interface, `WSDL2Java` generates an interface declaration, `ITarantellaWebtopSessionService`, and an implementation of the interface, `ITarantellaWebtopSessionServiceLocator`.
- **Message Element Classes.** Classes that define the request and response message elements for the operation.

The following sections discuss the generated classes in more detail.

`Item` Class

The `com/sun/sgd/webservices/param` folder contains the `Item` class . This class defines the `Item` complex data type, a key-value data type used to store and manipulate objects.

The `Item` data type is defined in the WSDL file as follows:

```
<complexType name="Item">
    <sequence>
     <element name="key" nillable="true" type="xsd:string"/>
     <element name="value" nillable="true" type="xsd:anyType"/>
    </sequence>
 </complexType>
```

`WSDL2Java` generates the following class:

```
package com.sun.sgd.webservices.param;

public class Item implements java.io.Serializable {
    private java.lang.String key;
    private java.lang.Object value;
          ...
public java.lang.String getKey() {...}
public void setKey(java.lang.String key) {...}
public java.lang.Object getValue() {...}
public void setValue(java.lang.Object value) {...}
   ...
}
```

A pair of `get` and `set` methods are defined for each of the elements of the complex data type.

`ITarantellaWebtopSession` Class

The `com/sun/sgd/webservices/document/webtopsession` folder contains the `ITarantellaWebtopSession` class. This class is generated from the `<portType>` section in the WSDL file and defines the interface that is used to access the operations of the web service. In the WSDL file, the following definitions apply:

- A *port type* is an abstract definition of a port.
- A *port* is a collection of *operations*.
- An *operation* is a method or function provided by the web service.

The `webtopsession.wsdl` file has the following `<portType>` definition:

```
<wsdl:portType name="ITarantellaWebtopSession">
 <wsdl:operation name="adminEndSession"
    ...
 </wsdl:operation>
    ...
</wsdl:portType>
```

`WSDL2Java` generates the following class:

```
public interface ITarantellaWebtopSession extends java.rmi.Remote
{
public java.lang.String adminEndSession(java.lang.String
sessioncookie, java.lang.String sessionid, boolean logout) throws
java.rmi.RemoteException;
 ...
 ...
}
```

WebtopsessionSoapBindingStub Class

The com/sun/sgd/webservices/document/webtopsession folder contains the WebtopsessionSoapBinding class. The name of this class is generated from the <binding> element in the WSDL file.

Port types are protocol independent. The protocol used is defined in the <binding> section of the WSDL file. The following WSDL code fragment from webtopsession.wsdl defines a SOAP binding using an HTTP transport and using the Document style.

```
<wsdl:binding name="webtopsessionSoapBinding" type=
"impl:ITarantellaWebtopSession">
 <wsdlsoap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="adminEndSession">
    ...
  </wsdl:operation>
    ...
</wsdl:binding>
```

WSDL2Java generates the following class for this port type:

```
public class WebtopsessionSoapBindingStub extends org.apache.axis.client.Stub
implements
com.sun.sgd.webservices.document.webtopsession.ITarantellaWebtopSession {
    ...
public java.lang.String adminEndSession(java.lang.String sessioncookie,
java.lang.String sessionid, boolean logout) throws java.rmi.RemoteException {
   ...
  }
   ...
}
```

ITarantellaWebtopSessionService Class

The com/sun/sgd/webservices/document/webtopsession folder contains the ITarantellaWebtopSessionService class. This class defines a get method for each <port> element listed in the <service> section of the WSDL file.

The following WSDL code in `webtopsession.wsdl` defines a port called `webtopsession` in the `ITarantellaWebtopSessionService` service using the `webtopsessionSOAPBinding` binding defined elsewhere in the WSDL file.

```
<wsdl:service name="ITarantellaWebtopSessionService">
 <wsdl:port binding="impl:webtopsessionSoapBinding" name="webtopsession">
   <wsdlsoap:address location=
"http://server.example.com/axis/services/document/webtopsession"/>
  </wsdl:port>
 </wsdl:service>
```

`WSDL2Java` generates the following class for this port.

```
public interface ITarantellaWebtopSessionService extends javax.xml.rpc.Service
{
public java.lang.String getwebtopsessionAddress();

public com.sun.sgd.webservices.document.webtopsession.ITarantellaWebtopSession
getwebtopsession() throws javax.xml.rpc.ServiceException;

public com.sun.sgd.webservices.document.webtopsession.ITarantellaWebtopSession
getwebtopsession(java.net.URL portAddress) throws
javax.xml.rpc.ServiceException;
    ...
}
```

`ITarantellaWebtopSessionServiceLocator` Class

The `com/sun/sgd/webservices/document/webtopsession` folder contains the `ITarantellaWebtopSessionServiceLocator` class. This class implements the `get` methods defined by the Service Interface.

This class is used to obtain a stub instance that points to the endpoint URL. The endpoint URL is specified in the `<service>` section of the WSDL file.

The following example shows how the endpoint URL is hard coded in the `ITarantellaWebtopSessionServiceLocator` class.

CODE EXAMPLE 2-1 Endpoint URL Definition in the Service Locator Class

```
public class ITarantellaWebtopSessionServiceLocator extends
org.apache.axis.client.Service implements
com.sun.sgd.webservices.document.webtopsession.ITarantellaWebtopSessionService
{
    ...
  // Use to get a proxy class for webtopsession
```

CODE EXAMPLE 2-1 Endpoint URL Definition in the Service Locator Class

```
   private java.lang.String webtopsession_address =
   "http://server.example.com/axis/services/document/webtopsession";

   public java.lang.String getwebtopsessionAddress() {
      return webtopsession_address;
     }
      ...
public com.sun.sgd.webservices.document.webtopsession.ITarantellaWebtopSession
getwebtopsession() throws javax.xml.rpc.ServiceException {
    java.net.URL endpoint;
    try {
          endpoint = new java.net.URL(webtopsession_address);
        }
          catch (java.net.MalformedURLException e) {
          throw new javax.xml.rpc.ServiceException(e);
        }
    return getwebtopsession(endpoint);
    }
   ...
 }
```

Message Element Classes

The `com/sun/sgd/webservices/document/webtopsession` folder contains message element class files. A pair of message element class files is generated for each operation, one for the request message, and one for the response message.

Message elements are defined in the `<types>` section of the WSDL file. The request message is designated by an element with the same name as the operation, for example, `<element name="adminEndSession">`. The response message is designated by an element with *response* added to the name of the operation, for example, `<element name="adminEndSessionResponse">`.

The following section of a WSDL file defines the `adminEndSession` request message elements.

```
<types>
 <schema>
    ...
  <element name="adminEndSession">
   <complexType>
    <sequence>
      <element name="sessioncookie" type="xsd:string"/>
      <element name="sessionid" type="xsd:string"/>
      <element name="logout" type="xsd:boolean"/>
    </sequence>
   </complexType>
  </element>
   ...
 </schema>
</types>
```

WSDL2Java generates the following class for the request message. For each of the message elements, get and set methods are generated.

```
package com.sun.sgd.webservices.document.webtopsession;

public class AdminEndSession implements java.io.Serializable {
    private java.lang.String sessioncookie;
    private java.lang.String sessionid;
    private boolean logout;
          ...
public java.lang.String getSessioncookie() {...}
public void setSessioncookie(java.lang.String sessioncookie) {...}
public java.lang.String getSessionid() {...}
public void setSessionid(java.lang.String sessionid) {...}
public boolean isLogout() {...}
public void setLogout(boolean logout) {...}
   ...
}
```

Creating a Client Application

This section describes how you can create a simple Java client application using the stub classes generated by WSDL2Java.

The client application example described in this section is extremely simple. It creates a new user session on the SGD server, and then returns the SOAP response to the client. The example illustrates how a SOAP call can be constructed using just a few lines of Java code.

The Client Application Code

Using a text editor, create the following file and save it as `Client1.java` in the `com/sun/sgd/webservices/document/webtopsession/` folder. Replace *user* and *passwd* with a valid SGD user name and password.

CODE EXAMPLE 2-2 A Java Client Using the Stub Classes Generated by `WSDL2Java`

```
package com.sun.sgd.webservices.document.webtopsession;

public class Client1 {
  public static void main(String[] args) throws Exception
  {
  //  Create a new Service object
  ITarantellaWebtopSessionServiceLocator wtoplocator =
  new ITarantellaWebtopSessionServiceLocator();

  // Use getportname() to create a new client stub
  ITarantellaWebtopSession wtop = wtoplocator.getwebtopsession();

  // Invoke the method on the stub
  String resp = wtop.authenticate("user","passwd","clientid","en");

  // Print the SOAP response string
  System.out.println(resp);
  }
}
```

The example code uses the `authenticate` operation of the `ITarantellaWebtopSession` service. This operation authenticates a user, and then starts a user session on the SGD server. Amongst other attributes, a unique `sessioncookie` ID is returned to the client.

CODE EXAMPLE 2-2 does not end the active user session. This is done using the `endSession` operation and is outlined in "Extending the Client Application" on page 13.

Running the Client Application

This section describes how to compile the client application class files, and run the client application.

▼ How to Run the Client Application

1. **Open a terminal window.**
2. **Change to the top level directory for the generated stub files.**

   The top level directory is one level above the `com/` folder.
3. **Compile the class files.**

```
$ javac com/sun/sgd/webservices/document/webtopsession/Client1.java
```

   The `.java` source files are compiled into `.class` files in the `webtopsession/` and `param/` directories.

Note – Setting the CLASSPATH and debugging Java code compilation errors is beyond the scope of this document. The Apache Axis documentation contains information about modifications to your CLASSPATH variable.

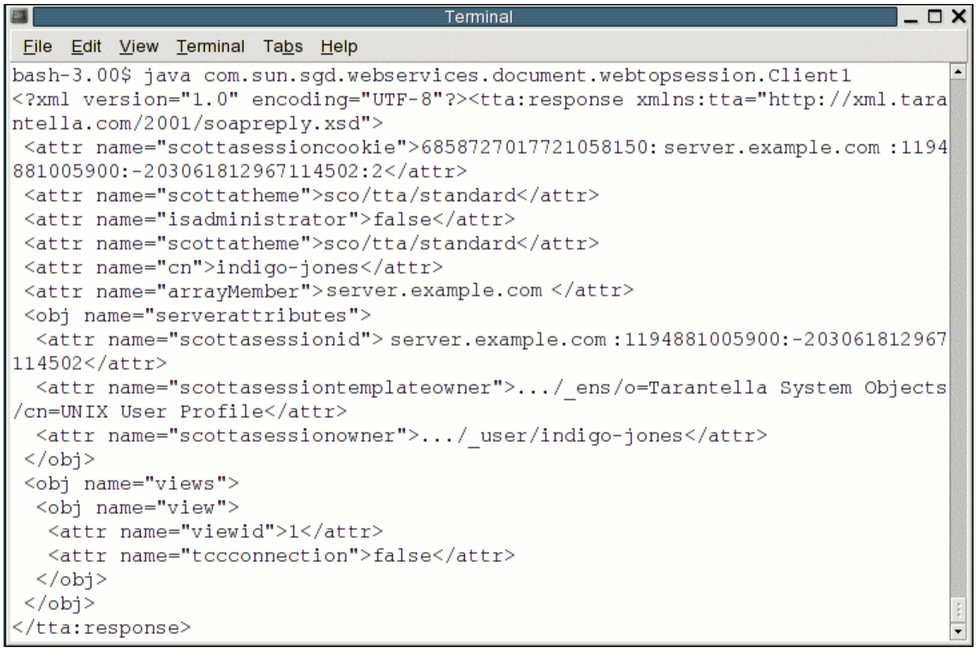
---

4. **Run the client application.**

```
$ java com.sun.sgd.webservices.document.webtopsession.Client1
```

   Following successful execution of the client, a new user session is started on the SGD server, and a SOAP response string is returned to the terminal window.

FIGURE 2-1 SOAP Response String For The Client Application



```
bash-3.00$ java com.sun.sgd.webservices.document.webtopsession.Client1
<?xml version="1.0" encoding="UTF-8"?><tta:response xmlns:tta="http://xml.tara
ntella.com/2001/soapreply.xsd">
 <attr name="scottasessioncookie">6858727017721058150: server.example.com :1194
881005900:-203061812967114502:2</attr>
 <attr name="scottatheme">sco/tta/standard</attr>
 <attr name="isadministrator">false</attr>
 <attr name="scottatheme">sco/tta/standard</attr>
 <attr name="cn">indigo-jones</attr>
 <attr name="arrayMember">server.example.com </attr>
 <obj name="serverattributes">
  <attr name="scottasessionid"> server.example.com :1194881005900:-203061812967
114502</attr>
  <attr name="scottasessiontemplateowner">.../_ens/o=Tarantella System Objects
/cn=UNIX User Profile</attr>
  <attr name="scottasessionowner">.../_user/indigo-jones</attr>
 </obj>
 <obj name="views">
  <obj name="view">
   <attr name="viewid">1</attr>
   <attr name="tccconnection">false</attr>
  </obj>
 </obj>
</tta:response>
```

Alternatively, you can use a SOAP monitoring program to observe the response. A suitable utility program, `tcpmon`, is supplied in the Apache Axis distribution.

Extending the Client Application

The simple client program can be extended in the following ways:

- Change the service endpoint URL.
- Parse the XML response string to extract the session cookie and other important data.
- Get the assigned applications for the user.
- Log out and close down the user session.

Changing the Service Endpoint URL

When you use Axis to generate client stubs from a WSDL file, the endpoint URL specified in the WSDL file is hard coded in the client stub classes. However, when you come to deploy the client application you might need to update the endpoint URL. Any of the following methods can be used to update the endpoint URL. In the following examples, *server.example.com* is the name of an SGD server.

- **Use a different WSDL file.** Obtain a WSDL file from the SGD server that you wish to use with the client application. Use `WSDL2Java` to generate stub classes.
- **Modify your client application code.** You can specify an endpoint URL using the `getwebtopsession` method, as shown in the following example.

```
java.net.URL new_endpoint= \
new java.net.URL("http://server.example.com/axis/services/document/webtopsession");
ITarantellaWebtopSession wtop=wtoplocator.getwebtopsession(new_endpoint);
```

- **Modify the WSDL file.** The endpoint URL is defined by the `<address location>` field in the `<service>` section of the WSDL file, as shown in the following example.

```
<service name="ITarantellaWebtopSessionService">
 <port binding="impl:webtopsessionSoapBinding" name="webtopsession">
  <address location=
"http://server.example.com/axis/services/document/webtopsession"/>
 </port>
</service>
```

  After modifying the WSDL file, use `WSDL2Java` to generate stub classes.

- **Modify the client stub class code.** The endpoint URL is hard coded in the `ITarantellaWebtopSessionServiceLocator` class using the `webtopsession_address` string, as shown in the following example.

```
// Use to get a proxy class for webtopsession
private java.lang.String webtopsession_address = \
"http://server.example.com/axis/services/document/webtopsession";
```

Parsing the SOAP Response

The SOAP response message from the server can be parsed to extract object and attribute information.

The most important attribute you might wish to extract from the SOAP response is the value of the *session cookie*. This value uniquely defines the user session, and is a mandatory input parameter for most SGD web service operations.

The easiest way to parse a SOAP response from an SGD server is to use the custom SGD web services parser classes from the `com.tarantella.tta.webservices.parser` package. See the SGD Javadoc™ documentation for more details about these classes.

You can also use a third party parser, such as Xerces, to parse the XML SOAP response. See `http://xerces.apache.org` for more information.

Getting the Assigned Applications

Assigned applications are the applications and documents that a user can access through SGD.

The `ITarantellaWebtopContent` web service can be used to query and list the assigned applications for a user. See Chapter 16 for more information on how you can use this web service.

Ending a User Session

The `endSession` operation is used to log a user out of SGD and terminate the user session.

The following code fragment illustrates how to end a user session.

```
// Invoke the method on the stub
String resp = wtop.endSession(sessionCookie,true);

// Print the SOAP response string
System.out.println(resp);
```

The `endSession` method uses the session cookie value. "Parsing the SOAP Response" on page 14 describes how this value can be extracted from the SOAP response.

CHAPTER 3

Developing Applications With the ITarantella Package

Pre-release draft. This chapter is not available.

Using the ITarantella Package

CHAPTER 4

Developing Applications With the SessionBean Package

Pre-release draft. This chapter is not available.

Using the SessionBean Package

CHAPTER 5

Developing Applications With JavaServer Pages

Pre-release draft. This chapter is not available.

Using JavaServer Pages

CHAPTER 6

Example Administrator Applications

This chapter discusses sample applications that are included with SGD. The sample applications use SGD web services to perform SGD Administrator-level tasks.

This chapter includes the following topics:

- "Print Manager (`printman`)" on page 23

Print Manager (`printman`)

The Print Manager application, `printman`, is an application that enables an SGD Administrator to view and manage print jobs in the print queue on an SGD server.

The Main Features of the Application

The Print Manager application consists of several JavaServer Pages (JSP) files. This section looks at some of the main features of the JSP code. Each JSP file in the application is then presented in more detail.

Interesting features of the JSP code for this application include the following:

- Using the `SessionBean` class
- Running SGD `tarantella` commands
- Session tracking in the application
- Handling exceptions

Using the SessionBean Class

This application uses classes from the com.tarantella.webservices.client.views package. This package is also called the client.views.* package.

The client.views.* package includes the SessionBean class, com.tarantella.tta.webservices.client.views.SessionBean.

The SessionBean class represents an SGD user session. SessionBean can be used to do the following:

- Authenticate users and create new user sessions.
- Start the SGD Client
- Cache data for a user session.
- Maintain views of print jobs, events, and assigned applications (*webtop content*).

Using SessionBean and related classes in the client.views.* package simplifies the applications development procedure when using SGD web services.

See Chapter 4 for information about using the client.views.* package.

Running SGD tarantella Commands

The Print Manager application uses the ITarantellaAdmin.runCommand method to implement an SGD command line.

The Print Manager application runs the following SGD commands to list and manage print jobs:

- tarantella print list
- tarantella print cancel

See Chapter 8 for more information about the ITarantellaAdmin web service.

Session Tracking in the Application

The Print Manager application manages HTTP sessions by storing a SessionBean object to the pageContext.SESSION_SCOPE context of the JSP.

The sessionObj session bean containing the user session information is *stored* to the JSP pageContext object as follows:

```
pageContext.setAttribute(TTA_SESSION,sessionObj,
PageContext.SESSION_SCOPE);
```

The session bean containing the session information is *retrieved* from the JSP `pageContext` object as follows:

```
SessionBean sessionObj = (SessionBean)pageContext.getAttribute \
(TTA_SESSION,sessionObj, PageContext.SESSION_SCOPE);
```

Handling Exceptions

Errors, such as being unable to start a user session, cause a `java.rmi.RemoteException` to be thrown. The exception includes the SOAP fault that contains the reason for the error and, if appropriate, what additional data is required.

How the Print Manager application accesses the exception data depends on the implementation used for the client-side APIs. For example, if using the implementation supplied in the SGD web services infrastructure, the exception is an instance of `com.tarantella.tta.webservices.TTAException`.
If the implementation was generated from WSDL using the Axis utility `WSDL2Java`, then the exception is an instance of `org.apache.axis.AxisFault`.

The `handleException` utility method of the application detects and handles each type of exception accordingly.

Locating the Source Files

The JSP source files for the Print Manager application are in the SGD installation directory at:
/*install-dir*/webserver/tomcat/*version*/webapps/sgd/admin/printman.

The application consists of the following files:

File Name	Description
`index.jsp`	The entry page for the Print Manager application. Starts an SGD user session and authenticates the user.
`printman.jsp`	Displays a listing of print jobs in the print queue.
`printjob.jsp`	Enables the user to view more details for a specified print job or to delete a job from the print queue.

Each of these files is discussed in more detail in this chapter.

Starting the Print Manager Application

To start the Print Manager application, go to `http://`*server.example.com*`/sgd/admin/printman`, where *server.example.com* is an SGD server. The login page for the application is displayed.

Note – Print Manager is an Administrator-level application. You must log in to the application using an SGD Administrator account or a superuser user account.

The `index.jsp` File

The `index.jsp` file is the entry page for the Print Manager application and performs the following operations:

- Creates a new `SessionBean` object.
- Displays the login form.
- Authenticates the user, and starts a new user session.
- Loads the `printman.jsp` page, if authentication is successful.

Creating a New `SessionBean` Object

The Print Manager application initializes the following constants:

- `MY_LOGIN_FORM` – a string that defines the HTML login form.
- `TTA_SESSION` – a string tag used to identify a `SessionBean` object stored to the JSP `pageContext`.

The application then checks whether a `SessionBean` object already exists. If it does not, the application creates a new `SessionBean` object and saves it to the JSP `pageContext` object, as follows:

```
sessionObj = new SessionBean();
...
pageContext.setAttribute(TTA_SESSION, sessionObj,
PageContext.SESSION_SCOPE);
```

Authenticating the User

The application uses `SessionBean.isAuthenticated` to check if the user has already been authenticated. If not, the username and pasword fields entered at the login form are retrieved from the HTTP Request using the `getParam` utility method.

The `SessionBean.authenticate` method authenticates the user and then starts a user session, as follows:

```
if (!sessionObj.isAuthenticated())
{
    String username = getParam(request, "username");
    String password = getParam(request, "password");
    ...
    sessionObj.authenticate(username,password,"Me","en" );
...
}
```

If authentication is successful, the application loads, as follows:

```
<jsp:forward page="printman.jsp" />
```

If authentication, is successful, the response returns data of use to an application developer. The data returned includes the user's name and the security level of the connection. The Print Manager application ignores any such returned data.

The `printman.jsp` page

The `printman.jsp` page performs the following tasks:

- Creates an `ITarantellaAdmin` object
- Generates a list of the available print jobs for the SGD array
- Creates URL links to show more information for each print job
- Creates URL links to delete each print job

Creating an `ITarantellaAdmin` Object

The application retrieves the `SessionBean` for the current user session from the JSP's `pageContext` as follows:

```
SessionBean sessionObj = (SessionBean) pageContext.getAttribute
(TTA_SESSION, PageContext.SESSION_SCOPE);
```

The `SessionBean.getLocator` method creates an instance of `IServiceLocator`, which loads the SGD implementation of the client proxy classes.

```
IServiceLocator locator = sessionObj.getLocator();
```

The `IServiceLocator.getAdmin` method loads an implementation of `ITarantellaAdmin` as follows:

```
ITarantellaAdmin admin = locator.getAdmin();
```

Generating a List of Print Jobs

The Print Manager application uses `ITarantellaAdmin.runCommand` to implement a `tarantella print list` command. This command returns an XML SOAP response containing a listing of print jobs.

The following code fragment shows how a `tarantella print list` command is implemented in the Print Manager application.

```
String[] objAttrs = new String[0];
    ...
{
    ...
ITarantellaAdmin admin = locator.getAdmin();
xml = admin.runCommand \
(sessionObj.getSessionCookie(), "print","list", objAttrs);
    ...
}
```

If you specify an empty parameter string array `objAttrs`, all print jobs across the SGD array are returned.

SGD includes the custom parser classes, `parser.IResponse` and `parser.ResponseParser`. These classes can be used to parse SOAP responses.

The Print Manager application uses `IResponse.getAttribute` to parse the XML response from the `tarantella print list` command and obtain attributes for each of the available print jobs, as follows:

```
resp = parser.parse(xml);
enm = resp.getChildren();
while (enm.hasMoreElements())
{
resp = (IResponse) enm.nextElement();
...
htmlBuffer.append(resp.getAttribute (ISchemaNames.PRINT_FILENAME)[0]);
...
htmlBuffer.append(resp.getAttribute (ISchemaNames.PRINT_DOCNAME)[0]);
...
htmlBuffer.append(resp.getAttribute (ISchemaNames.PRINT_THIRD_TIER_USER)[0]);
...
}
```

The following print job attribute values are extracted from the SOAP response. Each of the attributes is defined in `ISchemaNames`.

Attribute Name	Description
`scottaprintfilename`	A unique JobID for the print job.
`scottaprintname`	The name of the document to be printed.
`scottaprintsize`	The size of the print file in bytes.
`scottaprinttimestamp`	A unique timestamp, allocated when a print job joins the queue.
`scottaprintusername`	The object name for the originator of the print job.
`scottaprintt3username`	The user account on the application server that owns the print job.

Creating URL Links

For each print job, the application creates URL links which enable you to either display more details about the job or to delete the job from the print queue.

An `action=list` parameter in the HTTP GET query string of the URL link shows more details for the print job.

```
urlBuffer.append("printjob.jsp?action=list&job="...);
```

An `action=cancel` parameter in the HTTP GET query string of the URL link deletes the print job.

```
urlBuffer.append("printjob.jsp?action=cancel&job="...);
```

The `printjob.jsp` page

The `printjob.jsp` page performs the following tasks:

- Creates an `ITarantellaAdmin` object
- Lists attribute information for a selected print job
- Cancels printing for a selected print job

Creating an `ITarantellaAdmin` Object

On loading the `printjob.jsp` page, an `ITarantellaAdmin` object is created in the same way as for the `printman.jsp` page. See "Creating an `ITarantellaAdmin` Object" on page 27.

Processing the Selected Print Job

The `job=` parameter in the HTTP GET request string specifies the `scottaprintfilename` attribute for the selected print job.

The `action=` parameter specified in the HTTP GET request string specifies the action required (`list` or `cancel`) for the selected print job.

The application uses `ITarantellaAdmin.runCommand` to implement a `tarantella print list` command or a `tarantella print cancel` command as appropriate, depending on the `action=` parameter. An XML SOAP response is returned. The following code fragment shows how the the selected print job is processed using `runCommand`.

```
String jobName = request.getParameter("job");
String[] objAttrs = new String[] {"--jobid",jobName};
...
{
     ...
String action = request.getParameter("action");
    ...
xml = admin.runCommand(sessionObj.getSessionCookie(), "print",action,objAttrs);
    ...
}
```

The application uses `IResponse.getAttribute` as follows to parse the XML response and obtain attributes for the specified print job, as follows:

```
ResponseParser parser = new ResponseParser();
IResponse resp;
Enumeration enm;
    ...
resp = parser.parse(xml);
enm = resp.getChildren();
    ...
{
    resp = (IResponse) enm.nextElement();
    enm = resp.getAttributeNames();
        ...
    while (enm.hasMoreElements())
        {
                String name = (String) enm.nextElement();
                String[] values = resp.getAttribute(name);
                    ...
           }
                ...
 }
```

Note – The `cancel` action only returns a single attribute, `scottaprintfilename`. The `list` action returns a more complete list of print job attributes.

CHAPTER 7

Example User Applications

Pre-release draft. This chapter is not available.

About User Applications

CHAPTER 8

The ITarantellaAdmin Web Service

This chapter describes how you can use the `ITarantellaAdmin` web service to run an SGD command line. The available operations and SOAP message elements for this web service are described in detail.

This chapter includes the following topics:

Using ITarantellaAdmin

This section describes how to use the `ITarantellaAdmin` web service to run an SGD `tarantella` command. The supported `tarantella` commands and subcommands are listed.

About the SGD Command Line

An SGD command line has the following format:

```
tarantella command subcommand [option...]
```

Note – The `subcommand` and `option` fields are not required for all SGD commands.

For example, the following command line lists the contents of the `o=applications` object in the SGD datastore.

```
tarantella object list_contents --name "o=applications"
```

In this example, `object` is the *command*, `list_contents` is the *subcommand*, and `--name` and `"o=applications"` are the command line *options*.

Supported Commands

Not all SGD commands are supported by the `ITarantellaAdmin` web service. For example, commands which stop or start SGD services are not available when using `ITarantellaAdmin`.

The following table lists the SGD commands that you can implement using `ITarantellaAdmin`.

TABLE 8-1 SGD Commands Supported by `ITarantellaAdmin`

Command	**Description**
`array`	Creates and manages arrays of SGD servers.
`config`	Edits global and server-specific configuration for an array of SGD servers.
`emulatorsession`	Lists and controls application sessions.
`license`	Adds, lists, and removes SGD license keys.
`object`	Manipulates objects in the SGD datastore.
`passcache`	Manipulates the SGD password cache.
`print`	Controls SGD printing services.
`query`	Examines the SGD server's log files.
`role`	Configures role occupants and their assigned applications.
`status`	Shows the current status of SGD servers in the array.
`tokencache`	Manipulates the token cache used by the authentication token autentication mechanism.
`version`	Displays version information for installed SGD packages.
`webtopsession`	Lists and controls user sessions.

For the SGD commands that are supported by `ITarantellaAdmin`, not all of the possible subcommands are available when using SGD web services. The following table indicates which subcommands are supported by `ITarantellaAdmin`.

TABLE 8-2 Subcommands Supported by `ITarantellaAdmin`

Command	Subcommand	Supported?
`array`	`detach`	✓
	`join`	✓
	`list`	✓
	`make_primary`	✓
`config`	`edit`	✓
	`list`	✓
`emulatorsession`	`end`	✓
	`info`	✓
	`list`	✓
	`shadow`	✓
	`suspend`	✓
`license`	`add`	✓
	`info`	✓
	`list`	✓
	`query`	✓
	`remove`	✓
	`status`	✓

TABLE 8-2 Subcommands Supported by `ITarantellaAdmin` *(Continued)*

Command	Subcommand	Supported?
`object`	`add_host`	✓
	`add_link`	✓
	`add_member`	✓
	`delete`	✓
	`edit`	✓
	`help`	✓
	`list_attributes`	✓
	`list_contents`	✓
	`new_3270app`	✗
	`new_5250app`	✗
	`new_charapp`	✓
	`new_container`	✓
	`new_dc`	✓
	`new_doc`	✓
	`new_group`	✓
	`new_host`	✓
	`new_orgunit`	✓
	`new_person`	✓
	`new_windowsapp`	✓
	`new_xapp`	✓
	`remove_host`	✓
	`remove_link`	✓
	`remove_member`	✓
	`rename`	✓
	`script`	✗
`passcache`	`delete`	✓
	`edit`	✗
	`list`	✓
	`new`	✓
`print`	`cancel`	✓
	`list`	✓
	`move`	✗
	`pause`	✗
	`resume`	✗
	`start`	✗
	`status`	✓
	`stop`	✗

TABLE 8-2 Subcommands Supported by `ITarantellaAdmin` *(Continued)*

Command	Subcommand	Supported?
`query`	`audit`	✗
	`billing`	✗
	`errlog`	✗
	`uptime`	✓
`role`	`add_link`	✓
	`add_member`	✓
	`list`	✓
	`list_links`	✓
	`list_members`	✓
	`remove_link`	✓
	`remove_member`	✓
`status`	–	–
`tokencache`	`delete`	✓
	`list`	✓
`version`	–	–
`webtopsession`	`list`	✓
	`logout`	✓

The *Sun Secure Global Desktop 4.4 Reference Manual* includes more information about the available `tarantella` commands.

Running an SGD Command

To run an SGD command, use the `runCommand` operation. The following example shows how to implement a `tarantella object list_attributes` command:

```
ITarantellaAdmin admin = locator.getAdmin();
String[] desAttrs = new String[]  \
 {"--name",
  "o=organization/cn=indigo-jones",
  "--attribute",
  ISchemaNames.SURNAME,
  "--attribute",
  ISchemaNames.COMMON_NAME
  "--attribute",
  ISchemaNames.USER};

String xml = admin.runCommand (sessionCookie,"object", \
"list_attributes",desAttrs);
```

SOAP Reference

This section describes the operations and SOAP message elements for the `ITarantellaAdmin` web service.

The following table lists the WSDL file location, namespace information, and end point URL for the `ITarantellaAdmin` web service, where *server.example.com* is an SGD server.

WSDL File	`http://`*server.example.com*`/axis/services/document/admin?wsdl`
SOAP Operation Namespace	`http://sgd.sun.com/webservices/document/admin`
HTTP POST URL	`http://`*server.example.com*`/axis/services/document/admin`

Request Message Elements

The following table lists the request message elements for the `ITarantellaAdmin` web service.

Element	Data Type	Description
`sessioncookie`	`xsd:string`	Session cookie of the user session performing the operation.
`command`	`xsd:string`	The SGD `tarantella` command to run, for example, `config`.
`subcommand`	`xsd:string`	Subcommand for the specified SGD command, if applicable. For example, `list`.
`parameters`	`xsd:string`	Array of options for the command line, if applicable. For example, `--server-login`.

Response Message Elements

This section describes the elements returned in response messages for the `ITarantellaAdmin` web service.

<tta:response> Element

The <tta:response> element is the top-level container used for all responses from SGD web services.

An empty response (<tta:response/>) indicates the following:

- The operation was successful
- The operation has no data to return

The <tta:response> element can contain multiple <attr> or <obj> child elements. The returned elements depend on the SGD command implemented by the runCommand operation.

The following example shows the format of a response message for a command line of tarantella config list --server-login.

```
<tta:response>
 <obj name="local">
  <attr name="tarantella-config-host-acceptnewconnections">enabled</attr>
 </obj>
 <obj name="global">
  <attr name="tarantella-config-host-acceptnewconnections">enabled</attr>
 </obj>
</tta:response>
```

Operations

The following table lists the available operations for the ITarantellaAdmin web service.

Operation Name	Purpose
runCommand	Runs an SGD tarantella command

runCommand

Runs an SGD tarantella command line.

Request Message

The format of the request message for this operation is as follows:

<runCommand>

```
 <sessioncookie>xsd:string</sessioncookie>
 <command>xsd:string</command>
 <subcommand>xsd:string</subcommand>
 <parameters>xsd:string</parameters> [1..unbounded]
</runCommand>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
 <xml>xsd:string</xml>
</ttaresponse>
```

Note – The returned XML response string varies, depending on which SGD command you implement using `runCommand`.

CHAPTER 9

The ITarantellaDatastore Web Service

Pre-release draft. This chapter is not available.

Using ITarantellaDatastore

CHAPTER 10

The ITarantellaEmulatorsession Web Service

Pre-release draft. This chapter is not available.

Using ITarantellaEmulatorsession

CHAPTER 11

The ITarantellaEvent Web Service

Pre-release draft. This chapter is not available.

Using ITarantellaEvent

CHAPTER 12

The ITarantellaExternalAuth Web Service

Pre-release draft. This chapter is not available.

Using ITarantellaExternalAuth

CHAPTER 13

The ITarantellaPrint Web Service

This chapter describes how you can use the ITarantellaPrint web service to perform print queue operations and search the SGD print queue. The available operations and SOAP message elements for this web service are described in detail.

This chapter includes the following topics:

- "Using ITarantellaPrint" on page 51
- "SOAP Reference" on page 58

Using ITarantellaPrint

The ITarantellaPrint web service enables an SGD user to manage their own print queue on an SGD server. ITarantellaPrint also includes operations that enable an SGD Administrator to search for print jobs in the SGD print queue. This section describes how you can use the ITarantellaPrint web service to do the following:

- Cancel print jobs
- Look up information about a print job
- Search for print jobs
- Print out print jobs

The ITarantellaPrint web service defines the following types of operations:

- **User-level operations.** These operations enable users to manage their own print jobs on an SGD server.
- **Administrator-level operations.** These operations enable an SGD Administrator to search the SGD print queue.

Administrator-level operations are designated by an *admin* prefix to the operation name, for example, adminSearchStart.

For other Administrator-level operations, for example, where an SGD Administrator needs to to manage the print jobs of other users, you can use the `ITarantellaAdmin.runCommand` operation to implement an SGD `tarantella print` command line. See Chapter 8 for more details about the `ITarantellaAdmin` web service.

Note – The `printman` sample application included in the SGD distribution uses `ITarantellaAdmin.runCommand` to query and control the print queue on an SGD server. See Chapter 6 for more information about this sample application.

About SGD Printing Services

SGD enables users to print from Windows, X, and character applications to a printer attached to their client device. SGD does this by cooperating with the `lp` or `lpr` printing system on the SGD host and the native printing system on the application server.

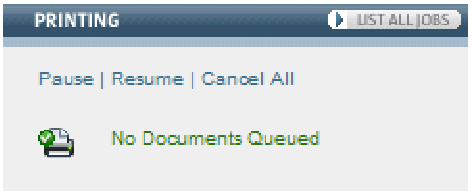
When a user prints, the print job is sent from the application server to an SGD server. The SGD server then sends the print job to the client, which sends it to the user's printer.

SGD uses distributed printing. Print jobs are always sent to the SGD server hosting the application session for the application. This means that a user's print jobs are distributed across the array.

The SGD Webtop Printer Tray

The default SGD webtop features a printer tray where users can manage their own print jobs. The printer tray indicates the number of jobs currently in the print queue and includes controls for print queue management.

FIGURE 13-1 Printer Tray on the SGD Webtop



The default SGD webtop uses the `client.views.*` package to implement the functionality for the printer tray area. However, the lower-level `ITarantellaPrint` operations can be used to duplicate some of the printer tray functionality. This is discussed in the following section.

Print Queue Management using ITarantellaPrint

This section describes how the `ITarantellaPrint` web service can be used to implement basic print queue management tasks. The following tasks are covered:

- Querying the print queue
- Performing operations on the print queue
- Finding out print environment details

All of the tasks listed accept a `sessionCookie` parameter, where `sessionCookie` is the session cookie of the user session performing the operation. Administrator-level operations require a session cookie for an SGD Administrator user session.

Querying the Print Queue

The following tasks involve querying the print queue to obtain information about print jobs.

Counting the Jobs in a User's Print Queue

A user can use `countJobs` to count the number of jobs in their own print queue, as follows:

```
countJobs(sessionCookie);
```

The number of jobs in the user's print queue is returned in the `total` attribute.

The `countJobs` operation can also be used to check if jobs are present in the user's print queue.

Counting the Jobs in the SGD Print Queue

An SGD Administrator can use adminCount to count the number of jobs in the SGD print queue, as follows:

```
adminCount(sessionCookie,"(objectclass=scottaprintjob)");
```

The number of jobs in the SGD print queue is returned in the `total` attribute. A `sessionCookie` for an SGD Administrator user session must be specified for this operation.

The `admincountJobs` operation can also be used to check if jobs are present in the SGD print queue.

Searching For Print Jobs in a User's Print Queue

A user can search for jobs in their own print queue as follows:

```
search(sessionCookie,"(objectclass=scottaprintjob)",
IConstants.LOCAL_ATTRS);
```

This example lists all print jobs owned by the specified user session. The `IConstants.LOCAL_ATTRS` parameter returns a subset of the available attributes.

Searching For Print Jobs in the SGD Print Queue

The Administrator-level search operations, `adminSearchStart`, `adminSearchNext`, and `adminSearchEnd` enable SGD Administrators to search for print jobs in the SGD print queue. A `sessionCookie` for an SGD Administrator user session must be specified when using these operations.

To initiate a search of the SGD print queue, use `adminSearchStart` as follows:

```
adminSearchStart(sessionCookie,"(objectclass=scottaprintjob)", \
IConstants.LOCAL_ATTRS, 200,"");
```

This example starts a search on all the print jobs in the SGD print queue. The `IConstants.LOCAL_ATTRS` parameter returns a subset of the available attributes.

In this example, the number of results returned is limited to 200. If more results are returned, they are cached on the SGD server. Further "pages" of results for the search can be retrieved using `adminSearchNext`, as follows:

```
adminSearchNext(sessionCookie,searchId,100);
```

This returns the next 100 results for the search identified by `searchId`. The `searchId` is an identifier string returned by the `adminSearchStart` operation.

Before you start a search, you can use the `adminCount` operation to query the number of matching print jobs for a given search.

After finishing a search, if there are still some search results to return, you can use `adminSearchEnd` to clear up the server-side cache and free resources. If an `adminSearchStart` or `adminSearchNext` operation returns a `searchId` string, there are still cached search results on the server.

Looking Up Details for Specific Print Jobs

A user can look up print job details in their own print queue as follows:

```
lookupJobs(sessionCookie,jobids[],IConstants.LOCAL_ATTRS);
```

The `IConstants.LOCAL_ATTRS` parameter returns a subset of the available attributes.

The `jobid` of a print job is returned in the `scottaprintfilename` attribute for the `printjob` object.

Paused Print Jobs on Log Out

If a user logs out while printing is paused on the SGD server, their print jobs in the queue are preserved.

When the user logs in again, their queued print jobs are associated with the new user session and can be queried using the `lookupJobs` or `search` operations.

Performing Operations on the Print Queue

The following tasks involve basic control of the user's print queue.

Printing Out Documents

The following example prints out all print jobs in the user's print queue.

```
printAllJobs(sessionCookie);
```

The following example prints out the specified print jobs in the user's print queue.

```
printJobs(sessionCookie,jobids[]);
```

The `jobid` of a print job is returned in the `scottaprintfilename` attribute for the `printjob` object.

Deleting Print Jobs from a User's Print Queue

The following example deletes all print jobs in the user's print queue.

```
cancelAllJobs(sessionCookie);
```

The following example deletes the specified print jobs in the user's print queue.

```
cancelJobs(sessionCookie,jobids[]);
```

The `jobid` of a print job is returned in the `scottaprintfilename` attribute for the `printjob` object.

Pausing and Resuming the Print Queue

Pausing or resuming the print queue cannot be done using the `ITarantellaPrint` web service. Use `ITarantellaWebtopsession.modifySession` to change the print queue state by modifying an attribute on the user session object, as follows:

```
Item printQueue = new Item();
printQueue.setKey(IAttributeNames.CLIENT_PRINT_QUEUE);
printQueue.setValue(IConstants.PAUSED);
Item[] data = new Item[] { printQueue };
...
sess.modifySession(sessionCookie,data,IConstants.MODE_REPLACEATTR);
```

In this example, `sess` is the user session object to modify.

Obtaining Information About the Printing Environment

The following tasks involve obtaining useful print data.

Getting the Name of the Client Printer

Use the `search` or `adminSearchStart` operation to query the print queue.

The client printer name is returned in the `scottaprintprintername` attribute of the `printjob` object.

Identifying the Print Job ID

Use the `search` or `adminSearchStart` operation to query the print queue.

The print job ID is returned in the `scottaprintfilename` attribute of the `printjob` object.

Identifying the Print Job Owner

Use the `search` or `adminSearchStart` operation to query the print queue.

The full name for the owner of a print job is returned in the `scottaprintusername` attribute of the `printjob` object.

Checking the Print Queue Status

The `ITarantellaPrint` web service does not provide operations to check the print queue status, for example, if the print queue is *paused* or *active*.

You can use the `ITarantellaWeptopSession` operations, `lookupSession` or `adminLookupSession`, to retrieve the `clientprintqueue` attribute for a user session object.

SOAP Reference

This section describes the operations and SOAP message elements for the `ITarantellaPrint` web service.

The following table lists the WSDL file location, namespace information, and end point URL for the `ITarantellaPrint` web service, where *server.example.com* is an SGD server.

WSDL File	`http://`*server.example.com*`/axis/services/document/print?wsdl`
SOAP Operation Namespace	`http://sgd.sun.com/webservices/document/print`
HTTP POST URL	`http://`*server.example.com*`/axis/services/document/print`

Request Message Elements

The following table lists the request message elements for the `ITarantellaPrint` web service.

Element	**Data Type**	**Description**
`desattrs`	`xsd:string`	Array of attributes to be returned in the response. The attributes can be any of the attributes in the `printjob` object. List the individual attributes you want to return or use the `localattributes` shortcut to return all attributes which are available locally. See "`localattributes` Shortcut" on page 59. The attributes are returned in their corresponding `printjob` object elements.
`jobids`	`xsd:string`	Array of print job identifiers. Used by the `cancelJobs`, `lookupJobs`, and `printJobs` operations. Returned in the `scottaprintfilename` attribute of the `printjob` object when you use the `adminSearchStart`, `adminSearchNext`, or `search` operations.
`noofresults`	`xsd:int`	The number of results to return from a search. If `noofresults` is zero or a negative integer, all search results are returned.
`searchId`	`xsd:string`	A unique identifier string for a password cache or token cache search. The value of this element is returned when you use the `adminSearchStart` operation.

Element	Data Type	Description
searchspec	xsd:string	An RFC 2254-compliant LDAP search filter to use for selecting printjob objects. For example, to select all print jobs, use the following search filter: (objectclass=scottaprintjob)
searchtype	xsd:string	Reserved for future use. Use an empty string.
sessioncookie	xsd:string	The session cookie of the user session performing the operation.

localattributes Shortcut

For the ITarantellaPrint web service, the localattributes shortcut returns all of the attributes for a printjob object, apart from the scottaexternaldnsname attribute.

To retrieve the scottaexternaldnsname attribute, you must specify it explicitly by name.

See "printjob Object" on page 60 for details of the printjob object attributes.

Response Message Elements

This section describes the elements returned in response messages for the ITarantellaPrint web service.

<tta:response> Element

The <tta:response> element is the top-level container used for all responses from SGD web services.

An empty response (<tta:response/>) indicates either of the following:

- The operation was successful
- The operation has no data to return

The <tta:response> element can contain <obj name="printjob"> child elements. See "printjob Object" on page 60 for more details about this object.

The `<tta:response>` element can contain the following `<attr>` child elements.

Attribute Name	Description
`searchid`	A unique identifier string for a search. Returned by an `adminSearchStart` or `adminSearchNext` operation. Can be used to identify the search in subsequent `adminSearchNext` and `adminSearchEnd` operations. An empty `<searchid>` element indicates that there are no more results to return.
`total`	The number of print jobs owned by the user session. Returned by the `countJobs` operation.

`printjob` Object

A `printjob` object is designated by the `<obj name="printjob">` element. A `printjob` object contains data about a single print job in the SGD print queue.

One or more `printjob` objects can be returned by the `adminSearchStart`, `adminSearchNext` and `search` operations. Matching SGD print jobs for a search are returned as `printjob` objects. The desired attributes specified in the search are returned as attributes of each `printjob` object.

A `printjob` object can contain the following `<attr>` attribute elements.

Attribute Name	Description
`objectclass`	The LDAP schema object class. For print job objects, this is `scottaprintjob`.
`scottaexternaldnsname`	The external DNS name of the server hosting the user session. An external DNS name is the DNS name that client devices use, for example, `www.example.com`.
`scottapeerdnsname`	The peer DNS name of the SGD server on which the print job is queued. A peer DNS name is the DNS name that SGD servers use to identify themselves and each other, for example, `server.example.com`.
`scottaprintcopies`	The number of copies of the document to be printed.
`scottaprintfilename`	A unique identifier string for the print job.
`scottaprintformat`	The format of the print job, for example, Postscript or PCL (Printer Command Language).
`scottaprintprintername`	The name of the client printer used for this print job.

Attribute Name	Description
`scottaprintname`	The name of the document to be printed.
`scottaprintsize`	The size of the print file, in bytes.
`scottaprintserver`	The DNS name for the print server. This is the DNS name of the application server from which the print job originated.
`scottaprintstatus`	The status of the print job. *This attribute is deprecated*.
`scottaprintt3username`	The user account on the application server that owns the print job.
`scottaprinttfnname`	The fully qualified name for the print file.
`scottaprinttimestamp`	A unique timestamp allocated to a print job when it joins the print queue.
`scottaprintusername`	The user profile object name for the originator of the print job.

The following example shows the typical attributes returned for a `printjob` object.

The following example shows the format of a response message to a `adminSearchStart` operation. The response message returns `printjob` objects and a unique `searchid` attribute that identifies the search.

```
<tta:response>
 <obj name="printjob">
   <attr name="scottaprintt3username">root</attr>
   <attr name="scottaprintprintername">sales02</attr>
   <attr name="scottaprintformat">Unknown</attr>
   <attr name="scottaprinttimestamp">1195211742991</attr>
   <attr name="scottaprintsize">2167</attr>
   <attr name="scottaprintname">sales-forecast-2007.odt</attr>
   <attr name="scottaprintcopies">1</attr>
   <attr name="scottaprintfilename">root-9523</attr>
   <attr name="scottapeerdnsname">server.example.com</attr>
   <attr name="objectclass">scottaprintjob</attr>
   <attr name="scottaprintserver">server</attr>
   <attr name="scottaprintstatus">0</attr>
   <attr name="scottaprinttfnname">.../_dns/server.example.com/_service/sco/
tta/print/root-9523</attr>
   <attr name="scottaprintusername">.../_ens/o=Tarantella System Objects/
cn=Administrator</attr>
 </obj>
 <obj name="printjob">
  <attr name="scottaprintt3username">root</attr>
  <attr name="scottaprintprintername">sales02</attr>
  <attr name="scottaprintformat">Unknown</attr>
  <attr name="scottaprinttimestamp">1195211740849</attr>
  <attr name="scottaprintsize">2167</attr>
  <attr name="scottaprintname">sales-forecast-2006.odt</attr>
  <attr name="scottaprintcopies">1</attr>
  <attr name="scottaprintfilename">root-9439</attr>
  <attr name="scottapeerdnsname">server.example.com</attr>
  <attr name="objectclass">scottaprintjob</attr>
  <attr name="scottaprintserver">server</attr>
  <attr name="scottaprintstatus">0</attr>
  <attr name="scottaprinttfnname">.../_dns/server.example.com/_service/sco/
tta/print/root-9349</attr>
  <attr name="scottaprintusername">.../_ens/o=Tarantella System Objects/
cn=Administrator</attr>
 </obj>
 <attr name="searchId">server.example.com:1195051376977:-3865342581179161190:
PR-1195216694083</attr>
</tta:response>
```

Operations

The following table lists the available operations for the `ITarantellaPrint` web service.

Operation Name	Purpose
`adminCount`	Counts the number of matching print jobs a search would return. For use by SGD Administrators only.
`adminSearchEnd`	Releases server resources for a given search. For use by SGD Administrators only.
`adminSearchNext`	Retrieves the next subset of search results. For use by SGD Administrators only.
`adminSearchStart`	Starts a search of the SGD print queue, returning a subset of results. For use by SGD Administrators only.
`cancelAllJobs`	Cancels all print jobs owned by a user session.
`cancelJobs`	Cancels specified print jobs owned by a user session.
`countJobs`	Counts the print jobs owned by a user session.
`lookupJobs`	Queries the status of specified print jobs owned by a user session.
`printAllJobs`	Prints all print jobs owned by a user session.
`printJobs`	Prints a range of jobs owned by a user session.
`search`	Searches the print queue for print jobs owned by a user session.

adminCount

Counts the number of matching print jobs for a given search filter.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminCount>
 <sessioncookie>xsd:string</sessioncookie>
 <searchspec>xsd:string</searchspec>
</adminCount>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="total">xsd:string</attr>
</tta:response>
```

adminSearchEnd

Used with the `adminSearchStart` and `adminSearchNext` operations to perform searches on the SGD print queue.

The `adminSearchEnd` operation releases server resources for a given search, identified by the `<searchid>` element. Use this operation if you have finished with a search, but not all of the results have been retrieved.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminSearchEnd>
 <sessioncookie>xsd:string</sessioncookie>
 <searchid>xsd:string</searchid>
</adminSearchEnd>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

adminSearchNext

Used with the adminSearchStart and adminSearchEnd operations to perform searches on the SGD print queue.

The adminSearchNext operation returns the next subset of search results for a given search, identified by the <searchid> element. The <noofresults> element defines the maximum number of results to return from the search.

Use the adminSearchEnd if you have finished with a search, but not all of the results have been retrieved.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminSearchNext>
 <sessioncookie>xsd:string</sessioncookie>
 <searchid>xsd:string</searchid>
 <noofresults>xsd:int</noofresults>
</adminSearchNext>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="searchid">xsd:string</attr>
 <obj name="printjob"> [0..unbounded]
</tta:response>
```

adminSearchStart

Used with the adminSearchNext and adminSearchEnd operations to perform searches on the SGD print queue.

The adminSearchStart operation starts a search of the SGD print queue, returning a subset of the results, specified by the <noofresults> element.

If the search returns more results than <noofresults>, the search is cached on the SGD server. Subsequent results can be returned using the adminSearchNext operation.

Use the adminSearchEnd operation if you have finished with a search, but not all of the results have been retrieved.

This operation is for use by SGD Administrators only. For user-level searches, the search operation is provided.

Request Message

The format of the request message for this operation is as follows:

```
<adminSearchStart>
 <sessioncookie>xsd:string</sessioncookie>
 <searchspec>xsd:string</searchspec>
 <desattrs>xsd:string</desattrs> [1..unbounded]
 <noofresults>xsd:int</noofresults>
 <searchtype>xsd:string</searchtype>
</adminSearchStart>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="searchid">xsd:string</attr>
 <obj name="printjob"> [0..unbounded]
</tta:response>
```

cancelAllJobs

Cancels all print jobs owned by the user session specified in the <sessioncookie> element.

Request Message

The format of the request message for this operation is as follows:

```
<cancelAllJobs>
```

```
 <sessioncookie>xsd:string</sessioncookie>
</cancelAllJobs>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

cancelJobs

Cancels a selection of print jobs for the user session specified in the `<sessioncookie>` element.

Request Message

The format of the request message for this operation is as follows:

```
<cancelJobs>
 <sessioncookie>xsd:string</sessioncookie>
 <jobids>xsd:string</jobids> [1..unbounded]
</cancelJobs>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

countJobs

Returns the number of print jobs owned by the user session specified in the `<sessioncookie>` element. To count all SGD print jobs, use the administrator-level opereration, `adminCount`.

Request Message

The format of the request message for this operation is as follows:

```
<countJobs>
 <sessioncookie>xsd:string</sessioncookie>
```

```
</countJobs>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="total">xsd:string</attr>
</tta:response>
```

lookupJobs

Queries the status of specified print jobs for the user session specified in the `<sessioncookie>` element.

Request Message

The format of the request message for this operation is as follows:

```
<lookupJobs>
 <sessioncookie>xsd:string</sessioncookie>
 <jobids>xsd:string</jobids> [1..unbounded]
 <desattrs>xsd:string</desattrs> [1..unbounded]
</lookupJobs>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <obj name="printjob"> [0..unbounded]
</tta:response>
```

printAllJobs

Prints all jobs for the user session specified in the `<sessioncookie>` element.

Request Message

The format of the request message for this operation is as follows:

```
<printAllJobs>
 <sessioncookie>xsd:string</sessioncookie>
</printAllJobs>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

`printJobs`

Prints a range of jobs for the user session specified in the `<sessioncookie>` element.

Request Message

The format of the request message for this operation is as follows:

```
<printJobs>
 <sessioncookie>xsd:string</sessioncookie>
 <jobids>xsd:string</jobids> [1..unbounded]
</printJobs>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

`search`

Searches the print queue owned by the user session specified in the `<sessioncookie>` element. To perform an administrator-level search on all SGD print jobs, use the `adminSearchStart` operation.

Request Message

The format of the request message for this operation is as follows:

```
<search>
```

```
 <sessioncookie>xsd:string</sessioncookie>
 <searchspec>xsd:string</searchspec>
 <desattrs>xsd:string</desattrs> [1..unbounded]
</search>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <obj name="printjob"> [0..unbounded]
</tta:response>
```

CHAPTER 14

The ITarantellaUser Web Service

Pre-release draft. This chapter is not available.

CHAPTER 15

The ITarantellaUtility Web Service

This chapter describes how you can use the `ITarantellaUtility` web service. The available operations and SOAP message elements for this web service are described in detail.

This chapter includes the following topics:

- "Using ITarantellaUtility" on page 73
- "SOAP Reference" on page 78

Using ITarantellaUtility

The `ITarantellaUtility` web service provides supporting operations for the SGD web services. This section describes how you can use the `ITarantellaUtility` web service to do the following:

- Keep a connection to the SGD Web Server open
- Send a batch of SOAP requests
- Search for password cache and token cache entries

Keeping a Connection Open

A client application might require that the network connection to the SGD Web Server be kept open for long periods of time.

You can use the `sendKeepAlive` operation to send a data packet to keep the network connection open as follows:

```
sendKeepAlive(sessionCookie);
```

The sessionCookie parameter identifies the user session performing the operation

Sending a Batch of SOAP Requests

The sendBatchRequest operation groups together several similar operations, and sends them as a single SOAP request. This reduces the number of SOAP transactions sent over the network, as a multiple request-response SOAP conversation is reduced to a single SOAP request-response conversation.

A typical situation where you can use sendBatchRequest is to register multiple event listeners. An event listener is a program component that detects user input or changes to an object. As registration of each event listener uses the same operation, registerClientSideEventListener, the process lends itself to a batch operation. Also, if one event listener registration operation fails this does not affect the other event listeners. See "Using a Batch Request to Register Event Listeners" on page 76.

The sendBatchRequest operation uses the complex data types defined by the following objects:

- A RequestObject object contains a single web service request.
- A ResponseObject object contains a single web service response.

The sendBatchRequest operation accepts an array of RequestObjects and returns an array of ResponseObjects, as follows:

```
ResponseObject[] responses = sendBatchRequest(String sessionCookie,
RequestObject[] requests);
```

RequestObject Object

The RequestObject object contains a single web service request. A RequestObject object includes fields containing the following information:

- The namespace for the web service
- The web service operation to invoke
- Input parameters for the web service operation
- Whether to abort the batch if the request fails

For example, a RequestObject for an ITarantellaPrint.countJobs operation can be coded as follows.

CODE EXAMPLE 15-1 Creating a New RequestObject object

```
RequestObject reqObj = new RequestObject();
reqObj.setUrnName("http://sgd.sun.com/webservices/document/print");
reqObj.setMethodName(IConstantsWSDL.MTHD_COUNTJOBS);
reqObj.setAbortRequest(true);
reqObj.setParameters(new Object[]{sessionCookie});
```

The setAbortRequest method defines the abort behaviour for the batch should the request fail. If set to true, a failed request means that all subsequent requests in the batch are aborted. If false, subsequent requests in the batch are processed as normal.

The default value for setAbortRequest is false.

ResponseObject Object

The ResponseObject object contains a single web service response. A ResponseObject object includes the following fields:

- The XML SOAP response string.
- The fault flag value, indicating whether the request generated a SOAP fault.

You can parse a single ResponseObject from a batch response as follows:

CODE EXAMPLE 15-2 Parsing a Single ResponseObject From a Batch Response

```
// Get batch response
ResponseObject [] responses = util.sendBatchRequest \
(sessionCookie, RequestObject[] requests);
// Get first response object
ResponseObject resp1 = responses[0];
// Get XML SOAP response string
string xml = resp1.getXML();
// Get Fault flag
Boolean flag = resp1.getIsFault();
```

Using a Batch Request to Register Event Listeners

The following code example illustrates how to batch together several client event listener registration operations.

CODE EXAMPLE 15-3 Using a Batch Request to Register Client Event Listeners

```
RequestObject[] requests = new RequestObject[3];
// Listen for webtop state change
requests[0] = new RequestObject();
requests[0].setUrnName("http://sgd.sun.com/webservices/document/event");
requests[0].setMethodName(IConstantsWSDL.MTHD_REGISTERCLIENT);
requests[0].setParameters(new Object[] \
{"",IConstants.WEBTOP_STATE_EVT, stateChangedURL, "WebtopFrame",listenerId});
// Listen for webtop content change
requests[1] = new RequestObject();
requests[1].setUrnName("http://sgd.sun.com/webservices/document/event");
requests[1].setMethodName(IConstantsWSDL.MTHD_REGISTERCLIENT);
requests[1].setParameters(new Object[] \
{"",IConstants.WEBTOP_CHANGED_EVT, stateChangedURL, "WebtopFrame",listenerId});
// Listen for log out of user session
requests[2] = new RequestObject();
requests[2].setUrnName("http://sgd.sun.com/webservices/document/event");
requests[2].setMethodName(IConstantsWSDL.MTHD_REGISTERCLIENT);
requests[2].setParameters(new Object[] \
{"",IConstants.LOGGED_OUT_EVT,loggedOutURL, "_top", listenerId});
// Send the batch request
ResponseObject[] responses = util.sendBatchRequest(sessionCookie, requests);
// Parse the responses
for (int i=0;i<responses.length;i++) {
ResponseObject rspobj = responses[i];
if (!rspobj.getIsFault()) {
    String batchresp = rspobj.getXML();
        ...
    } else {
    System.out.println("Batch response "+i+": FAILED\n");
    }
}
```

All of the requests in the batch use the `requestClientSideEventListener` operation of the `ITarantellaEvent` web service.

See Chapter 10 for more details about registering event listeners.

Searching for Password or Token Cache Entries

The `ITarantellaUtility` web service provides the following operations to search for entries in the password cache or token cache:

- `adminSearchStart`
- `adminSearchNext`
- `adminSearchEnd`

Note – These operations are for SGD Administrators only. A session cookie for an SGD Administrator user session must be specified when using these operations.

Use the `adminSearchStart` operation to commence a search for cache entries.

The following example searches through all entries in the password cache, returning all attributes for all matches:

```
adminSearchStart(sessionCookie,.../_service/sco/tta/passcache, \
"",(objectclass=*),onelevel,"",0,"");
```

The following example searches through all entries in the token cache, returning the `IConstantsXML.DATASTOREITEMNAME` attribute for the first 50 matches:

```
adminSearchStart(sessionCookie, .../_service/sco/tta/tokencache, \
new String[]{IConstantsXML.NAME_DATASTOREITEMNAME},(objectclass=*),
onelevel,"",50,"");
```

The `IConstantsXML.DATASTOREITEMNAME` attribute identifies the user profile that owns the token cache entry.

If more than 50 matches are found for this search, the `adminSearchStart` operation returns a `searchId` attribute and caches the remaining results on the SGD server. The `searchId` attribute is a unique identifier for the search.

You can use the `searchId` attribute with the `adminSearchNext` operation to return subsequent “pages” of results for the search. The following example returns the next 100 results for the search identified by `searchId`.

```
adminSearchNext(sessionCookie, searchId, 100);
```

If no further results are available for this search, the `adminSearchNext` operation returns an empty `searchId` attribute in the response.

If you have finished with a search but have not returned all the results, use the `adminSearchEnd` operation as follows to release server resources:

```
adminSearchEnd(sessionCookie, searchId);
```

SOAP Reference

This section describes the operations and SOAP message elements for the `ITarantellaUtility` web service.

The following table lists the WSDL file location, namespace information, and end point URL for the `ITarantellaUtility` web service, where *server.example.com* is an SGD server.

WSDL File	`http://`*server.example.com*`/axis/services/document/utility?wsdl`
SOAP Operation Namespace	`http://sgd.sun.com/webservices/document/utility`
HTTP POST URL	`http://`*server.example.com*`/axis/services/document/utility`

Request Message Elements

The following table lists the request message elements for the `ITarantellaUtility` web service.

Element	Data Type	Description
`abortRequest`	`xsd:boolean`	Defines the action taken if a request in a `sendBatchRequest` operation results in a SOAP fault, as follows: • `true` – If the request fails, subsequent requests in the batch are abandoned. • `false` – If the request fails, subsequent requests in the batch are processed.
`depth`	`xsd:string`	Defines the depth of sub-tree to search, as follows: • `onelevel` – Searches one level under the `rootname`. • `subtree` – Searches the whole sub-tree under `rootname`.
`desattrs`	`xsd:string`	Array of attributes to be returned in the response. The attributes can be any of the attributes in the `datastoreitem` object. List the individual attributes you want to return or use the `localattributes` shortcut to return all attributes which are available locally. See "`localattributes` Shortcut" on page 79. The attributes are returned in their corresponding `datastoreitem` object elements.
`methodName`	`xsd:string`	The name of the web service operation for a request in a `sendBatchRequest` operation.

Element	Data Type	Description
namefilter	xsd:string	A regular expression to filter on object names.
noofresults	xsd:int	The number of results to return from a search. If noofresults is zero or a negative integer, all search results are returned.
parameters	xsd:anyType	Parameters for a single request in a sendBatchRequest operation. An array of item objects, where an item object is a single parameter for a request, having a data type of xsd:anyType.
requests	tns1:Req	A SOAP request object, used in a sendBatchRequest operation. A Req request object is a complex data type that defines a single web service operation. A Req object consists of the following fields: • abortRequest • methodName • parameters • urnName See "Req Request Object" on page 80.
searchId	xsd:string	A unique identifier string for a password cache or token cache search. The value of this element is returned when you use the adminSearchStart operation.
searchspec	xsd:string	An RFC 2254-compliant LDAP search filter to use for selecting objects.
searchtype	xsd:string	Reserved for future use. Use an empty string.
sessioncookie	xsd:string	The session cookie of the user session performing the operation.
rootname	xsd:string	Defines a root object to start a search from, as follows: • .../_service/sco/tta/passcache – Searches the password cache. • .../_service/sco/tta/tokencache – Searches the token cache.
urnname	xsd:string	Namespace for a web service operation. Used in the sendBatchRequest operation.

localattributes Shortcut

For the ITarantellaUtility web service, the localattributes shortcut returns the following attributes for a datastoreitem object:

- objectclass
- datastoreitemname

To retrieve other attributes, you must specify them explicitly by name.

See "datastoreitem Object" on page 81 for details of the datastoreitem object attributes.

Req Request Object

The sendBatchRequest operation sends an array of Req request objects. A Req request object is a complex data type that defines a single web service request as follows:

```
<Req>
 <abortRequest>xsd:boolean</abortRequest>
 <methodName>xsd:string</methodName>
 <parameters>
  <item>xsd:anyType</item> [0..unbounded]
 </parameters>
 <urnName>xsd:string</urnName>
</Req>
```

Response Message Elements

This section describes the elements returned in response messages for the ITarantellaUtility web service.

<tta:response> Element

The <tta:response> element is the top-level container used for all responses from SGD web services.

For the sendBatchRequest operation, each operation of the batch request returns a single <tta:response> element wrapped by the <XML> element.

An empty response (<tta:response/>) indicates either of the following:

- The operation was successful
- The operation has no data to return

The <tta:response> element can contain the following <obj> child elements.

Object Name	Description
datastoreitem	An object representing an entry in the password cache or token cache. One or more datastoreitem objects can be returned by an adminSearchStart or adminSearchNext operation. See "datastoreitem Object" on page 81.
Resp	An object containing SOAP response data for a single operation request in a batch response. Returned by a sendBatchRequest operation. See "Resp Response Object" on page 82.

The `<tta:response>` element can contain `<attr name="searchid">` child elements. This attribute is a unique identifier for a token cache or password cache search and is returned by an `adminSearchStart` or `adminSearchNext` operation.

`datastoreitem` Object

A `datastoreitem` object is designated by the `<obj name="datastoreitem">` element. A `datastoreitem` object represents an entry in the password cache or token cache.

A `datastoreitem` object can contain the following `<attr>` elements.

Attribute Name	**Description**
`datastoreitemname`	Fully Qualified Name of the user profile owning the cache entry.
`filemodifiedtime`	When the cache entry was last modified.
`filename`	Unique filename for the cache entry.
`filesize`	Size of the cache file, in bytes.
`fileaccessedtime`	When the cache entry was last accessed.
`objectclass`	Parent object class for the cache entry.
`filepath`	Location of the cache file on the SGD server.
`filecreationtime`	When the cache file was created.

The following example shows the format of a response message to an `adminSearchStart` operation. The response message returns `datastoreitem` objects and a unique `searchid` attribute that identifies the search.

```
<tta:response>
 <obj name="datastoreitem">
  <attr name="datastoreitemname">".../_ens/o=Tarantella System Objects/cn=
Administrator"</attr>
  <attr name="filemodifiedtime">1194878178000</attr>
  <attr name="filename">77eRLi7vt5BfZW5z77eQbz1UYXJhbnRlbGxhIFN5c3RlbSBPY
mplY3Rz77eQY249QWRtaW5pc3RyYXRvcg==</attr>
  <attr name="filename">77eRLi7vt5BfZW5z77eQbz1...QWRtaW5pc3RyYXRvcg==</attr>
  <attr name="filesize">188</attr>
  <attr name="fileaccessedtime">1195222073000</attr>
  <attr name="objectclass">top</attr>
  <attr name="objectclass">file</attr>
  <attr name="filepath">/opt/tarantella/var/password/tier3cache/77eRLi7vt5BfZ
W5z77eQbz1UYXJhbnRlbGxhIFN5c3RlbSBPYmplY3Rz77eQY249QWRtaW5pc3RyYXRvcg==</attr>
  <attr name="filecreationtime">1194878178000</attr>
 </obj>
```

```
  <obj name="datastoreitem">
   <attr name="datastoreitemname">".../_user/barrie"</attr>
   <attr name="filemodifiedtime">1195052886000</attr>
   <attr name="filename">77eRLi7vt5BfdXNlcu+3kGJhcnJpZQ==</attr>
   <attr name="filesize">188</attr>
   <attr name="fileaccessedtime">1195052886000</attr>
   <attr name="objectclass">top</attr>
   <attr name="objectclass">file</attr>
   <attr name="filepath">/opt/tarantella/var/password/tier3cache/77eRLi7vt5Bfd
XNlcu+3kGJhcnJpZQ==</attr>
   <attr name="filecreationtime">1195052886000</attr>
   </obj>
   <attr name="searchId">server.example.com:1195222072758:4638226410762143120:
DS-1195224229797</attr>
</tta:response>
```

`Resp` Response Object

The `sendBatchRequest` operation returns an array of `Resp` response objects. A single `Resp` response object is returned for each operation in the batch request.

The `Resp` response object is a complex data type designated by a `<type="Resp">` attribute. The `Resp` response object contains SOAP response data for a single request from the batch request. This object can contain the following elements.

Element Name	Description
`XML`	The response XML string for the operation request.
`isFault`	A Boolean flag, indicating whether the request succeeded or returned a SOAP fault, as follows: • `true` – The request generated a SOAP fault. • `false` – The request did not generate a SOAP fault.

The response message format for a `Resp` reponse object is as follows:

```
<Resp>
 <XML>xsd:string</XML>
 <isFault>xsd:boolean</isFault>
</Resp>
```

The `<XML>` section wraps the `<tta:response>` top level container for the response. The returned `<XML>` section for a `Resp` response object varies, depending on the web service request.

A batch request containing three request operations returns the same number of Resp response objects, as shown in the following example.

```
<XML>
 <tta:response>
    ...
 </tta:response>
</XML>
<isFault>false</isFault>
...
<XML>
 <tta:response>
    ...
 </tta:response>
</XML>
<isFault>false</isFault>

<XML>
 <tta:response>
...
 </tta:response>
</XML>
<isFault>false</isFault>
```

The attributes returned in the <tta:response> elements depend on the operations you specify for the batch request.

Operations

The following table lists the available operations for the ITarantellaUtility web service.

Operation Name	Purpose
adminSearchEnd	Releases server-side resources for a given search. For use by SGD Administrators only.
adminSearchNext	Retrieves the next subset of search results. For use by SGD Administrators only.
adminSearchStart	Starts a search of the password cache or token cache, returning a subset of results. For use by SGD Administrators only.
sendBatchRequest	Sends a batch of web service requests.
sendKeepAlive	Sends a keep-alive packet over the network connection.

adminSearchEnd

Used with the adminSearchStart and adminSearchNext operations to perform searches on the password cache or token cache.

The adminSearchEnd operation releases server resources for a given search, identified by the <searchid> element. Use this operation if you have finished with a search, but not all of the results have been retrieved.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminSearchEnd>
 <sessioncookie>xsd:string</sessioncookie>
 <searchid>xsd:string</searchid>
</adminSearchEnd>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

adminSearchNext

Used with the adminSearchStart and adminSearchEnd operations to perform searches on the password cache or token cache.

The adminSearchNext operation returns the next subset of search results for a given search, identified by the <searchid> element. The <noofresults> element defines the maximum number of results to return from the search.

Use the adminSearchEnd operation if you have finished with a search, but not all of the results have been retrieved. Doing this releases server resources.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminSearchNext>
 <sessioncookie>xsd:string</sessioncookie>
 <searchid>xsd:string</searchid>
 <noofresults>xsd:int</noofresults>
</adminSearchNext>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="searchId">xsd:string</attr>
 <obj name="datastoreitem"> [0..unbounded]
</tta:response>
```

adminSearchStart

Used with the adminSearchNext and adminSearchEnd operations to perform searches on the password cache or token cache.

The adminSearchStart operation starts a search of the password cache or token cache, returning a subset of the results, specified by the <noofresults> element.

If the search returns more results than <noofresults>, the search is cached on the SGD server. Subsequent results can be returned using the adminSearchNext operation.

The adminSearchStart operation returns a unique identifier for the search, searchId. This can be used in subsequent adminSearchNext operations.

Use the adminSearchEnd operation if you have finished with a search, but not all of the results have been retrieved. Doing this releases server resources.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminSearchStart>
 <sessioncookie>xsd:string</sessioncookie>
 <rootname>xsd:string</rootname>
 <desattrs>xsd:string</desattrs> [1..unbounded]
 <searchspec>xsd:string</searchspec>
 <depth>xsd:string</depth>
 <namefilter>xsd:string</namefilter>
 <noofresults>xsd:int</noofresults>
 <searchtype>xsd:string</searchtype>
</adminSearchStart>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <obj name="datastoreitem"> [0..unbounded]
 <attr name="searchId">xsd:string</attr>
</tta:response>
```

sendBatchRequest

Sends a batch of web service requests. Use this operation to optimize network traffic by reducing the number of requests and responses.

Request Message

The format of the request message for this operation is as follows:

```
<sendBatchRequest>
 <sessioncookie>xsd:string</sessioncookie>
 <requests>tns1:Req</requests> [1..unbounded]
</sendBatchRequest>
```

The request message sends an array of `Req` request objects. A `Req` object is a complex data type that defines a single web service request. See "`Req` Request Object" on page 80 for more details.

Response Message

The format of the response message for this operation is as follows:

```
<responses>tns1:Resp</responses> [0..unbounded]
```

The response message returns an array of `Resp` response objects. A `Resp` object is a complex data type that defines a single web service response. See "`Resp` Response Object" on page 82 for more details.

sendKeepAlive

Sends a keep-alive packet over the network connection.

Request Message

The format of the request message for this operation is as follows:

```
<sendKeepAlive>
 <sessioncookie>xsd:string</sessioncookie>
</sendKeepAlive>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

CHAPTER **16**

The ITarantellaWebtopContent Web Service

This chapter describes how you can use the `ITarantellaWebtopContent` web service to build an SGD webtop. The available operations and SOAP message elements for this web service are described in detail.

This chapter includes the following topics:

Using ITarantellaWebtopContent

An SGD Administrator assigns applications to a user, using the Administration Console or the command line.

The `ITarantellaWebtopContent` web service defines *user-level* operations that enable an SGD user to query their assigned applications. These operations are typically used when building a webtop for an SGD user.

`ITarantellaWebtopContent` can be used to do the following:

- Look up the attributes for a user's assigned applications
- Search through a user's assigned applications

For *Administrator-level* operations, where an SGD Administrator is able to create and delete SGD objects and configure the assigned applications for other SGD users, use the following SGD web services:

- Use `ITarantellaDatastore` to create, delete, or edit objects in the SGD datastore. See Chapter 9 for more information.

- Use `ITarantellaAdmin.runCommand` to run a `tarantella` command to edit and manipulate SGD objects, and define object links. See Chapter 8 for more information.

The `launchpad` sample application, included in the `examples/apis` folder of the SGD distribution, is an application that builds a user webtop and provides webtop links to start applications. The `launchpad` application code provides an example of how a user webtop can be generated using `ITarantellaWebtopContent`, together with related classes from the `com.tarantella.tta.webservices.*` package.

This section describes how to use `ITarantellaWebtopContent` to build a user webtop. It includes code snippets from the `launchpad` sample application, along with explanatory text.

About Webtops

A *webtop* is a web page that lists the applications assigned to a user. An SGD user can start applications and view documents from their webtop. The list of assigned applications is sometimes called webtop content.

This section describes the following webtops included with the SGD distribution:

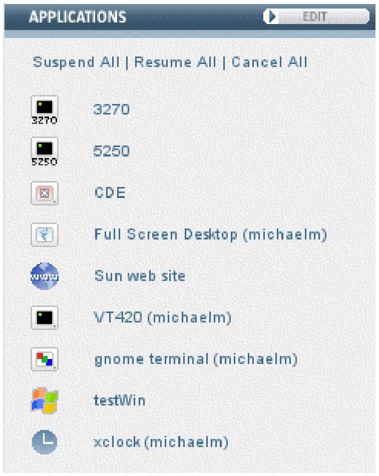
- The default SGD webtop
- The hierarchical webtop

The Default SGD Webtop

To display the default SGD webtop, go to `http://`*server.example.com*`/sgd`, where *server.example.com* is the name of an SGD server. Alternatively, log in to SGD using the SGD Web Server Welcome Page at http://*server.example.com*.

On the default SGD webtop, the assigned applications for a user are shown as a list of webtop links with associated icons.

FIGURE 16-1 Default SGD Webtop, Showing the Assigned Applications for a User



Suspend All | Resume All | Cancel All

3270
5250
CDE
Full Screen Desktop (michaelm)
Sun web site
VT420 (michaelm)
gnome terminal (michaelm)
testWin
xclock (michaelm)

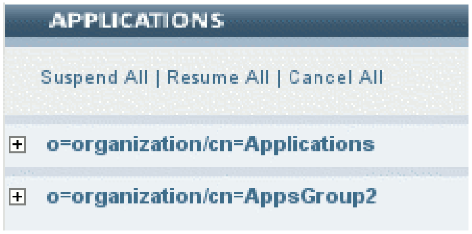
To start an application, a user clicks the webtop link.

Hierarchical Webtop

The hierarchical webtop is a webtop that lists a user's assigned applications according to the groups that the applications and documents belong to.

To display the hierarchical webtop, go to `http://`*server.example.com*`/sgd/hierarchy.jsp`, where *server.example.com* is the name of an SGD server.

FIGURE 16-2 Hierarchical Webtop



Suspend All | Resume All | Cancel All

o=organization/cn=Applications

o=organization/cn=AppsGroup2

`ITarantellaWebtopContent` includes a `searchWebtopContentHierarchy` operation. This operation preserves any existing group hierarchy and returns objects arranged by `webtopgroup`.

By comparison, the searchWebtopContent operation strips out all group information and returns a "flat" XML response.

Building a Webtop

This section describes how you can use ITarantellaWebtopContent to build a webtop for an SGD user.

Building a webtop for a user involves the following steps:

1. Query the assigned applications for the user
2. Look up useful attributes for building a webtop
3. Construct webtop links used to start applications

These steps are described in detail in the following sections.

Querying the Assigned Applications for a User

ITarantellaWebtopContent provides the following operations to query the assigned applications for a user:

- searchWebtopContent searches a user's applications. Group information *is not included* in the response.
- searchWebtopContentHierarchy searches a user's applications. Group information is included in the response. See "Hierarchical Webtop" on page 91.
- lookupWebtopItems returns the value of specific attributes for a user's applications.

The following example uses searchWebtopContent to retrieve all assigned applications for the owner of the user session designated by sessionCookie. The shortcut localattributes returns a subset of attributes for the matching application objects.

CODE EXAMPLE 16-1 Querying the Assigned Applications for a User

```
ITarantellaWebtopContent webtopContent = locator.getWebtopContent();
    ...
String xmlstring = webtopContent.searchWebtopContent(sessionCookie, \
"(objectclass=*)",new String[] {IConstants.LOCAL_ATTRS});
```

The XML SOAP response string, `xmlstring`, returned from the `searchWebtopContent` operation can be parsed to extract object attribute values. CODE EXAMPLE 16-2 uses the `parser.IResponse` and `parser.ResponseParser` classes to parse the response string. The code iterates through the returned objects and returns some attribute values.

CODE EXAMPLE 16-2 Parsing the XML Response From a `searchWebtopContent` Operation

```
IResponse resp = parser.parse(xmlstring);
Enumeration enm = resp.getChildren();
...
while (enm.hasMoreElements())
{
IResponse elem = (IResponse) enm.nextElement();
name = elem.getAttribute(ISchemaNames.WEBTOP_ITEM_NAME)[0];
displayName = elem.getAttribute(ISchemaNames.COMMON_NAME)[0];
placement = elem.getAttribute(ISchemaNames.APPLICATION_PLACEMENT)[0];
running = elem.getAttribute(ISchemaNames.RUNNING_INSTANCES)[0];
maxAllowed = elem.getAttribute(ISchemaNames.MAX_INSTANCES)[0];
    ...
}
```

If you know the full name of an application or document object, returned in the Name attribute (`scottawebtopobjectname`) for the object, you can use the `lookupWebtopItems` operation to query the value of one or more attributes for the object. The following example returns the value of the Maximum Instances (`scottamaxinstances`) and Window Type (`scottaapplicationplacement`) attributes for the specified application object.

CODE EXAMPLE 16-3 Looking Up an Attribute Value Using `lookupWebtopItems`

```
ITarantellaWebtopContent webtopContent=locator.getWebtopContent();
    ...
String xmlstring = webtopContent.lookupWebtopItems(sessionCookie, \
new String[] {".../_ens/o=applications/cn=dtterm"},               \
new String[] {ISchemaNames.MAX_INSTANCES,ISchemaNames.APPLICATION_PLACEMENT});
```

The XML response string, `xmlstring`, returned from the `lookupWebtopItems` operation can be parsed to extract object attribute values. See CODE EXAMPLE 16-2.

Many possible attributes for SGD objects can be returned using the `searchWebtopContent`, `searchWebtopContentHierarchy`, and `lookupWebtopItems` operations. Some of the key attributes you must consider when building a webtop and constructing webtop links to start applications are discussed in "Useful Attributes for Building Webtops" on page 94.

Useful Attributes for Building Webtops

Attribute values for an application object or document object can be queried using `ITarantellaWebtopContent` operations such as `searchWebtopContent` or `lookupWebtopItems`.

However, using these operations can return *all* of the attributes defined for the object. Not all attributes for an object are needed when building a webtop.

Some of the most useful attributes to consider when building a webtop and creating webtop links are as follows:

- Name
- Icon
- Webtop Item Name
- Application Type
- Window Type
- Client's Maximum Size
- New Browser Window
- URL
- Running Instances
- Maximum Instances
- Application Resumability
- Hints

You can retrieve these attributes by using the `localattributes` shortcut in a `searchWebtopContent` or `lookupWebtopItems` operation.

The following sections describe how you can use these attributes to build a webtop.

Name

The Name attribute, `cn`, defines a short descriptive name for a webtop object, for example:

```
<attr name="cn">xclock</attr>
```

On the default SGD webtop, the name for the webtop object is listed next to an icon representing the object. A webtop link for starting the application or document becomes active when you mouse over the link.

Icon

The Icon attribute, `scottaicon`, defines an icon for the application or document, for example:

```
<attr name="scottaicon">clock.gif</attr>
```

The `scottaicon` attribute is the name of a graphics file in `gif` format. The default SGD webtop uses icon graphic files located in the `/`*install-dir*`/webserver/tomcat/`*version*`/webapps/sgd/resources/images/icons` folder in the SGD installation directory.

The icon is displayed on the default SGD webtop, adjacent to the Name for the application or document.

Webtop Item Name

The Webtop Item Name attribute, `scottawebtopobjectname`, is the fully-qualified name for an object. This name identifies the object in the SGD datastore, for example:

```
<attr name="scottawebtopobjectname">.../_ens/o=applications/cn=
xclock</attr>
```

To start an application, you can pass the Webtop Item Name attribute to your launcher code as follows:

```
// Get the fully-qualified name for a webtop object
appname = elem.getAttribute(ISchemaNames.WEBTOP_ITEM_NAME)[0];
...
// Start an application session
ITarantellaEmulatorSession.startSession \
(sessionCookie, appname, Item[] launchOptions);
```

See Chapter 10 for details of how to start an application session using the SGD web services.

Application Type

The Application Type attribute, `scottawebtopobjectclass`, defines the type of the application or document.

For example, the following Application Type attribute defines an X application:

```
<attr name="scottawebtopobjectclass">scottaxapplication</attr>
```

Webtop objects inherit attributes from the corresponding top-level object classes defined in the SGD LDAP schema. Webtop objects belong to one of the following object classes, corresponding to an application or document type.

Description	**LDAP Schema Object Class**
3270 application	`scotta3270application`
5250 application	`scotta5250application`
Character application	`scottacharacterapplication`
Document	`scottahtmldocument`
Windows application	`scottawindowsapplication`
X application	`scottaxapplication`

To determine if an object is an application or document, query the Application Type attribute as follows:

```
boolean isDocument(IResponse object)
//Returns true if the object is a document
{
return ISchemaNames.DOCUMENT.equals  \
(object.getAttribute(ISchemaNames.APP_TYPE)[0]);
}
```

To determine if an object is a Microsoft Windows applications, query the Application Type attribute as follows:

```
boolean isWindowsApp(IResponse object)
//Returns true if the object is a windows application
{
return ("scottawindowsapplication").equals  \
(object.getAttribute(ISchemaNames.APP_TYPE)[0]);
}
```

Window Type

The Window Type attribute, `scottaapplicationplacement`, determines the default window type used to display the application. The following example displays an application object in kiosk mode.

```
<attr name="scottaapplicationplacement">kiosk</attr>
```

On the default SGD webtop, the Window Type is indicated as pop-up text when you mouse over the application name.

Applications can be started without prior knowledge of this attribute.

Client's Maximum Size

The Client's Maximum Size attribute, `scottafullscreen`, determines if the application fills the whole of the client device screen area when started. The following exanple displays an application object at maximum size on the client device.

```
<attr name="scottaafullscreen">1</attr>
```

New Browser Window

The New Browser Window attribute, `scottaoutofplace`, is applicable for document objects only. It specifies that a new browser window be opened to display the document, rather than displaying the document in the webtop browser window.

URL

The URL attribute, `scottalocation`, is applicable for document objects only. It defines the URL of the web page that is displayed when a user clicks the webtop link for the document. Use this attribute to display an HTML document as follows:

```
String docURL = (String)resp.getAttribute(ISchemaNames.LOCATION)[0];
if (docURL != null) {
    if (docURL.startsWith("http") || docURL.startsWith("/"))
    response.sendRedirect(response.encodeRedirectURL(docURL));
    }
```

The following example shows typical object attributes for an HTML document object. Because the `scottaoutofplace` attribute is set in this example, the document is displayed in a new browser window.

```
<attr name="scottalocation">http://www.sun.com</attr>
<attr name="scottawebtopobjectclass">scottahtmldocument</attr>
<attr name="scottaoutofplace">1</attr>
```

Maximum Instances

The Maximum Instances attribute, `scottamaxinstances`, defines the maximum number of instances of an application that a user can run simultaneously. In the following example, a maximum of three simultaneous instances of the `xclock` application can be run.

```
<attr name="cn">xclock</attr>
...
<attr name="scottamaxinstances">3</attr>
```

On the default SGD webtop, the maximum instances value is displayed as pop up text when you mouse over the webtop link.

Running Instances

The Running Instances attribute, `scottarunninginstances`, returns the number of instances of an application that are currently running. In the following example, no instances of the `xclock` application are currently active.

```
<attr name="cn">xclock</attr>
...
<attr name="scottarunninginstances">0</attr>
```

In the following example, the Running Instances attribute is compared to the Maximum Instances attribute, to determine whether to enable or disable an application launch link.

```
// Get number of running instances
running = elem.getAttribute(ISchemaNames.RUNNING_INSTANCES)[0];
// Get maximum number of instances
maxAllowed = elem.getAttribute(ISchemaNames.MAX_INSTANCES)[0];
if (maxAllowed != null && running != null && \
running.equals(maxAllowed) && !running.equals("0"))
   {
        // Do not start a new instance of this object
    }
```

On the default SGD webtop, the value of the Running Instances attribute is displayed in parentheses following the application's name.

Application Resumability

The Application Resumability attribute, `scottasuspend`, defines the resumability behaviour for an application. In the following example, the application is resumeable for 30 minutes after logging out from SGD.

```
<attr name="scottasuspend">forever</attr>
<attr name="scottaresumetimeout">30</attr>
```

On the default SGD webtop, resumability behaviour for an application object is indicated in pop up text when you mouse over the webtop link.

Hints

The Hints attribute, `scottahints`, is used to define one or more custom attributes that can be used to provide finer control over the publishing and display of objects on a webtop.

Webtop hints consist of name-value pairs that a developer can specify. For example, the following webtop hint might be used to specify the size of the icon for an application.

```
<attr name="scottahints">preferredsize=16</attr>
```

Constructing Webtop Links

Applications and documents on an SGD webtop are started using URL links called *webtop links*. Clicking a webtop link on an SGD webtop generates an HTTP GET request string, complete with application startup parameters. The request string is passed to launcher code that handles application session startup and the process of launching the application.

The launchpad sample application includes code that constructs webtop links. A simplified version of this code is included in CODE EXAMPLE 16-4.

CODE EXAMPLE 16-4 Constructing a Simple Webtop Link

```
name = elem.getAttribute(ISchemaNames.WEBTOP_ITEM_NAME)[0];
displayName = elem.getAttribute(ISchemaNames.COMMON_NAME)[0];
placement = elem.getAttribute(ISchemaNames.APPLICATION_PLACEMENT)[0];
    ...
// Create webtop links
workBuffer.setLength(0);
workBuffer.append("<A target='_blank' HREF='");
urlBuffer.setLength(0);
urlBuffer.append(getMyUrl(request));
urlBuffer.append("?action=launch&app=");
urlBuffer.append(urlEncode(name));
urlBuffer.append("&name=");
urlBuffer.append(urlEncode(displayName));
workBuffer.append(response.encodeURL(urlBuffer.toString()));
workBuffer.append("'>");
workBuffer.append(displayName);
workBuffer.append("</A>");
    ...
```

In the launchpad sample application, the action=launch URL parameter is used to initiate some application startup code. Values for the Webtop Item Name (scottawebtopobjectname), Window Type (scottaapplicationplacement), and Name (cn) attributes are included in the GET request string. Other webtop object attribute information can also be added to the GET request string, depending on the requirements for the application.

For a basic implementation, supply the Webtop Item Name for the application to launch and its Window Type attribute in the GET request string.

Starting an Application

The ITarantellaEmulatorSession.startSession method is used to start applications. This method has the following format.

```
startSession(sessionCookie, applicationName, Item[] launchOptions);
```

The launchOptions parameter is an optional array of overrides for the application launch.

More details about using ITarantellaEmulatorSession to configure and start application sessions are included in Chapter 10.

SOAP Reference

This section describes the operations and SOAP message elements for the `ITarantellaWebtopContent` web service.

The following table lists the WSDL file location, namespace information, and end point URL for the `ITarantellaWebtopContent` web service, where *server.example.com* is an SGD server.

WSDL File	`http://`*server.example.com*`/axis/services/document/webtopcontent?wsdl`
SOAP Operation Namespace	`http://sgd.sun.com/webservices/document/webtopcontent`
HTTP POST URL	`http://`*server.example.com*`/axis/services/document/webtopcontent`

Request Message Elements

The following table lists the request message elements for the `ITarantellaWebtopContent` web service.

Element	Data Type	Description
`desattrs`	`xsd:string`	Array of attributes to be returned in the response. The attributes can be any of the attributes in the `webtopitem` object. List the individual attributes you want to return or use the `localattributes` shortcut to return all attributes which are available locally. See "`IConstants.LOCAL_ATTRS` Shortcut" on page 102. The attributes are returned in their corresponding `webtopitem` object elements.
`groups`	`xsd:string`	Array of groups to be searched. list the individual groups you want to search. Use the `top` shortcut to return the top-level groups. See "`IConstants.TOP` Shortcut" on page 105.

Element	Data Type	Description
objectnames	xsd:string	Array of application or document objects to be looked up. The object name for an application or document is specified in the scottawebtopobjectname attribute of the webtopitem object.
searchspec	xsd:string	An RFC2254-compliant LDAP search filter to use for selecting applications or documents. For example, to select all X application objects, use the following search filter: (objectclass=scottaxapplication).
sessioncookie	xsd:string	The session cookie of the user session performing the operation.

IConstants.LOCAL_ATTRS Shortcut

For the ITarantellaWebtopContent web service, the IConstants.LOCAL_ATTRS shortcut returns a subset of the available attributes for a webtopitem object. Using this shortcut returns many useful attributes for building a webtop. See "Building a Webtop" on page 92.

The returned attributes for the IConstants.LOCAL_ATTRS shortcut vary, depending on the application type.

Using the IConstants.LOCAL_ATTRS shortcut for an *X application* returns the following webtopitem object attributes:

- cn
- objectclass
- scottaappletheight
- scottappletwidth
- scottaapplicationplacement
- scottadisplayenginepage
- scottafilepath
- scottafullscreen
- scottaicon
- scottamaxinstances
- scottarunninginstances
- scottascalable
- scottasuspend
- scottawebtopobjectclass

- `scottawebtopobjectname`

Using the `IConstants.LOCAL_ATTRS` shortcut for a *Windows application* returns the following `webtopitem` object attributes:

- `cn`
- `objectclass`
- `scottaappletheight`
- `scottappletwidth`
- `scottaapplicationplacement`
- `scottadisplayenginepage`
- `scottafullscreen`
- `scottaicon`
- `scottamaxinstances`
- `scottarunninginstances`
- `scottascalable`
- `scottasuspend`
- `scottatrylocalwindowsapplication`
- `scottawebtopobjectclass`
- `scottawebtopobjectname`
- `scottawindowsapplicationserver`

Using the `IConstants.LOCAL_ATTRS` shortcut for a *character application* returns the following `webtopitem` object attributes:

- `cn`
- `objectclass`
- `scottaappletheight`
- `scottappletwidth`
- `scottaapplicationplacement`
- `scottadisplayenginepage`
- `scottafilepath`
- `scottafullscreen`
- `scottaicon`
- `scottamaxinstances`
- `scottarunninginstances`
- `scottasuspend`
- `scottawebtopobjectclass`
- `scottawebtopobjectname`

Using the `IConstants.LOCAL_ATTRS` shortcut for a *3270 application* returns the following `webtopitem` object attributes:

- `cn`
- `objectclass`
- `scottaappletheight`
- `scottappletwidth`
- `scottaapplicationplacement`
- `scottadisplayenginepage`
- `scottafilepath`
- `scottafullscreen`
- `scottaicon`
- `scottamaxinstances`
- `scottarunninginstances`
- `scottascalable`
- `scottasuspend`
- `scottawebtopobjectclass`
- `scottawebtopobjectname`

Using the `IConstants.LOCAL_ATTRS` shortcut for a *5250 application* returns the following `webtopitem` object attributes:

- `cn`
- `objectclass`
- `scottaappletheight`
- `scottappletwidth`
- `scottaapplicationplacement`
- `scottadisplayenginepage`
- `scottafilepath`
- `scottafullscreen`
- `scottaicon`
- `scottamaxinstances`
- `scottarunninginstances`
- `scottascalable`
- `scottasuspend`
- `scottawebtopobjectclass`
- `scottawebtopobjectname`

Using the `IConstants.LOCAL_ATTRS` shortcut for a *document* returns the following `webtopitem` object attributes:

- `cn`
- `objectclass`
- `scottaicon`
- `scottalocation`
- `scottaoutofplace`
- `scottarunninginstances`
- `scottawebtopobjectclass`
- `scottawebtopobjectname`

`IConstants.TOP` Shortcut

The `IConstants.TOP` shortcut is used for the `searchWebtopContentHierarchy` operation. This shortcut returns the names of the top-level groups. The name of a webtop group is given by the `groupname` attribute of the `webtopgroup` object.

You can use this shortcut if you need to identify the names of the webtop groups for a user.

Response Message Elements

This section describes the elements returned in response messages for the `ITarantellaWebtopContent` web service.

`<tta:response>` Element

The `<tta:response>` element is the top-level container used for all responses from SGD web services.

An empty response (`<tta:response/>`) indicates that the operation has no data to return. For example, if `lookupWebtopItems` does not find a match.

The `<tta:response>` element can contain the following `<obj>` child elements.

Object Name	Description
`webtopgroup`	An object representing an applications group. Can contain one or more `webtopitem` objects. See "`webtopgroup` Object" on page 106.
`webtopitem`	An object representing an application or document. See "`webtopitem` Object" on page 107.

`webtopgroup` Object

A `webtopgroup` object is designated by the `<obj name="webtopgroup">` element. A `webtopgroup` object represents a group of applications and acts as a container object for one or more `webtopitem` objects.

A `webtopgroup` object can contain `<obj name="webtopitem">` child elements. See "`webtopitem` Object" on page 107 for more details about this object.

The name for the `webtopgroup` object is defined by the `groupname` attribute.

The following is an example of a `webtopgroup` object response message.

```
<obj name="webtopgroup" groupname=".../_ens/o=applications/cn=AppsGroup">
 <obj name="webtopitem">
  <attr name="scottaunixaudiopreload">0</attr>
  <attr name="scottaappletheight">600</attr>
  ...
  <attr name="cn">dttterm</attr>
  ...
 </obj>
 <obj name="webtopitem">
  <attr name="scottaunixaudiopreload">0</attr>
  <attr name="scottaappletheight">600</attr>
  ...
  <attr name="cn">gnome session</attr>
  ...
 </obj>
 <obj name="webtopitem">
  <attr name="scottaunixaudiopreload">0</attr>
  <attr name="scottaappletheight">600</attr>
  ...
  <attr name="cn">cde session</attr>
  ...
 </obj>
</obj>
```

webtopitem Object

A webtopitem object is designated by the <obj name="webtopitem"> element. A webtopitem object defines an application or document assigned to the user.

A webtopitem object can contain <attr> attribute elements. Appendix B contains a full listing of the available attributes for application and document objects.

The returned attribute elements vary, depending on the application type. The supported application types are described in "Application Type" on page 95. More information about the attributes of SGD objects can be found in the *Sun Secure Global Desktop 4.4 Reference Manual*.

In the following example, three webtopitem objects are returned in a response message.

```
<tta:response>
 <obj name="webtopitem">
  <attr name="scottaappletheight">480</attr>
  <attr name="scottawebtopobjectname">.../_ens/o=applications/cn=
Mozilla</attr>
  <attr name="objectclass">scottaxapplication</attr>
  <attr name="scottaapplicationplacement">multiplewindows</attr>
  <attr name="scottasuspend">session</attr>
  <attr name="cn">Mozilla</attr>
  <attr name="scottawebtopobjectclass">scottaxapplication</attr>
  <attr name="scottascalable">0</attr>
  <attr name="scottafilepath">/usr/sfw/bin/mozilla</attr>
  <attr name="scottaappletwidth">640</attr>
  <attr name="scottaicon">firefox.gif</attr>
  <attr name="scottadisplayenginepage">xde.html</attr>
  <attr name="scottamaxinstances">3</attr>
  <attr name="scottafullscreen">0</attr>
  <attr name="scottarunninginstances">0</attr>
 </obj>
 <obj name="webtopitem">
  <attr name="scottalocation">/sgdadmin</attr>
  <attr name="scottawebtopobjectclass">scottahtmldocument</attr>
  <attr name="scottaicon">world.gif</attr>
  <attr name="objectclass">scottahtmldocument</attr>
  <attr name="scottawebtopobjectname">.../_ens/o=Tarantella System Objects/cn=
  Administration Console</attr>
  <attr name="scottaoutofplace">1</attr>
  <attr name="cn">Administration Console</attr>
  <attr name="scottarunninginstances">0</attr>
 </obj>
 <obj name="webtopitem">
  <attr name="scottaappletheight">480</attr>
  <attr name="scottawebtopobjectname">.../_ens/o=applications/cn=VT420</attr>
```

```
   <attr name="objectclass">scottacharacterapplication</attr>
   <attr name="scottaapplicationplacement">awtwindow</attr>
   <attr name="scottasuspend">session</attr>
   <attr name="cn">VT420</attr>
   <attr name="scottawebtopobjectclass">scottacharacterapplication</attr>
   <attr name="scottafilepath">/opt/tarantella/bin/scripts/shell</attr>
   <attr name="scottaappletwidth">640</attr>
   <attr name="scottaicon">vt420.gif</attr>
   <attr name="scottadisplayenginepage">tde.html</attr>
   <attr name="scottamaxinstances">3</attr>
   <attr name="scottafullscreen">0</attr>
   <attr name="scottarunninginstances">0</attr>
  </obj>
 </tta:response>
```

Operations

The following table lists the available operations for the `ITarantellaWebtopContent` web service.

Operation Name	Purpose
`lookupWebtopItems`	Looks up the details of assigned applications.
`searchWebtopContent`	Searches for assigned applications, removing group hierarchy.
`searchWebtopContentHierarchy`	Searches for assigned applications, preserving group hierarchy.

`lookupWebtopItems`

Looks up attribute details for the specified objects.

Request Message

The format of the request message for this operation is as follows:

```
<lookupWebtopItems>
 <sessioncookie>xsd:string</sessioncookie>
 <objectnames>xsd:string</objectnames> [1..unbounded]
 <desattrs>xsd:string</desattrs> [1..unbounded]
</lookupWebtopItems>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <obj name="webtopitem">  [0..unbounded]
</tta:response>
```

searchWebtopContentHierarchy

Searches a user's assigned applications, preserving group information.

Request Message

The format of the request message for this operation is as follows:

```
<search>
 <sessioncookie>xsd:string</sessioncookie>
 <searchspec>xsd:string</searchspec>
 <desattrs>xsd:string</desattrs> [1..unbounded]
 <groups>xsd:string</groups> [1..unbounded]
</search>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <obj name="webtopgroup"> [0..unbounded]
  <obj name="webtopitem"> [0..unbounded]
 </obj>
</tta:response>
```

searchWebtopContent

Searches a user's assigned applications, removing group information.

Request Message

The format of the request message for this operation is as follows:

```
<search>
```

```
 <sessioncookie>xsd:string</sessioncookie>
 <searchspec>xsd:string</searchspec>
 <desattrs>xsd:string</desattrs> [0..unbounded]
</search>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <obj name="webtopitem"> [0..unbounded]
</tta:response>
```

CHAPTER 17

The ITarantellaWebtopSession Web Service

This chapter describes how you can use the `ITarantellaWebtopSession` web service to authenticate users, start and end user sessions, and configure and start the SGD Client. The available operations and SOAP message elements for this web service are described in detail.

This chapter includes the following topics:

- "SGD User Sessions Overview" on page 111
- "Using ITarantellaWebtopSession" on page 115
- "SOAP Reference" on page 127

SGD User Sessions Overview

This section includes some background information about SGD user sessions and related topics. The following topics are covered:

- User sessions
- Session moving and session joining
- SGD Client Helper
- SGD Client

About User Sessions

A user session is the means for tracking a user's interaction with an SGD server, for example, by recording which applications a user has started.

When you log in to SGD, a new user session is created. The user session is hosted by the SGD server you log in to. The user session ends when you log out of SGD.

In SGD web services, a user session is represented by a `webtopsession` object. A `webtopsession` object can have many possible attributes. See "`webtopsession` Object" on page 133.

If you want your application to work with SGD user sessions, you must use the operations provided by `ITarantellaWebtopSession`. Every time an application wants to use SGD web services, it needs to do the following:

1. Authenticate the SGD user and create a new user session, using the `authenticate` or `authenticateExt` operations.[1]
2. Perform further web service operations needed by the application. This might involve running the SGD Client, using the `setTCCConfiguration` and `startTCC` operations.
3. When finished, end the user session, using the `endSession` operation.

See "Using ITarantellaWebtopSession" on page 115 for more details of how to use the `ITarantellaWebtopSession` web service with SGD user sessions.

Controlling Access to SGD Web Services

When SGD creates a user session, it assigns the following unique values to the session:

- Session cookie
- Session ID

The session cookie and session ID values never change during the lifetime of the user session.

Many web services operations require a valid session cookie as their first parameter. SGD controls access to web services operations by checking the validity of the session cookie.

As the session cookie is the key to a user's authority to perform web services operations it is only returned once, in the response to the `authenticate`, `authenticateExt`, or `ITarantellaExternalAuth.setSessionIdentity` operation.

An application using SGD web services typically parses the value of the session cookie from the response and stores it in memory for subsequent use.

1. Third-party authentication is also supported. Given a user name authorized by a third party, you can use the `setSessionIdentity` operation of the `ITarantellaExternalAuth` web service to create a new user session. See also Chapter 12.

If you need to pass the session cookie to another application, ideally you should keep it secret. The `getSessionToken` and `getSessionCookie` operations are provided to help you do this. Together, they provide a secure way to pass a session cookie to another application.

Use the `getSessionToken` operation to generate a session token for a given session cookie, as follows:

```
String xml = userSess.getSessionToken(sessionCookie);
IResponse resp = parser.parse(xml);
//Get the session token
String sessionToken = resp.getAttribute("sessiontoken")[0];
```

You can then pass the session token to the application. This avoids exposing the session cookie.

In the other application, use the `getSessionCookie` operation to retrieve the session cookie for a given session token, as follows:

```
String xml = userSess.getSessionCookie(sessionToken);
IResponse resp = parser.parse(xml);
//Get the session cookie
String cookie = resp.getAttribute(ISchemaNames.SESSIONCOOKIE)[0];
```

Note – A session token can only be used *once* to look up the session cookie with the `getSessionCookie` operation. Subsequent `getSessionCookie` operations using the same session token return an error.

Only an SGD Administrator can perform web service operations on another person's user session. To do this, they look up session ID of the user session, for example, using `adminSearchStart` or `adminSearchSessions`, and then specify their own session cookie as the first parameter in an Administrator-level web service operation. You cannot look up the session cookie for another user session.

When a user session ends, the SGD server deletes the record of the session cookie, ensuring that it cannot be used again to access SGD web services.

Session Moving and Session Joining

A user session can have an associated client ID. The application is responsible for generating the client ID, and its value must be unique. The application stores the value, for example, as a cookie and submits it to the SGD server with the `authenticate` operation.

Once a user session has been authenticated and started, the SGD server checks the client ID, if submitted.

If the user has a user session with a different client ID already running, the existing user session is *moved*. If the client ID details are the same, the user sessions *join*.

When a user session is moved, the old user session ends and any application sessions associated with it are either suspended or ended, depending on the resumability setting.

When user sessions join, a new user session is not created and any application sessions associated with the original user session continue to run. This enables a user to close their browser session without affecting their application sessions. When they log in again, their applications simply display as though they had never logged out.

Note – Session joining is only supported for user sessions hosted on the same SGD server in the array.

Views of a User Session

In SGD web services, each time you join a user session you create a new *view* of the user session. The current views for a user session are listed in the `views` object. Each individual view is identified by a `viewid` attribute. See "`views` Object" on page 132 for more information.

Closing a view ends or suspends any application sessions owned by that view, depending on the resumability setting.

You can use the `associateTCC` operation to enable a view to use an SGD Client connection that has been started by another view.

About the SGD Client Helper

If you are using a web browser with Java technology enabled, the SGD Client is supported by a Java applet called the SGD Client Helper.

The SGD Client Helper performs the following functions:

- Downloads and installs the SGD Client.
- Obtains proxy server settings from the web browser and sends them to the SGD Client. This depends on the settings in the user's profile.
- Starts the SGD Client. This only happens when a user starts a web browser and goes to the login URL.

- Responds to instructions received from the SGD Client, for example, prompting the web browser to redraw the screen.

The SGD Client Helper is optional. Alternatively, you can download the SGD Client to your client device and run the SGD Client from the command line.

See "Using the SGD Client" on page 117 for more details about how the SGD Client Helper can be used to download and install the SGD Client automatically.

About the SGD Client

The SGD Client (also called the *TCC*) is a program that runs on the client device and communicates with the SGD server. You only need to use the SGD Client if you want to run applications.

The SGD Client performs the following functions:

- Gets information about the client device, such as the operating system, local printers, and client drives
- Manages the display of applications
- Maintains a SOAP connection with the SGD server, using the Adaptive Internet Protocol (AIP) protocol,
- Receives and acts on events from the SGD server, for example, the arrival of a print job

See "Using the SGD Client" on page 117 for more details about how to download, configure, and start the SGD Client.

Using ITarantellaWebtopSession

The `ITarantellaWebtopSession` web service can be used to create and manage SGD user sessions. `ITarantellaWebtopSession` is an essential web service to use if you are building SGD applications.

This section describes how you can use the `ITarantellaWebtopSession` web service to do the following tasks:

- Authenticate an SGD user
- Start a user session
- Configure the SGD Client
- Start the SGD Client
- Look up attributes for a user session

- Modify a user session
- End a user session

The ITarantellaWebtopSession web service defines the following types of operations:

- **User-level operations.** These operations enable users to perform operations on their own user session.
- **Administrator-level operations.** These operations enable an SGD Administrator to search for, configure, and manage a user session belonging to another user.

Administrator-level operations are designated by an admin prefix to the operation name, for example, adminSearchStart.

Most of the tasks listed in the following sections require a sessionCookie parameter, where sessionCookie is the session cookie of the user session performing the operation. Administrator-level operations require a session cookie for an SGD Administrator user session.

Authenticating an SGD User

Use the authenticate operation to authenticate an SGD user, as follows:

```
authenticate(username,passwd,"clientId","en");
```

Here, the user's SGD user name and password are given by username and passwd.

The clientId string is an optional identifier for the client device. This identifier determines whether the SGD server joins or moves any existing user session. Sessions are joined when the authentication credentials map to the same user and the clientId used to authenticate both sessions is the same. See "Session Moving and Session Joining" on page 113 for information about session joining and session moving.

The en string is an optional parameter that identifies the preferred locale for the user. This parameter defines the language used for any dialogs and messages displayed by the SGD Client and overrides the language defined in the user's client profile. Specifying a locale that is not supported by SGD results in the default locale, en, being used.

The value of the preferred locale parameter is stored in the applicationattributes attributes group of the webtopsession object and so can be accessed by an application as required.

If the authentication is successful, a new user session is started. Attributes relating to the user session are returned in the SOAP response. Some of the most useful attributes returned are as follows:

- **Session cookie.** The session cookie attribute, `scottasessioncookie`, is a cookie that uniquely identifies the user session. This attribute is a mandatory input parameter for all other `ITarantellaWebtopSession` operations.
- **Name.** The name attribute, `cn`, is the common name of the user.
- **Session ID.** The session ID attribute, `scottasessionid`, is a public identifier for the user session. An SGD Administrator can use the session ID attribute to perform operations on other people's user sessions.
- **User profile.** The session template owner attribute, `scottasessiontemplateowner`, defines the user profile for the user session. This determines the SGD settings for a user and can determine the assigned applications.
- **User identity.** The session owner attribute, `scottasessionowner`, defines the user identity that owns the user session.
- **Views.** When a user session *joins* an existing user session, a new view is created. The SGD Client connection attribute, `tccconnection`, defines whether an SGD Client connection is associated with the view. See "Session Moving and Session Joining" on page 113 for more details about session joining.

If the login mechanism challenges for additional credentials following an `authenticate` operation, for example, if there is an ambiguous login, use the `authenticateExt` operation. This operation enables additional credentials to be supplied when authorizing a user.

Starting a User Session

If authentication is successful, the `authenticate` operation starts a new user session on the SGD server.

See "Authenticating an SGD User" on page 116 for more details.

Using the SGD Client

The SGD Client is not installed on the client device by default. Using the SGD Client on the client device involves the following steps:

1. Specify a SGD Client to download
2. Specify additional connection settings (optional)
3. Download the SGD Client
4. Start the SGD Client

These tasks are described in the following sections.

Specifying a SGD Client to Download

The SGD Client program must be downloaded from the SGD server to the client device, as follows:

- If you are using a browser with Java technology enabled, the download process is handled automatically by the SGD Client Helper.
- If you are not using a browser with Java technology enabled, you need to download and install the client manually.

The SGD Client Helper applet and SGD Client are contained in Java™ Archive (JAR) files on the SGD server. You use the `setTCCConfiguration` operation to specify a location for downloading these JAR files from.

Multiple versions of the SGD Client are available on the SGD server, one for each of the supported SGD client operating systems (OS). Use the `setTCCConfiguration` operation to supply client device information to the SGD server, to ensure that the correct SGD Client binary for the client operating system (OS) is downloaded. Typical use of `setTCCConfiguration` is shown in the following code example:

```
//Get browser details
String userAgent = request.getHeader("User-Agent");
//Configure download directory for the SGD Client and SGD Client Helper
String baseUrl = "http://server.example.com/sgd/tcc";
//Configure SGD Client version
String version = "4.40.903";
//Configure type of SGD Client Helper applet
String type = "java";
...
userSess.setTCCConfiguration(sessionCookie,userAgent,baseUrl,version,type);
```

The `userAgent` parameter contains information from the browser's User-Agent string. The User-Agent string can be used to identify the browser and OS on the client device. For example, a Firefox browser on Solaris can return the following User-Agent string:

```
Mozilla/5.0 (X11; U; SunOS sun4v; en-US; rv:1.8.1.8)
Gecko/20071019 Firefox/2.0.0.8
```

The `userAgent` parameter is used to select the correct SGD Client binary for the client device OS and, if you are using a browser with Java technology enabled, to determine the correct applet used by the SGD Client Helper.

The `baseUrl` and `type` parameters are used to construct a download location for the SGD Client, for example:

```
http://server.example.com/sgd/tcc/java
```

The `version` parameter identifies the version of the SGD Client. This is typically the same as the SGD version and is used to configure the install directory on the client device. For example, on UNIX platform devices, the SGD Client is installed to the `$HOME/.tarantella/tcc/`*version* directory, where *version* is the SGD Client version.

Note – You still need to use the `setTCCConfiguration` operation, even if you do not plan to use the SGD Client Helper to download the SGD Client. Specify a User-Agent string for `setTCCConfiguration`, to ensure that the `startTCC` operation returns the correct attributes for your client OS.

Specifying Additional Connection Settings

The `setTCCConfigurationOverrides` operation differs from `setTCCConfiguration` in that it also enables you to specify additional settings for an SGD Client connection.

The additional settings are called *overrides*, as they override the default settings. Overrides are specified as an array of `Item` objects, as follows:

```
userSess.setTCCConfiguration (sessionCookie,userAgent,baseUrl, \
version,type,Item[] overrides);
```

The supported overrides are listed in the following table.

Override	Description
`IAttributeNames.BASEROUTE`	Routing information to connect to the SGD server.
`IAttributeNames.FAILEDCONNECTIONURL`	URL that the SGD Client loads if it fails to establish a connection to the server.
`IAttributeNames.FIREWALLTRAVERSAL`	Flag indicating that the SGD server is running in firewall traversal mode.
`IAttributeNames.HOST`	SGD server name for the SGD Client connection.
`IAttributeNames.PORT`	Port for the SGD Client connection.
`IAttributeNames.SECURITY`	Security of the initial connection. *This attribute is deprecated.*
`IAttributeNames.TCCPORT`	Port on which the SGD Client listens for connections.

Override	Description
`IAttributeNames.TCCPORTFILE`	Name of the file where the SGD Client writes the port on which it listens for connections.
`IAttributeNames.TCCROAMING`	Flag indicating that the SGD Client should assume roaming profiles.
`IAttributeNames.TCC_DIRECTORY`	Directory where the SGD Client is installed.
`IAttributeNames.TCC_ARGUMENTS`	Command-line arguments to the SGD Client.

If the connection between the client device and the SGD server is not direct, for example, if a gateway is being used, you can use `setTCCConfigurationOverrides` to configure routing information for the connection.

Downloading the SGD Client

The SGD Client can be downloaded to the client device either automatically or manually, as follows:

- **Automatic download.** If you are using a browser with Java technology enabled, the SGD Client Helper applet handles downloading of the SGD Client program to the client device. You can configure the download location for the SGD Client Helper and SGD Client using `setTCCConfiguration`. See "Specifying a SGD Client to Download" on page 118.
- **Manual download.** If you are not using a browser with Java technology enabled, you can download and install the SGD Client program manually. The SGD Client can be downloaded from `http://`*server.example.com*, where *server.example.com* is the name of an SGD server.

Starting the SGD Client

The `startTCC` operation returns the information you need to start the SGD Client. See "Attributes Returned by `startTCC`" on page 123 for information about the returned attributes.

Depending on the availability of a browser with Java technology enabled on the client device, you can start the SGD Client in one of the following ways:

- **Using the SGD Client Helper.** The SGD Client Helper applet handles startup of the SGD Client automatically. The SGD Client Helper applet is only available if you are using a browser with Java technology enabled.

- **From the command line.** Start the SGD Client from the command line on the client device. You must write your own code to do this. Also, the SGD Client must be downloaded and installed on the client device.

These scenarios are discussed in the following sections.

Using the SGD Client Helper to Start the SGD Client

If you are using a browser with Java technology enabled, you can use the SGD Client Helper applet to start the SGD Client automatically.

The `startTCC` operation generates the required HTML and parameters needed to start the SGD Client Helper applet. This data is returned in the `codefragment` attribute and can be parsed from the response as follows:

```
String xml = userSess.startTCC(sessionCookie,HELPER_ID,"","");
IResponse resp = parser.parse(xml);
//Get the applet HTML and parameters
String frag = resp.getAttribute(IAttributeNames.CODEFRAGMENT)[0];
```

Loading the contents of `codefragment` into a browser with Java technology enabled starts the SGD Client Helper applet.

Starting the SGD Client From the Command Line

If you are not using a browser with Java technology enabled, you can start the SGD Client by constructing a command line using attributes returned in the `startTCC` response.

The following code example shows how you can parse the required attributes from the `startTCC` response:

```
String xml = userSess.startTCC(sessionCookie,"","","");
IResponse resp = parser.parse(xml);
//Get the cookie for the SGD Client connection
String conn_cookie = resp.getAttribute(IAttributeNames.CONNECTCOOKIE)[0];
//Get the SGD server
String server = resp.getAttribute(IAttributeNames.HOST)[0];
//Get the connection port
String port = resp.getAttribute(IAttributeNames.PORT)[0];
```

Other attributes can be parsed from the response, depending on the required command line.

You start the SGD Client with the `tcc` command on Microsoft Windows client platforms, or the `ttatcc` command on UNIX, Linux, or Mac OS X client platforms. The following example command line starts the SGD Client and connects to an SGD server:

```
$ ttatcc -connectioncookie conn_cookie -server server.example.com -port 5307
```

where *server.example.com* is an SGD server, and *conn_cookie* is the connection cookie returned by the `startTCC` operation.

An application can use the `java.lang.Runtime.exec` method, as follows, to execute this command line:

```
Runtime rt = Runtime.getRuntime();
String[] cmd = new String[] \
{"/bin/sh","-c","$HOME/bin/ttatcc","-connectioncookie",conn_cookie, \
"-server","server.example.com","-port","5307"};
Process p = rt.exec(cmd);
```

Some useful command-line arguments for the `ttatcc` and `tcc` commands are listed in the following table.

Argument	Description
`-baseroute`	Network route the SGD Client uses to connect to the SGD server.
`-connectioncookie`	Cookie used by the SGD server to identify the user session for which the SGD Client is being used.
`-firewalltraversal`	Indicates that the SGD server is using firewall forwarding. Connections to the SGD server and the web application both use the same port, usually TCP port 443.
`-no-browser`	Do not start a browser on SGD Client startup.
`-port`	Port on which the SGD Client connects to the SGD server. By default, this is TCP port 5307.
`-portfile`	Name of a file into which the SGD Client writes its listening port number.
`-psn`	For use with Mac OS X client devices only. Ensures an X server is running.
`-secure`	Create a secure connection to the SGD server. *This attribute is deprecated.*
`-server`	Fully-qualified DNS name of the SGD server.

See the *Sun Secure Global Desktop 4.4 Administration Guide* for more details about the available command-line arguments for the SGD Client.

Attributes Returned by `startTCC`

The `startTCC` operation returns the information required to start the SGD Client. The following table lists the returned attributes for `startTCC`.

Attribute	**Description**
`IAttributeNames.BASEROUTE`	Routing information for the SGD Client to connect to the SGD server.
`IAttributeNames.CODEFRAGMENT`	A fragment of HTML code used to configure and start the SGD Client Helper applet. The SGD Client Helper applet downloads and starts the SGD Client.
`IAttributeNames.CONNECTCOOKIE`	Connection cookie used by the SGD Client to connect to the SGD server.
`IAttributeNames.FIREWALLTRAVERSAL`	Flag indicating that the SGD server is running in firewall traversal mode.
`IAttributeNames.HELPERID`	Identifier for the SGD Client Helper used to download and start the SGD Client.
`IAttributeNames.HOST`	Name of the SGD server the SGD Client connects to.
`IAttributeNames.PORT`	Port for the SGD Client connection.
`IAttributeNames.SESSIONSECURITY`	Security level, secure or standard, for the connection between the SGD Client and SGD server.
`IAttributeNames.TCCPORT`	Port on which the SGD Client listens for connections.
`IAttributeNames.TCCPORTFILE`	Name of the file where the SGD Client writes the port on which it listens for connections.
`IAttributeNames.VERSION`	Version of the SGD Client version.

The following code example shows a typical response for a `startTCC` operation.

```
<tta:response>
 <attr name="helperid">Helper1</attr>
 <attr name="connectioncookie">0123456789012:server.example.com:...</attr>
 <attr name="security">STD</attr>
 <attr name="server">server.example.com</attr>
 <attr name="port">5307</attr>
 <attr name="version">4.40.907</attr>
 <attr name="baseroute"></attr>
 <attr name="tccport"></attr>
 <attr name="tccportfile">1195051413347.port</attr>
 <attr name="firewalltraversal">false</attr>
 <attr name="codefragment"><![CDATA[
 <applet archive="ttalwG-jps.jar,ttalwwin32G-jps.jar"
 width="2"
 height="2"
 codebase="http://server.example.com/sgd/tcc/java"
 code="com.tarantella.tta.client.tcc.lwplugin.pluginG.TCCHelper"
 alt="Please enable Java before continuing.">
 <param name="helperid" value="Helper1">
 <param name="connectioncookie" value="0123456789012:server.example.com:...">
 <param name="security" value="STD">
 <param name="server" value="server.example.com">
 <param name="port" value="5307">
 <param name="version" value="4.40.907">
 <param name="baseroute" value="">
 <param name="tccport" value="">
 <param name="tccportfile" value="1195051413347.port">
 <param name="StartTCC" value="true">
 <param name="firewalltraversal" value="false">
 <param name="TCCURL" value=
"http://server.example.com/sgd/tcc/win32/tcc_comp.exe">
 <param name="ForceTCCDownload" value="false">
 <param name="PreferredLanguage" value="en">
 <param name="tccroaming" value="missing">
 <param name="BrowserId" value="1">
 </applet>]]>
 </attr>
</tta:response>
```

The `<codefragment>` attribute contains the HTML code required to configure and start the SGD Client Helper applet.

Looking Up User Session Attributes

Use the `lookupSession` operation to return the attributes for a user session.

A user session can have many attributes. The following example uses the `IConstants.SERVER_ATTRS` shortcut with the `lookupSession` operation to return only those attributes that are set by the SGD Server.

```
//Use a shortcut to define a group of attributes to return
String[] attrs = new String[]{IConstants.SERVER_ATTRS};
//Lookup attributes for the user session
String xml = userSess.lookupSession(sessionCookie,attrs);
```

You can use the following shortcuts with the lookupSession and adminLookupSession operations:

Shortcut	Description
IConstants.APPLICATION_ATTRS	Returns attributes set by the web application.
IConstants.CLIENT_ATTRS	Returns attributes set by the SGD Client.
IConstants.CONNECTION_ATTRS	Returns attributes for the connection between the SGD Client and the SGD server.
IConstants.PROFILE_ATTRS	Returns attributes from the user's client profile.
IConstants.SERVER_ATTRS	Returns attributes set by the SGD server.

See "Using Shortcuts to Return Groups of Attributes" on page 129 for more details about using shortcuts to return specific groups of attributes.

SGD Administrators can use the adminLookupSession operation to look up attributes for another person's user session, as follows:

```
adminLookupSession(sessionCookie,sessionId,attrs);
```

In this example, the user session to look up attributes for is identified by sessionId. The attrs parameter defines the attributes to return. Shortcuts can be used to specify groups of attributes. A sessionCookie for an SGD Administrator user session must be specified when using this operation.

Modifying a User Session

You can use the modifySession operation to modify selected attributes for a user session, as follows:

```
Item printQueue = new Item();
printQueue.setKey(IAttributeNames.CLIENT_PRINT_QUEUE);
printQueue.setValue(IConstants.PAUSED);
Item data = new Item[]{printQueue};
String mode = IConstants.MODE_REPLACEATTR;
...
userSess.modifySession(sessionCookie,data,mode);
```

In this example, the status of the user's print queue is changed to *paused*.

The `mode` parameter determines how the attribute is modified. The supported `mode` options for `modifySession` are as follows.

Mode	**Description**
`IConstants.MODE_APPENDATTR`	Append the values to any existing values on the named attribute.
`IConstants.MODE_REMOVEATTR`	Remove the named attributes. Attributes that are not named are left untouched.
`IConstants.MODE_REPLACEATTR`	Replace the values on the named attributes with new values. Attributes that are not named are left untouched.
`IConstants.MODE_REPLACEALLATTR`	Remove all attributes and replace them with the named values.

Note – The `modifySession` operation can only be used to change the attributes in the `applicationattributes` attributes group object.

Ending a User Session

You can end a user session as follows:

```
endSession(sessionCookie,true);
```

Where `sessionCookie` defines the user session to end. The `true` flag logs the user out of SGD, after ending the session.

SGD Administrators can use the `adminEndSession` operation to end other person's user sessions as follows:

```
adminEndSession(sessionCookie,sessionId[],true);
```

In this example, the user sessions to end are identified using an array of session ID attributes. The `true` flag means users are logged out of SGD. A `sessionCookie` for an SGD Administrator user session must be specified when using this operation.

SOAP Reference

This section describes the operations and SOAP message elements for the `ITarantellaWebtopSession` web service.

The following table lists the WSDL file location, namespace information, and end point URL for the `ITarantellaWebtopSession` web service, where *server.example.com* is an SGD server.

WSDL File	`http://`*server.example.com*`/axis/services/document/webtopsession?wsdl`
SOAP Operation Namespace	`http://sgd.sun.com/webservices/document/webtopsession`
HTTP POST URL	`http://`*server.example.com*`/axis/services/document/webtopsession`

Request Message Elements

The following table lists the request message elements for the `ITarantellaWebtopSession` web service.

Element	Data Type	Description
`baseurl`	`xsd:string`	Location on an SGD server from which the SGD Client and SGD Client helper are downloaded.
`creds`	`tns1:Item`	Array of additional credentials supplied for an `authenticateExt` operation.
`clientId`	`xsd:string`	An optional identifier for the client device. Used to determine if the SGD server joins or moves an exisiting session belonging to the user.
`clientData`	`xsd:string`	Identifies the SGD client attributes to return using the `refreshClientInfoEx` operation.
`data`	`tns1:Item`	For the `modifySession` operation, an array of attributes you want to change.
`desattrs`	`xsd:string`	Array of attributes to be returned in the response. The attributes can be any of the attributes in the `webtopsession` object. List the individual attributes you want to return or use a shortcut to return a group of attributes. See "Using Shortcuts to Return Groups of Attributes" on page 129. The attributes are returned in their corresponding object elements.
`endview`	`xsd:boolean`	Whether a view created using the `createView` operation is ended when the parent view is ended.

Element	Data Type	Description
feature	tns1:Item	Array of parameters for a multi-stage login. This is returned in the SOAP fault from a previous authentication request.
helperid	xsd:string	Identifier for the SGD Client Helper used to download and start the SGD Client.
id	xsd:string	Identifier for a view on a user session.
locale	xsd:string	Preferred locale for the user session. This determines the language used for messages from the SGD server.
logout	xsd:boolean	After ending a user session, whether to log the user out of SGD.
mode	xsd:string	For the modifySession operation, the type of modification to perform. See "Modifying a User Session" on page 125.
noofresults	xsd:int	The number of results to return from a search. If noofresults is zero or a negative integer, all search results are returned.
overrides	tns1:Item	Array of attribute values to override, using the setTCCConfigurationOverrides operation.
passwd	xsd:string	Password credential for logging in to SGD.
searchid	xsd:string	A unique identifier string for a user session search. The value of this element is returned when you use the adminSearchStart operation.
searchspec	xsd:string	An RFC2254-compliant LDAP search filter for selecting user sessions. You can search for any attribute in the *serverattributes* group of attributes. Use (objectclass=scottawebtopsession) to search for all user sessions.
searchtype	xsd:string	Reserved for future use. Use an empty string.
sessioncookie	xsd:string	The session cookie of the user session performing the operation. For Administrator-level operations, the session cookie must be for an SGD Administrator's user session.
sessionid	xsd:string	Public identifier for a user session. The sessionid is used in Administrator-level operations, such as adminEndSession, to perform operations on another person's user session.
sessiontoken	xsd:string	Session token generated by the getSessionToken operation. Used in the getSessionCookie operation, to look up the sessioncookie for a user session.
target	xsd:string	Name of the HTML frame or window that displays the SGD Client connection message. Use _blank to display the message in a new browser window.

Element	Data Type	Description
token	xsd:string	Cookie sent to SGD Client when SGD server is restarted. Used in the restart operation.
type	xsd:string	Type of SGD Client Helper to use. Only java is supported as a value for this element.
url	xsd:string	URL for the SGD Client connection message.
useragent	xsd:string	Information about the browser used to host the SGD Client Helper.
username	xsd:string	User name credential for logging in to SGD.
version	xsd:string	Version number of the SGD Client. This is the same as the SGD version.

Item object

Some of the ITarantellaWebtopSession operations require an request message element of data type tns1:Item. This data type represents an Item object.

An Item object is a utility object that can be used to store and manipulate objects in SGD web services. An Item object consists of a key-value pair, as follows:

```
String key;
Object value;
Item item = new Item(key,value);
```

The com.tarantella.tta.webservices.Item class includes get and set methods for configuring the parameters of an Item object.

Using Shortcuts to Return Groups of Attributes

User session attributes are stored in groups, depending on their source. With operations such as lookupSession or adminSearchSession, you can specify the attributes you want to return. Alternatively, you can use a shortcut in these operations to return a whole group of attributes.

The available shortcuts for ITarantellaWebtopSession operations are as follows:

- IConstants.APPLICATION_ATTRS. Returns attributes that are set by the application. The attributes are returned in an <obj name= "applicationattributes"> object. See "applicationattributes Object" on page 134.

- IConstants.CLIENT_ATTRS. Returns attributes that are sent by the SGD Client. The attributes are returned in an <obj name="clientattributes"> object. See "clientattributes Object" on page 135.
- IConstants.CONNECTION_ATTRS. Returns attributes for the connection between the SGD Client and the SGD server. The attributes are returned in an <obj name="connectionattributes"> object. See "connectionattributes Object" on page 137.
- IConstants.PROFILE_ATTRS. Returns attributes from the user's client profile. The attributes are returned in an <obj name="profileattributes"> object. See "profileattributes Object" on page 138.
- IConstants.SERVER_ATTRS. Returns attributes set by the SGD server. The attributes are returned in an <obj name="serverattributes"> object. See "serverattributes Object" on page 140.

Response Message Elements

This section describes the elements returned in response messages for the ITarantellaWebtopSession web service.

<tta:response> Element

The <tta:response> element is the top-level container used for all responses from SGD web services.

An empty response (<tta:response/>) indicates that the operation has no data to return. For example, if lookupSession does not find a match.

The <tta:response> element can contain the following <obj> child elements.

Object Name	Description
src	An object representing an SGD server in the array. A src object can contain one or more webtopsession objects. See "src Object" on page 131.
views	A container object for one or more view objects. See "views Object" on page 132.
webtopsession	An object representing a user session. See "webtopsession Object" on page 133.

The <tta:response> element can contain the following <attr> child elements.

Attribute Name	Description
arrayMember	The SGD server in the array that hosts the user session. *This attribute is deprecated.*
cn	Common name of the user in the SGD datastore.
isadministrator	Whether the user is an SGD Administrator.
scottacdmlastactivity	The number of seconds since there was client drive mapping activity for the user session.
scottaemulatorsessionlastactivity	The number of seconds since there was application activity for the user session.
scottaprintlastactivity	The number of seconds since there was printing activity for the user session.
scottasessioncookie	A cookie that uniquely defines the user session. This is a mandatory parameter in most ITarantellaWebtopSession operations.
scottatheme	The theme used for the user's webtop. *This attribute is deprecated.*
searchId	A unique identifier string for a search. Returned by an adminSearchStart or adminSearchNext operation. Can be used to identify the search in subsequent adminSearchNext and adminSearchEnd operations. An empty <searchid> element indicates that there are no results to return.
sessiontoken	A one-time token, that can be used to look up the session cookie. See the getSessionCookie operation.
total	Number of results returned by a user session search. Returned by the adminCount operation.

src Object

A src object is designated by the <src name="server.example.com"> element, where *server.example.com* is the name of an SGD server. A src object represents an SGD server in the array.

A src object is returned when you use the following SGD Administrator search operations:

- adminSearchStart
- adminSearchNext
- adminSearchSession

Search results for each SGD server in the array are stored in a `src` object, as follows:

```
<tta:response>
 <src name="server1.example.com"
  <obj name="webtopsession">
   ...
  </obj>
  <obj name="webtopsession">
   ...
  </obj>
 </src>
 <src name="server2.example.com"
  <obj name="webtopsession">
   ...
  </obj>
  <obj name="webtopsession">
   ...
  </obj>
 </src>
</tta:response>
```

`views` Object

A `views` object is designated by the `<obj name="views">` element. A `views` object represents the possible views for a user session. This object can be returned by the `authenticate` or `createView` operation.

A `views` object contains one or more `<obj name="view">` child objects. See "`view` Object" on page 133 for more details about this object.

The following example shows part of an `authenticate` response messsage, where a `views` object is returned.

```
<tta:response>
 ...
 <obj name="views">
  <obj name="view">
   <attr name="viewid">1<attr>
   <attr name="tccconnection">true</attr>
  </obj>
  <obj name="view">
   <attr name="viewid">2<attr>
   <attr name="tccconnection">false</attr>
  </obj>
 </obj>
</tta:response>
```

view Object

A view object is designated by the <obj name="view"> element. A view object represents a single view for a user session.

A view object contains the following <attr> child elements.

Attribute Name	Description
viewid	A number identifying the user session view.
tccconnection	If the view is associated with an SGD Client connection.

webtopsession Object

A webtopsession object is designated by the <obj name="webtopsession"> element. A webtopsession object represents an SGD user session.

A webtopsession object can contain the following <obj> child elements. These objects define groups of attributes.

Object Name	Description
applicationattributes	An attributes group object containing attributes set by the application. See "applicationattributes Object" on page 134.
clientattributes	An attributes group object containing attributes set by the SGD Client. See "clientattributes Object" on page 135.
connectionattributes	An attributes group object containing attributes related to the connection between the SGD Client and the SGD server. See "connectionattributes Object" on page 137.
profileattributes	An attributes group object containing attributes from the user's client profile. See "profileattributes Object" on page 138.
serverattributes	An attributes group object containing attributes set by the SGD server. See "serverattributes Object" on page 140.

The following example shows how attribute group child objects are returned in a parent webtopsession object.

```
<tta:response>
 <obj name="webtopsession">
  <obj name="clientattributes">
   ...
  </obj>
  <obj name="connectionattributes">
   ...
  </obj>
  <obj name="applicationattributes">
   ...
  </obj>
  <obj name="profileattributes">
   ...
  </obj>
  <obj name="serverattributes">
   ...
  </obj>
 </obj>
 ...
</tta:response>
```

The following sections discuss the attribute group objects in more detail.

`applicationattributes` Object

An `applicationattributes` object is designated by the `<obj name="applicationattributes">` element. An `applicationattributes` object is an attributes group containing attributes that are set by the application.

An `applicationattributes` object contains the following default `<attr>` child elements.

Attribute Name	Description
`clientprintqueue`	Status of the user's print queue. For example, *paused* or *resumed*.
`preferredlanguage`	Preferred locale for the user session. Set using the `authenticate` operation.

Note – This object can contain other arbitrary attributes defined by the application.

The following example shows part of a `lookupSession` operation, where an `applicationattributes` object is returned.

```
 <tta:response>
  <obj name="webtopsession">
    ...
    <obj name="applicationattributes">
     <attr name="clientprintqueue">resumed</attr>
     <attr name="preferredlanguage">en</attr>
    </obj>
    ...
  </obj>
</tta:response>
```

clientattributes Object

A `clientattributes` object is designated by the `<obj name="clientattributes">` element. A `clientattributes` object is an attributes group containing attributes that are sent by the SGD Client.

A `clientattributes` object can contain the following `<attr>` child elements.

Attribute Name	Description
`tccarchitecture`	Architecture of the CPU on the client device.
`tcccode`	Where the SGD Client is installed on the client device.
`tcccolordepth`	Color depth of the display on the client device.
`tccinputlangms`	Locale setting for Windows client devices. A Microsoft Windows locale ID is used. For example, 2057 is UK English.
`tccinputlangrfc`	Locale setting for non-Windows client devices. A locale string is used. For example, en-GB is UK English.
`tcclogfile`	Location of the SGD Client log file on the client device.
`tccmaxoutofplaceheight`	Maximum usable screen height, in pixels, on the client device.
`tccmaxoutofplacewidth`	Maximum usable screen width, in pixels, on the client device.
`tccos`	Generic name for the operating system on the client device, for example, Windows.
`tccprintername`	Name of the default printer on the client device.
`tccscreenheight`	Maximum display height, in pixels, on the client device.
`tccscreenwidth`	Maximum display width, in pixels, on the client device
`tccscreenvirtualheight`	Virtual screen height on the client device.
`tccscreenvirtualwidth`	Virtual screen width on the client device.

Attribute Name	Description
tccscottasessiontccclientip	IP address of the client device on the local area network.
tcctimezonestd	Time zone setting on the client device.
tcctimezonestdoffset	Offset from Coordinated Universal Time (UTC) on the client device.
tcctimezonedst	Daylight saving time zone setting on the client device.

The returned attributes can vary, depending on the client OS.

The following example shows part of a `lookupSession` operation, where a `clientattributes` object is returned.

```
<tta:response>
 <obj name="webtopsession">
   ...
  <obj name="clientattributes">
   <attr name="tcctimezonestdoffset">0</attr>
   <attr name="tccscreenvirtualheight">1200</attr>
   <attr name="tccscreenvirtualwidth">1600</attr>
   <attr name="tccprintername">AGFA-AccuSet v52.3</attr>
   <attr name="tcctimezonestd">GMT Standard Time</attr>
   <attr name="tccscottasessiontccclientip">123.456.789.012</attr>
   <attr name="tccscreenheight">1200</attr>
   <attr name="tcclogfile">C:\Documents and Settings\auser\Local Settings\temp\
tcc\4.40.907\tcc.txt</attr>
   <attr name="tccscreenwidth">1600</attr>
   <attr name="tccmaxoutofplacewidth">1592</attr>
   <attr name="tccmaxoutofplaceheight">1166</attr>
   <attr name="tccinputlangms">2057</attr>
   <attr name="tccarchitecture">x86</attr>
   <attr name="tcccode">C:\Documents and Settings\auser\Local Settings\temp\
tcc\4.40.907cc.exe</attr>
   <attr name="tcctimezonedst">GMT Standard Time</attr>
   <attr name="tcccolordepth">32</attr>
   <attr name="tccos">Windows</attr>
  </obj>
   ...
 </obj>
</tta:response>
```

`connectionattributes` Object

A `connectionattributes` object is designated by the `<obj name="connectionattributes">` element. A `connectionattributes` object is an attributes group containing attributes for the connection between the SGD Client and the SGD server.

A `connectionattributes` object can contain the following `<attr>` child elements.

Attribute Name	Description
`baseroute`	Routing information to connect to the SGD server.
`baseurl`	Directory on the SGD server where the SGD Client and SGD Client Helper archives are located.
`connectionclientip`	Client IP address for the SGD Client connection.
`connheadersecurity`	Security level, standard or secure, for the connection between the SGD Client and the SGD server.
`firewalltraversal`	Whether the SGD server is running in firewall traversal mode.
`port`	Port number on the SGD server that the SGD Client connects to.
`security`	Security level, standard or secure, for the connection between the SGD Client and the SGD server. STD means only standard connections are available. SSL means standard and secure (SSL) connections are available.
`server`	DNS name for the SGD server that the SGD client connects to. For example, *server.example.com*.
`tccconnection`	Whether the user session has an SGD Client connection
`tcchelpertype`	Applet type for the SGD Client Helper. Only `java` is supported.
`tccport`	Port number that the SGD Client is listening on.
`useragent`	User-Agent identifier string for the browser on the client device.

The following example shows part of a `lookupSession` operation, where a `connectionattributes` object is returned.

```
<tta:response>
 <obj name="webtopsession">
   ...
  <obj name="connectionattributes">
   <attr name="tccconnection">true</attr>
   <attr name="useragent">Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)</attr>
   <attr name="port">5307</attr>
```

```
    <attr name="baseroute"></attr>
    <attr name="server">server.uk.sun.com</attr>
    <attr name="firewalltraversal">false</attr>
    <attr name="connheadersecurity">0</attr>
    <attr name="baseurl">http://server.example.com/sgd/tcc</attr>
    <attr name="tcchelpertype">java</attr>
    <attr name="connectionclientip">123.456.789.012</attr>
    <attr name="security">std</attr>
    <attr name="tccport">4272</attr>
    </obj>
    ...
   </obj>
  </tta:response>
```

`profileattributes` Object

A `profileattributes` object is designated by the `<obj name="profileattributes">` element. A `profileattributes` object is an attributes group containing attributes from the user's client profile.

A `profileattributes` object can contain the following `<attr>` child elements.

Attribute Name	Description
`autologin`	Log in to SGD automatically when the SGD Client starts up.
`autologin_state`	*For internal use only.*
`autostart`	Add the applications for an SGD user to their desktop Start or Launch menu on the client device. This is called *integrated mode*.
`editable`	Whether the client profile can be edited.
`httphost`	Server name used when configuring manual proxy settings for an HTTP proxy server.
`httpport`	Port number used when configuring manual proxy settings for an HTTP proxy server.
`lastproxyhost`	Proxy settings are cached by default. This is the proxy host from the cached settings.
`lastproxyport`	Proxy settings are cached by default. This is the proxy port from the cached settings.
`lastproxytype`	Proxy settings are cached by default. This is the proxy type, browser or manual, from the cached settings.
`logging`	Logging level for the SGD Client log file.
`mobile_client`	Establish proxy settings on user session start. Do not use cached proxy settings.

Attribute Name	Description
mode	Whether the SGD Client is used in browser mode or integrated mode.
objectclass	Parent object class for this object.
preferredlang	Default language used for dialogs and messages displayed by the SGD Client.
proxy	Whether to use browser proxy settings, or to use manual proxy settings.
reconnect_attempts	Number of reconnection attempts, if the connection to the SGD server is lost.
reconnect_interval	Interval, in seconds, between reconnection attempts, if the connection to the SGD server is lost.
reconnect_mode	Reconnection mode, if the connection to the SGD server is lost. Whether to always reconnect, never reconnect, or to prompt the user.
sessioncookie	*For internal use only.*
trylocalx	Checks if there is an X server running on the client device. Only applies to Windows client devices.
url	URL used to log in to SGD, for example, `http://`*server.example.com*`/sgd`, where *server.example.com* is an SGD server.
usehttp	Whether to configure manual proxy settings. You can only specify an HTTP proxy server.
usesocks	Whether to configure SOCKS proxy settings. *This attribute is deprecated.*

The number of returned attributes can vary, depending on the settings made in the client profile.

The following example shows part of a `lookupSession` operation, where a `profileattributes` object is returned.

```
<tta:response>
 <obj name="webtopsession">
   ...
   <obj name="profileattributes">
    <obj name="Default">
     <attr name="sessioncookie"></attr>
     <attr name="lastproxytype"></attr>
     <attr name="usesocks">false</attr>
     <attr name="objectclass">sunsgdprofile</attr>
     <attr name="url">http://server.example.com/sgd/index.jsp</attr>
     <attr name="autologin">false</attr>
```

```
      <attr name="lastproxyport">0</attr>
      <attr name="preferredlang">en</attr>
      <attr name="autologin_state">enabled</attr>
      <attr name="logging">logging_errorsonly</attr>
      <attr name="lastproxyhost"></attr>
      <attr name="reconnect_attempts">6</attr>
      <attr name="mode">integrated</attr>
      <attr name="reconnect_interval">10</attr>
      <attr name="usehttp">false</attr>
      <attr name="trylocalx">false</attr>
      <attr name="mobile_client">false</attr>
      <attr name="proxy">browser</attr>
      <attr name="reconnect_mode">reconnect_always</attr>
      <attr name="autostart">false</attr>
     </obj>
     <attr name="editable">false</attr>
    </obj>
    ...
   </obj>
  </tta:response>
```

serverattributes Object

A serverattributes object is designated by the <obj name="serverattributes"> element. A serverattributes object is an attributes group containing attributes that are set by the SGD server.

A serverattributes object can contain the following <attr> child elements.

Attribute Name	Description
objectclass	Parent object class for this object.
scottaauthoritativesource	DNS name of the SGD server where the serverattributes information was obtained.
scottaclientprintstate	Status of the client print queue, for example, *paused* or *resumed*.
scottadeviceaccessibledata	Configuration data for client devices on the SGD server, for example, serial ports, printers, and external drives.
scottasessionexternaldnsname	External DNS name of the SGD server hosting the user session. An external DNS name is the DNS name that client devices use to access the SGD server, for example, www.server.example.com.

Attribute Name	Description
`scottasessionid`	Public identifier for the user session. Can be used by SGD Administrators to perform operations on other people's user sessions.
`scottasessionlogintime`	When the user last logged in to SGD.
`scottasessionowner`	User identity that owns this user session.
`scottasessionpeerdnsname`	A peer DNS name is the DNS name that other SGD servers in the array use to identify themselves and each other. For example, `server1.example.com`.
`scottasessionsecurity`	Whether or not security services have been enabled on the SGD server. STD means only standard connections are available. SSL means standard and secure (SSL) connections are available.
`scottasessiontemplateowner`	User profile that owns this user session. The user profile defines the assigned applications for an SGD user.
`scottasessiontoken`	*This attribute is deprecated.*
`scottawebtopsessionemulatorsecurity`	Security level, standard or secure, for the connection between the SGD server and the application server. STD means only standard connections are available. SSL means standard and secure (SSL) connections are available.
`scottawebtopstate`	Whether there is an SGD Client connection between the client device and the SGD server.

The number of returned attributes can vary, depending on the authentication mechanism used and whether the SGD Client is running on the client device.

The following example shows part of a `lookupSession` operation, where a `serverattributes` object is returned.

```
<tta:response>
 <obj name="webtopsession">
   ...
   <obj name="serverattributes">
    <attr name="scottasessionsecurity">STD</attr>
    <attr name="scottasessionlogintime">1194968629717</attr>
    <attr name="scottasessiontoken">.../_token/DqCkentBbF+VL9kXa0paNhgWY</attr>
    <attr name="scottaauthoritativesource">server.example.com</attr>
    <attr name="objectclass">top</attr>
    <attr name="objectclass">scottasession</attr>
    <attr name="objectclass">scottawebtopsession</attr>
    <attr name="scottawebtopstate">connected</attr>
    <attr name="scottasessionpeerdnsname">server.uk.sun.com</attr>
    <attr name="scottawebtopsessionemulatorsecurity">STD</attr>
    <attr name="scottadeviceaccessibledata">AUDIO:...</attr>
    <attr name="scottadeviceaccessibledata">IOC:...</attr>
    <attr name="scottadeviceaccessibledata">LPD:...</attr>
    <attr name=
"scottasessionid">server.example.com:1194968629717:7854436686215328242</attr>
    <attr name="scottasessiontemplateowner">.../_ens/o=organization/cn=
 multi</attr>
    <attr name=
 "scottasessionowner">.../_dns/server.example.com/_anon/1194883433663</attr>
    <attr name="scottaclientprintstate">1</attr>
    <attr name="scottasessionexternaldnsname">server.example.com</attr>
   </obj>
   ...
  </obj>
</tta:response>
```

Operations

The following table lists the available operations for the `ITarantellaWebtopSession` web service.

Operation Name	Purpose
`adminCount`	Counts the number of matching user sessions that a search will return. For use by SGD Administrators only.
`adminEndSession`	Ends the specified user session. For use by SGD Administrators only.
`adminEndSessions`	Ends the specified user sessions. For use by SGD Administrators only.
`adminGetSessionPEInfo`	Gets the latest protocol engine activity data for a user session. For use by SGD Administrators only.
`adminLookupSession`	Looks up data for a user session. For use by SGD Administrators only.
`adminSearchEnd`	Releases server resources for a given search. For use by SGD Administrators only.

Operation Name	Purpose
adminSearchNext	Retrieves the next subset of search results. For use by SGD Administrators only.
adminSearchSession	Searches for user sessions. For use by SGD Administrators only.
adminSearchStart	Starts a search for user sessions, returning a maximum number of results. For use by SGD Administrators only.
associateTCC	Associates a user session view with an existing SGD Client connection.
authenticate	Authenticates and creates a new user session.
authenticateExt	Authenticates and creates a new user session, providing additional information to identify the user.
createView	Creates a new view on an existing user session.
endSession	Ends a user session.
getSessionCookie	Returns an identifier for a user session, based on a supplied session token.
getSessionToken	Generates a session token that can be used to look up the session cookie.
lookupSession	Looks up data for a user session.
modifySession	Modifies application attribute values for a user session.
refreshClientInfo	Requests an update of client information from the SGD Client.
refreshClientInfoEx	Requests an update of a subset of the client information from the SGD Client.
restart	Requests that a user session is linked to an SGD Client connection after an SGD server restart.
setTCCConfiguration	Sets the configuration needed to determine the SGD Client to use for a user session.
setTCCConfigurationOverrides	Sets the configuration needed to override SGD Client settings.
startTCC	Gets the information needed to start the SGD Client.

adminCount

Counts the number of matching user sessions for a given search filter.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminCount>
 <sessioncookie>xsd:string</sessioncookie>
 <searchspec>xsd:string</searchspec>
</adminCount>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="total">xsd:string</attr>
</tta:response>
```

`adminEndSession`

End another person's user session. The user session to end is specified by its session ID attribute.

If the `<logout>` element is `true`, the user is logged out of SGD and their applications are closed down.

If the `<logout>` element is `false`, a user's applications are not closed down. Users can log in again and resume applications that are configured to be resumable during the user session, or to be always resumable.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminEndSession>
 <sessioncookie>xsd:string</sessioncookie>
 <sessionid>xsd:string</sessionid>
 <logout>xsd:boolean</logout>
</adminEndSession>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

adminEndSessions

End one or more user sessions. The user sessions to end are specified by their session ID attributes.

If the `<logout>` element is `true`, the users are logged out of SGD and their applications are closed down.

If the `<logout>` element is `false`, users' applications are not closed down. Users can log in again and resume applications that are configured to be resumable during the user session, or to be always resumable.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminEndSessions>
 <sessioncookie>xsd:string</sessioncookie>
 <sessionids>xsd:string</sessionids> [1..unbounded]
 <logout>xsd:boolean</logout>
</adminEndSessions>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

adminGetSessionPEInfo

Gets the latest protocol engine (PE) activity data for the user session specified by `<sessionid>` element. The operation returns the time period, in seconds, since the most recent activity for the following PEs:

- Client drive mapping
- Printing

- Emulator session (mouse or keyboard activity)

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminGetSessionPEInfo>
 <sessioncookie>xsd:string</sessioncookie>
 <sessionid>xsd:string</sessionid>
</adminGetSessionPEInfo>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="scottaemulatorsessionlastactivity">xsd:string</attr>
 <attr name="scottacdmlastactivity">xsd:string</attr>
 <attr name="scottaprintlastactivity">xsd:string</attr>
</tta:response>
```

`adminLookupSession`

Looks up attributes for another person's user session.

The attributes to return are specified by the `<desattrs>` element.

Attributes are returned in their respective attribute group. See "Using Shortcuts to Return Groups of Attributes" on page 129.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminLookupSession>
 <sessioncookie>xsd:string</sessioncookie>
 <sessionid>xsd:string</sessionid>
 <desattrs>xsd:string</desattrs> [1..unbounded]
</adminLookupSession>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <obj name="webtopsession">
  <obj name="clientattributes"> [0..1]
  <obj name="connectionattributes"> [0..1]
  <obj name="applicationattributes"> [0..1]
  <obj name="profileattributes"> [0..1]
  <obj name="serverattributes"> [0..1]
 </obj>
</tta:response>
```

adminSearchEnd

Used with the adminSearchStart and adminSearchNext operations to perform searches for user sessions.

The adminSearchEnd operation releases server resources for a given search, identified by the <searchid> element. Use this operation if you have finished with a search, but not all of the results have been retrieved.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminSearchEnd>
 <sessioncookie>xsd:string</sessioncookie>
 <searchid>xsd:string</searchid>
</adminSearchEnd>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

adminSearchNext

Used with the adminSearchStart and adminSearchEnd operations to perform searches for user sessions.

The adminSearchNext operation returns the next subset of search results for a given search, identified by the <searchid> element. The <noofresults> element defines the maximum number of results to return from the search.

Search results are grouped by SGD server, designated by the <src> element.

Use the adminSearchEnd operation if you have finished with a search, but not all of the results have been retrieved.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminSearchNext>
 <sessioncookie>xsd:string</sessioncookie>
 <searchid>xsd:string</searchid>
 <noofresults>xsd:int</noofresults>
</adminSearchNext>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <src name="server.example.com"> [1..unbounded]
  <obj name="webtopsession"> [0..unbounded]
 </src>
 <attr name="searchid">xsd:string</attr>
</tta:response>
```

adminSearchSession

Performs a search for user sessions.

This operation differs from the adminSearchStart operation in that *all* search results are returned in the response. Also, a <searchid> element is not returned.

Search results are grouped by SGD server, designated by the <src> element.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminSearchSession>
 <sessioncookie>xsd:string</sessioncookie>
 <searchspec>xsd:string</searchspec>
 <desattrs>xsd:string</desattrs> [1..unbounded]
</adminSearchSession>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <src name="server.example.com"> [1..unbounded]
  <obj name="webtopsession"> [0..unbounded]
 </src>
</tta:response>
```

adminSearchStart

Used with the `adminSearchNext` and `adminSearchEnd` operations to perform searches for user sessions.

The `adminSearchStart` operation starts a user session search, returning a subset of the results, specified by the `<noofresults>` element.

If the search returns more results than `<noofresults>`, the search is cached on the SGD server. Subsequent results can be returned using the `adminSearchNext` operation.

Search results are grouped by SGD server, designated by the `<src>` element.

Use the `adminSearchEnd` operation if you have finished with a search, but not all of the results have been retrieved.

This operation is for use by SGD Administrators only.

Request Message

The format of the request message for this operation is as follows:

```
<adminSearchStart>
 <sessioncookie>xsd:string</sessioncookie>
 <searchspec>xsd:string</searchspec>
 <desattrs>xsd:string</desattrs> [1..unbounded]
 <noofresults>xsd:int</noofresults>
 <searchtype>xsd:string</searchtype>
</adminSearchStart>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <src name="server.example.com"> [1..unbounded]
  <obj name="webtopsession"> [0..unbounded]
 </src>
 <attr name="searchid">xsd:string</attr>
</tta:response>
```

associateTCC

Associates a user session view with an existing SGD Client connection.

This operation enables an SGD Client connection to be used by a different view to the one that started it. The `<id>` element identifies the view to associate with the SGD Client connection.

Request Message

The format of the request message for this operation is as follows:

```
<associateTCC>
 <sessioncookie>xsd:string</sessioncookie>
 <id>xsd:string</id>
</associateTCC>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

authenticate

Authenticates and creates a user session.

If authentication is successful, this operation returns a session cookie attribute, `<scottasessioncookie>`, that uniquely defines the user session.

A `view` object is returned if the new session joins an existing user session. The view number is specified by the `viewid` attribute. See "Session Moving and Session Joining" on page 113.

Request Message

The format of the request message for this operation is as follows:

```
<authenticate>
 <username>xsd:string</username>
 <passwd>xsd:string</passwd>
 <clientId>xsd:string</clientId>
 <locale>xsd:string</locale>
</authenticate>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="scottasessioncookie">xsd:string</attr>
 <attr name="scottatheme">xsd:string</attr>
 <attr name="isadministrator">xsd:string</attr>
 <attr name="cn">xsd:string</attr>
 <attr name="arrayMember">xsd:string</attr>
 <obj name="serverattributes">
   <attr name="scottasessionid">xsd:string</attr>
   <attr name="scottasessiontemplateowner">xsd:string</attr>
   <attr name="scottasessionowner">xsd:string</attr>
 </obj>
 <obj name="views"> [0..1]
  <obj name="view"> [1..unbounded]
   <attr name="viewid">xsd:string</attr>
   <attr name="tccconnection">xsd:boolean</attr>
  </obj>
 </obj>
</tta:response>
```

authenticateExt

Authenticates and creates a user session. Only use this operation if challenged for additional credentials by the login mechanism when using `authenticate`.

This operation differs from the `authenticate` operation in that additional credentials can be supplied to identify the user. For example, an email address can be supplied for an ambiguous login, or additional credentials can be supplied for multi-stage authentication.

If authentication is successful, this operation returns a session cookie attribute, `scottasessioncookie`, that uniquely defines the user session.

A `view` object is returned if the new session joins an existing user session. The view number is specified by the `viewid` attribute. See "Session Moving and Session Joining" on page 113.

Request Message

The format of the request message for this operation is as follows:

```
<authenticateExt>
 <username>xsd:string</username>
 <password>xsd:string</password>
 <clientId>xsd:string</clientId>
 <locale>xsd:string</locale>
 <creds>tns1:Item</creds> [1..unbounded]
 <feature>tns1:Item</feature> [1..unbounded]
</authenticateExt>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="scottasessioncookie">xsd:string</attr>
 <attr name="scottatheme">xsd:string</attr>
 <attr name="isadministrator">xsd:string</attr>
 <attr name="cn">xsd:string</attr>
 <attr name="arrayMember">xsd:string</attr>
 <obj name="serverattributes">
   <attr name="scottasessionid">xsd:string</attr>
   <attr name="scottasessiontemplateowner">xsd:string</attr>
   <attr name="scottasessionowner">xsd:string</attr>
 </obj>
 <obj name="views"> [0..1]
  <obj name="view"> [1..unbounded]
   <attr name="viewid">xsd:string</attr>
   <attr name="tccconnection">xsd:boolean</attr>
  </obj>
 </obj>
</tta:response>
```

createView

Creates a new view on an existing user session.

A view is created when a user session *joins* an existing session, as opposed to *moving* the session. See "Session Moving and Session Joining" on page 113 for more details about session joining.

A `view` object is returned for the new view. The view number is specified by the `viewid` attribute.

The `<endview>` element determines whether the view is ended when the user session is logged out. If `<endview>` is `false`, the user session view is not ended unless the user session is logged out from this view.

Request Message

The format of the request message for this operation is as follows:

```
<createView>
 <sessioncookie>xsd:string</sessioncookie>
 <endview>xsd:boolean</endview>
</createView>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="scottasessioncookie">xsd:string</attr>
 <attr name="scottatheme">xsd:string</attr>
 <attr name="isadministrator">xsd:string</attr>
 <attr name="cn">xsd:string</attr>
 <attr name="arrayMember">xsd:string</attr>
 <obj name="serverattributes">
   <attr name="scottasessionid">xsd:string</attr>
   <attr name="scottasessiontemplateowner">xsd:string</attr>
   <attr name="scottasessionowner">xsd:string</attr>
 </obj>
 <obj name="views"> [0..1]
  <obj name="view"> [1..unbounded]
   <attr name="viewid">xsd:string</attr>
   <attr name="tccconnection">xsd:boolean</attr>
  </obj>
 </obj>
</tta:response>
```

endSession

Ends the user session specified by the sessioncookie parameter.

If the `<logout>` element is `false`, users can log in again and resume applications that have an Application Resumability attribute value of `session` or `always`.

Request Message

The format of the request message for this operation is as follows:

```
<endSession>
 <sessioncookie>xsd:string</sessioncookie>
 <logout>xsd:boolean</logout>
</endSession>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

getSessionCookie

Returns the session cookie for a user session, based on a supplied session token.

You can use the `getSessionToken` operation to generate a session token for a session cookie. See "Controlling Access to SGD Web Services" on page 112 for more details.

Note – A session token can only be used *once* to look up the session cookie with this operation. Subsequent `getSessionCookie` operations using the same session token return an error.

Request Message

The format of the request message for this operation is as follows:

```
<getSessionCookie>
 <sessiontoken>xsd:string</sessiontoken>
</getSessionCookie>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="scottasessioncookie">xsd:string</attr>
</tta:response>
```

getSessionToken

Generates a session token for a user session. The session token can be used in the `getSessionCookie` operation to look up the session cookie for the user session.

This operation can be used to avoid exposing the session cookie value. See "Controlling Access to SGD Web Services" on page 112 for more details.

Note – A session token can only be used *once* to look up the session cookie with the `getSessionCookie` operation. Subsequent `getSessionCookie` operations using the same session token return an error.

Request Message

The format of the request message for this operation is as follows:

```
<getSessionToken>
 <sessioncookie>xsd:string</sessioncookie>
</getSessionToken>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="sessiontoken">xsd:string</attr>
</tta:response>
```

lookupSession

Looks up the attributes for a user session.

The attributes to return are specified by the `<desattrs>` element.

Attributes are returned in their respective attribute group. See "Using Shortcuts to Return Groups of Attributes" on page 129.

SGD Administrators can use `adminLookupSession` to look up other user's sessions.

Request Message

The format of the request message for this operation is as follows:

```
<lookupSession>
 <sessioncookie>xsd:string</sessioncookie>
 <desattrs>xsd:string</desattrs> [1..unbounded]
</lookupSession>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <obj name="webtopsession">
  <obj name="clientattributes"> [0..1]
  <obj name="connectionattributes"> [0..1]
  <obj name="applicationattributes"> [0..1]
  <obj name="profileattributes"> [0..1]
  <obj name="serverattributes"> [0..1]
 </obj>
</tta:response>
```

modifySession

Modifies the application attributes for a user session.

The <mode> element determines how the attribute is modified. See "Modifying a User Session" on page 125 for details of the suppported modes.

You can only use this operation to change the attributes stored in the applicationattributes attributes group. All other attribute groups are read-only.

Request Message

The format of the request message for this operation is as follows:

```
<modifySession>
 <sessioncookie>xsd:string</sessioncookie>
 <data>tns1:Item</data> [1..unbounded]
 <mode>xsd:string</mode>
</modifySession>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <obj name="applicationattributes"> [0..1]
</tta:response>
```

refreshClientInfo

Requests an update of all client device attributes from the SGD Client.

The SGD Client queries the client device and sends the current settings to the SGD server, where they are stored in the clientattributes attributes group for the user session. See "clientattributes Object" on page 135.

Request Message

The format of the request message for this operation is as follows:

```
<refreshClientInfo>
 <sessioncookie>xsd:string</sessioncookie>
</refreshClientInfo>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="tcctimezonestdoffset">xsd:string</attr>
 <attr name="tccscreenvirtualheight">xsd:string</attr>
 <attr name="tccscreenvirtualwidth">xsd:string</attr>
 <attr name="tccprintername">xsd:string</attr>
 <attr name="tcctimezonestd">xsd:string</attr>
 <attr name="tccscottasessiontccclientip">xsd:string</attr>
 <attr name="tccinputlangrfc">xsd:string</attr>
 <attr name="tccscreenheight">xsd:string</attr>
 <attr name="tccmaxoutofplacewidth">xsd:string</attr>
 <attr name="tccscreenwidth">xsd:string</attr>
 <attr name="tcclogfile">xsd:string</attr>
 <attr name="tccmaxoutofplaceheight">xsd:string</attr>
 <attr name="tccarchitecture">xsd:string</attr>
 <attr name="tcccode">xsd:string</attr>
 <attr name="tcccolordepth">xsd:string</attr>
 <attr name="tcctimezonedst">xsd:string</attr>
</tta:response>
```

`refreshClientInfoEx`

Requests an update of specified client device attributes from the SGD Client. The attributes to return are specified by the `<clientData>` element.

The SGD Client queries the client device and sends the current settings to the SGD server, where they are stored in the `clientattributes` attributes group for the user session. See "`clientattributes` Object" on page 135.

Request Message

The format of the request message for this operation is as follows:

```
<refreshClientInfoEx>
 <sessioncookie>xsd:string</sessioncookie>
 <clientData>xsd:string</clientData> [1..unbounded]
</refreshClientInfoEx>
```

Response Message

The format of the response message for this operation is similar to the response message for the `refreshClientInfo` operation, except that only the requested attribute values are returned.

`restart`

Requests that a user session is linked to an SGD Client connection, following an SGD server restart.

Following an SGD server restart, the SGD server includes a connection cookie in the restart event sent to the SGD Client. In the restart event handler, specify this cookie value in a `restart` operation if you want to use the existing SGD Client connection with a new user session.

Request Message

The format of the request message for this operation is as follows:

```
<restart>
 <sessioncookie>xsd:string</sessioncookie>
 <token>xsd:string</token>
</restart>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

`setTCCConfiguration`

Sets the required configuration for the SGD Client and the SGD Client Helper, if used. See "Specifying a SGD Client to Download" on page 118.

The `<userAgent>` and `<type>` elements determine the type of SGD Client Helper written to the HTML.

The `<userAgent>` element is used to select the correct SGD Client binary for the operating system (OS) on the client device.

The `<baseurl>` element is used to determine the download directory on the SGD server.

The `<version>` element is the SGD Client version number. This is the same as the SGD software version.

Request Message

The format of the request message for this operation is as follows:

```
<setTCCConfiguration>
 <sessioncookie>xsd:string</sessioncookie>
 <useragent>xsd:string</useragent>
 <baseurl>xsd:string</baseurl>
 <version>xsd:string</version>
 <type>xsd:string</type>
</setTCCConfiguration>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

`setTCCConfigurationOverrides`

Sets the required configuration for the SGD Client and the SGD Client Helper, if used.

This operation differs from `setTCCConfiguration` in that it enables you to override some of the attribute settings for an SGD Client connection. The `setTCCConfigurationOverrides` operation is useful when the connection between the SGD Client and the SGD server is not direct, for example, when connecting through a gateway.

The supported attribute overrides are listed in "Specifying Additional Connection Settings" on page 119.

Request Message

The format of the request message for this operation is as follows:

```
<setTCCConfigurationOverrides>
 <sessioncookie>xsd:string</sessioncookie>
 <useragent>xsd:string</useragent>
 <baseurl>xsd:string</baseurl>
 <version>xsd:string</version>
 <type>xsd:string</type>
 <overrides>tns1:Item</overrides> [1..unbounded]
</setTCCConfigurationOverrides>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response/>
```

startTCC

Gets the information required to start the SGD Client. Optionally, a message can be stored on the SGD server that is sent to the SGD Client when it connects.

A web application accessed using a browser with Java technology enabled can return configuration data to the browser, using the `codefragment` attribute. The SGD Helper applet then downloads and starts the SGD Client.

An application that does not use a browser with Java technology enabled can construct a suitable command line using other attributes from the response. See "Starting the SGD Client" on page 120 for more details.

Request Message

The format of the request message for this operation is as follows:

```
<startTCC>
 <sessioncookie>xsd:string</sessioncookie>
 <helperid>xsd:string</helperid>
 <url>xsd:string</url>
 <target>xsd:string</target>
</startTCC>
```

Response Message

The format of the response message for this operation is as follows:

```
<tta:response>
 <attr name="helperid">xsd:string</attr>
 <attr name="connectioncookie">xsd:string</attr>
 <attr name="security">xsd:string</attr>
 <attr name="server">xsd:string</attr>
 <attr name="port">xsd:string</attr>
 <attr name="version">xsd:string</attr>
 <attr name="baseroute">xsd:string</attr>
 <attr name="tccport">xsd:string</attr>
 <attr name="tccportfile">xsd:string</attr>
 <attr name="firewalltraversal">xsd:string</attr>
 <attr name="codefragment">xsd:string</attr>
</tta:response>
```

APPENDIX **A**

SOAP Error Messages

This chapter lists the SOAP fault codes which can be generated by SGD web services.

SOAP errors are returned in the XML response string.

The following example shows a SOAP fault response for a `ITarantellaPrint.cancelJobs` operation that has failed.

```
<soapenv:Body>
 <soapenv:Fault>
  <faultcode>soapenv:Client.JobsNotFound</faultcode>
  <faultstring>Requested print jobs not found</faultstring>
  <detail>
   <string>
    <attr name="jobid">test01</attr>
   </string>
  </detail>
 </soapenv:Fault>
</soapenv:Body>
```

This chapter includes the following topics:

SOAP Fault Codes Table

SGD web services support the following SOAP fault messages.

Fault String	Description
Client.AmbiguousApplication	The application session launch failed because the application name is ambiguous.
Client.AmbiguousLogin	The login credentials are ambiguous.
Client.DoesNotExist	The object does not exist.
Client.EventDeregistration	Event listener deregistration failed.
Client.EventLimit	The maximum number of event listeners have already been registered on the user session.
Client.InvalidAttribute	The request contains one or more invalid attributes.
Client.InvalidSessionCookie	The user session cookie is invalid.
Client.InvalidUrnName	The name is not a valid web service namespace.
Client.JobsNotFound	One or mode requested print jobs were not found.
Client.LoginFailed	The authentication of the user session has failed.
Client.LoginVetoed	The user is already logged in and the login attempt is vetoed.
Client.MandatoryArgsMissing	Mandatory command-line parameters are missing.
Client.MandatoryDataMissing	Mandatory command-line data are missing.
Client.MissingAttribute	The request is missing an attribute.
Client.NoAuthentication	The user session is not authenticated.
Client.NoAccess	The session does not have sufficient privileges to access the datastore.
Client.NoClientPrinter	A job could not be printed as a client printer has not been identified.
Client.NoClientSessionObject	The request requires data that is only available on the array node where the user session was created.
Client.NoCommandMatch	There is no match for the command-line command.
Client.NoFullyQualifedName	The login attempt has generated an internal error.
Client.NoMatch	No application matches the abbreviated name supplied in the request.
Client.NoUserContext	The login attempt has generated an internal error.

Fault String	Description
Client.ObjectExists	The object already exists.
Client.ObjectNotEmpty	The container object is not empty.
Client.PasswordAged	The second-tier password has expired.
Client.PermissionDenied	The request is denied as the user has insufficient privileges.
Client.ReduceCpu	Connections are being downgraded from secure to standard to reduce CPU usage.
Client.SecurityRequired	A secure connection is required before logins are accepted.
Client.UnknownLogin	The login attempt has generated an internal error.
Server.AlreadyAuthenticated	The user session is already authenticated.
Server.AlreadyRegistered	The event listener is already registered.
Server.AnonDisabled	Anonymous logins to the server are disabled.
Server.CannotDo	The request could not be performed.
Server.EvalExpired	The evaluation period has expired.
Server.Exception	An exception has been thrown in the server.
Server.LaunchFailed	The application session launch has failed.
Server.LogoutFailed	Logging-out has failed.
Server.MaxSessions	The maximum number of unauthenticated sessions has been reached.
Server.NoDefinition	A type has been used that has not been defined in the web service descriptor.
Server.NoEventAdaptor	The event/messaging sub-system has not been started.
Server.NoLicense	There is no license for the operation.
Server.NoOperationMatch	An operation has been used that has not been defined in the web service descriptor.
Server.NoResume	Resumption of the application session failed.
Server.NoSkid	There is no authentication between server components.
Server.NoTCCResponse	No response has been received from the client component.
Server.NoWrite	The server was unable to send data to the client component.
Server.ObjectAlreadyExists	The object already exists.

Fault String	Description
Server.ObjectDoesntExist	Object does not exist.
Server.RegistrationFailed	The event listener registration failed.
Server.ServerDisabled	The server is disabled and not accepting connections.
Server.UserException	The web server is unable to connect to the server.

APPENDIX **B**

Object Attribute Schema Reference

SGD represents users, resources, and organizational structure as *objects* in a directory. SGD objects have configuration settings, known as *attributes*. This chapter lists the supported attributes for SGD objects.

This chapter includes the following topics:

SGD Object Types

The object type for an SGD object is determined by its `objectclass` attribute. The following table lists the object types you can create in the SGD object hierarchy. The table shows the name used for the object in the SGD Administration Console, as well as the corresponding object class for the object.

Administration Console	**Object Class**
3270 application	`scotta3270application`
5250 application	`scotta5250application`
Active Directory container	`container`
Application server	`scottaauxhost`
Character application	`scottacharacterapplication`
Document	`scottahtmldocument`
Domain component	`domainDNS`
Group	`scottagroupofnames`

Administration Console	Object Class
Organization	scottaauxorganization
Organizational unit	scottaauxorganizationalunit
User profile	scottaauxperson
Windows application	scottawindowsapplication
X application	scottaxapplication

Attribute Names for SGD Objects

The names for SGD object attributes can vary, depending on where they are being used, as follows:

- **Administration Console.** Attribute names are shown in the SGD Administration Console.
- **Command line.** Attribute names can be specified on the command line, when using the `tarantella` command.
- **Schema name.** The names of all SGD object attributes are defined in an LDAP object schema.

The following table lists the supported attributes for objects in SGD. See the *Sun Secure Global Desktop Reference Manual* for more information about each attribute, including which SGD object types they are used for.

Administration Console	Command Line	Schema Name
Address	--address	scottadnsname
Answerback Message	--answermsg	scottaanswerbackmessage
Application Command	--app	scottafilepath
Application Load Balancing	--loadbal	scottatier3loadbalancingalgorithm
Application Resumability	--resumable	scottasuspend
Application Resumability: Timeout	--resumetimeout	scottaresumetimeout
Application Start	--available	scottahostavailable
Arguments for Command	--args	scottaarguments
Arguments for Protocol	--protoargs	scottaprotocolarguments
Assigned Applications Tab	--links	scottawebtopcontents

Administration Console	Command Line	Schema Name

Assigned User Profiles Tab	`--ldapusers`
	`--ldapgroups`
	`--ldapsearch`
		`scottamemberusers`
		`scottamembergroups`
		`scottamembersearch`

Attribute Map	`--attributemap`	`scottaattributemap`
Audio Redirection Library	`--unixaudiopreload`	`scottaunixaudiopreload`
Background Color	`--3270bg` `--bg`	`scotta3270backgroundcolor` `scotta5250backgroundcolor`
Bandwidth Limit	`--bandwidth`	`scottaaipbandwidth`
Border Style	`--border`	`scottaappearance`
Client Drive Mapping	`--cdm`	`scottadeviceconfig`
Client Printing	`--mapprinters`	`scottamapclientprinters`
Client Printing: Override	`--userprintingconfig`	`scottauserprintingconfig`
Client Profile Editing	`--editprofile`	`sunsgdeditprofile`
Code Page	`--codepage`	`scottacodepage`
Color Depth	`--depth`	`scottacolordepth`
Color Map	`--colormap`	`scottacolormap`
Color Quality	`--quality`	`scottacolorquality`
Command Compression	`--compression`	`scottacompression`
Command Execution	`--execution`	`scottacontinuousmode`
Comment	`--description`	`description`
Connection Closed Action	`--3270tn` `--tn`	`scotta3270tnclose` `scotta5250tnclose`
Connection Method	`--method`	`scottatransport`
Connection Method: ssh Arguments	`--ssharguments`	`sunsgdssharguments`
Connections	`--conntype`	`scottasecureconnection`
Copy and Paste	`--clipboard`	`sunsgdclipboardenabled`
Copy and Paste: Application’s Clipboard Security Level	`--clipboardlevel`	`sunsgdclipboardlevel`
Cursor	`--cursor`	`scottacursor`
Cursor Key Codes Modification	`--cursorkeys`	`scottacursorkeymode`
Delayed Updates	`--delayed`	`scottadelayedupdate`
Displayed Soft Buttons	`--3270bl` `--bl`	`scotta3270buttonlevels` `scotta5250buttonlevels`

Administration Console	Command Line	Schema Name
Domain Name	`--ntdomain`	`scottantdomain`
Email Address	`--email`	`mail`
Emulation Type	`--emulator`	`scottaemulator`
Environment Variables	`--env`	`scottaenvironment`
Escape Sequences	`--escape`	`scottacontrolcode`
Euro Character	`--euro`	`scottaeurokeymapping`
'File' and 'Settings' Menus	`--3270si` `--si`	`scotta3270settingsitem` `scotta5250settingsitem`
Font Family	`--font`	`scottafontfamily`
Font Size	`--fontsize`	`scottafontsize`
Font Size: Fixed Font Size	`--fixedfont`	`scottafixedfontsize`
Foreground Color	`--3270fg` `--fg`	`scotta3270foregroundcolor` `scotta5250foregroundcolor`
Graphics Acceleration	`--accel`	`scottagraphicsacceleration`
Hints	`--hints`	`scottahints`
Hosting Application Servers Tab	`--appserv`	`scottahosts`
Icon	`--icon`	`scottaicon`
Inherit Assigned Applications From Parent	`--inherit`	`scottainheritwebtop`
Interlaced Images	`--interlaced`	`scottainterlacedimages`
Keep Launch Connection Open	`--keepopen`	`scottaviewhostreply`
Keyboard Codes Modification	`--appkeymode`	`scottaapplicationkeymode`
Keyboard Map	`--keymap`	`scottakeyboardmap`
Keyboard Map: Locked	`--lockkeymap`	`scottakeymaplock`
Keyboard Type	`--3270kt` `--kt`	`scotta3270keyboardtype` `scotta5250keyboardtype`
Line Wrapping	`--autowrap`	`scottaautowrap`
Load Balancing Groups	`--location`	`l`
Login	`--enabled`	`scottaaccountenabled`
Login: Multiple	`--shared`	`scottasharedlogin`
Login Name	`--user`	`uid`
Login Script	`--login`	`scottaloginscript`

Administration Console	Command Line	Schema Name
Make Universal PDF Printer the Default	`--pdfisdefault`	`scottapdfisdefault`
Make Universal PDF Viewer the Default	`--pdfviewerisdefault`	`scottapdfviewerisdefault`
Members Tab	`--member`	`member`
Menu Bar	`--3270mb` `--mb`	`scotta3270topmenubar` `scotta5250topmenubar`
Middle Mouse Timeout	`--middlemouse`	`scottamiddlemousetimeout`
Monitor Resolution	`--dpi`	`scottamonitorresolution`
Mouse	`--force3button`	`scottaforce3buttonmouse`
Name	`--name`	`cn` `dc` (for Domain Component objects) `o` (for Organization objects) `ou` (for OU objects)
Number of Sessions	`--maxinstances`	`scottamaxinstances`
Numpad Codes Modification	`--keypad`	`scottakeypadmode`
Password Cache Usage	`--auth`	`scottatryttapassword`
Postscript Printer Driver	`--pdfdriver`	`scottapdfdriver`
Prompt Locale	`--hostlocale`	`scottahostlocale`
Scroll Style	`--scrollstyle`	`scottascrollstyle`
Serial Port Mapping	`--serialport`	`sunsgdserialportenabled`
Server Address	`--hostname`	`scottahostname`
Server Port	`--portnumber`	`scottaportnumber`
Session Termination	`--endswhen`	`scottasessionexit`
Share Resources Between Similar Sessions	`--share`	`scottashareresources`
Status Line	`--statusline`	`scottastatusline`
Surname	`--surname`	`sn`
Terminal Type	`--termtype`	`scottaterminaltype`
Universal PDF Printer	`--pdfenabled`	`scottapdfenabled`
Universal PDF Viewer	`--pdfviewerenabled`	`scottapdfviewerenabled`
URL	`--url`	`scottalocation`
Window Close Action	`--windowclose`	`scottawindowcloseaction`

Administration Console	Command Line	Schema Name
Window Color	`--roottype`	`scottaroottype`
Window Color: Custom Color	`--rootcolor`	`scottarootcolor`
Window Management Keys	`--remotewindowkeys`	`sunsgdremotewindowkeys`
Window Manager	`--winmgr`	`scottaauxiliaryapps`
Window Size: Client's Maximum Size	`--maximize`	`scottafullscreen`
Window Size: Columns	`--cols`	`scottaterminaltype`
Window Size: Height	`--height`	`scottaappletheight`
Window Size: Lines	`--lines`	`scottalines`
Window Size: Maximized	`--3270ma` `--ma`	`scotta3270maximize` `scotta5250maximize`
Window Size: Scale to Fit Window	`--scalable`	`scottascalable`
Window Size: Width	`--width`	`scottaappletwidth`
Window Type	`--displayusing`	`scottaapplicationplacement`
Window Type: New Browser Window	`--newbrowser`	`scottaoutofplace`
Windows Protocol	`--winproto`	`scottawindowsapplicationserver`
Windows Protocol: Try Running From Client First	`--trylocal`	`scottatrylocalwindowsapplication`
X Security Extension	`--securityextension`	`scottasecurityextension`