Oracle Responsys

SOAP API Developer's Guide — Standard

IMPORTANT: The Oracle Responsys Web Services SOAP API is in maintenance mode. There will be no new enhancements to the SOAP API.

Oracle Responsys continues to support the SOAP API but encourages you to use the REST API. Documentation for the REST API can be found on the Responsys documentation page:
https://docs.oracle.com/cloud/latest/marketingcs_gs/responsys.html

November 2020

Documentation for developers who use the Oracle Responsys SOAP API to access the data, content, and campaign management features of Oracle Responsys.

Oracle Responsys SOAP API Developer's Guide

E65152-18

Address permission requests, comments, or suggestions about Oracle Responsys documentation by creating a MOS Service Request at https://support.oracle.com.

Contents

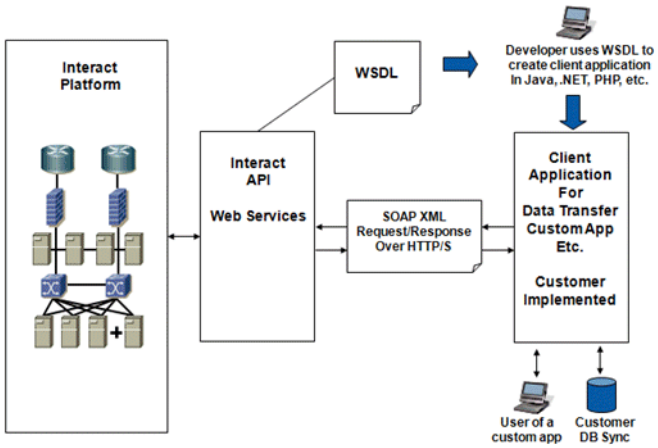
Introducing the Oracle Responsys Interact API

The Oracle Responsys® Interact API (Oracle Responsys API) gives you standards-based access to the data, content, and campaign management features of Oracle Responsys. Using the Oracle Responsys API, you can build solutions for marketing data automation, customize your campaign and content management processes, and remotely trigger events for recipients thereby entering them into Oracle Responsys-based life cycle messaging programs.

Specifically, you may want to use the Oracle Responsys API to:

- Synchronize marketing data between enterprise and partner systems
- Trigger individual email or mobile messages in response to some external event or activity detected by your web site or enterprise information systems
- Automate the import of creative content needed for your campaigns

This conceptual diagram shows how to use the Oracle Responsys API.



Because the Oracle Responsys API is based on a service-oriented architecture (SOA) and other industry-standard technologies such as SOAP and WSDL, your developers can use their choice of programming language and development environment to gain full programmatic access to your organization's Oracle Responsys account. The Oracle Responsys API supports easy integration of your enterprise systems with the campaigns and data stored in your Oracle Responsys account - enabling greater automation of marketing tasks and processes.

Oracle Responsys API functionality

The Oracle Responsys API supports the following subset of the functionality of the Oracle Responsys user interface and platform.

Session Management

- Login/Logout of an Oracle Responsys API session
- Retrieving the current Oracle Responsys timestamp

List and Data Management

- Insert, update, and delete records in Lists and Supplemental Tables
- Retrieve records from Lists, Supplemental Tables, and Profile Extension Tables
- Retrieve updated list member records

Content Management

- Create or delete document objects
- Set or get image files for a document object
- Set or get the markup content for a document object

Campaign Management

- Launch a campaign
- Get campaign launch status

Lifecycle Messaging Programs

- Trigger campaign messages to individual recipients
- Trigger custom events for individual recipients

About Oracle Responsys API URLs

When your account is enabled for access to the Oracle Responsys API, the Responsys Support team gives you the Web Services URLs you need to develop your projects. Depending on where your account is set up in the Responsys data center, you'll get Web Services URLs for the Interact 2 pod, the Interact 5 pod, or the Interact 8 pod.

Development environments

The Oracle Responsys API works with modern SOAP development environments such as Visual Studio .NET, Apache Axis, and others. Development platforms vary in their SOAP implementations and differences in implementation might prevent access to some or all of the features in the API. If you are using Visual Studio for .NET development, we recommend that you use Visual Studio 2003 or later.

Oracle Responsys maintenance and downtime

Oracle Responsys undergoes maintenance downtimes on a monthly or bi-monthly schedule. During these downtimes, Campaign login sessions are not available. Attempts to create a login session during downtimes return an error and client applications need to take the appropriate action, which may include alerts to support staff, integration job queuing, and/or scheduled re-tries.

Monitoring and throttling the frequency of API requests

Responsys monitors and throttles the frequency of API requests that are submitted from each Oracle Responsys account. This is to ensure that the best possible level of service is offered to API clients in a shared environment.

You can use the Get Throttling Limits REST API to obtain a list of API throttling limits for key interfaces for your Responsys account. For more information about using this API, see the Get Throttling Limits topic in the *REST API for Oracle Responsys Marketing Cloud Service* guide.

Depending on the type of API function, a specific frequency rate limit is imposed on the basis of an account's number of requests made per minute for that function. For example, the API function for triggering email messages can be called more times per minute than the API function for launching a campaign. By default, the throttling limit for high volume API functions (for example, triggering email messages or merging records into a profile list) is set to 200 requests per minute.

When an account exceeds its allowable frequency rate limit for an API request, you see the error code API_LIMIT_EXCEEDED and this message "You exceeded your allowable limit to call the <function_name> API function. Please try again in a minute." (See Sample Code for Handling Exceeded Account Limits on page 73 for the appropriate block of sample code.) On the other hand, if a specific user of an account is blocked from using selected API functions, the user sees the error code API_BLOCKED with this message: "The <function_name> is currently not available to this user. Please contact tech support."

Backward compatibility

Responsys supports backward compatibility as new versions of the Oracle Responsys API are released. This means that an application created to work with a given Oracle Responsys API version will continue to work with that same Oracle Responsys API version in future platform releases. Each version of the Oracle Responsys API has a unique endpoint URL. Your applications will continue to work with the Oracle Responsys API endpoint URLs of previous releases. You can migrate your client applications to newer Oracle Responsys API version endpoint URLs to take advantage of enhanced functionality and bug fixes on a schedule that meets your needs.

Responsys does not guarantee that an application written against one Oracle Responsys API version will work with future API versions, because changes in method signatures and data representations are often required to enhance Oracle Responsys. However, we strive to keep the Oracle Responsys API consistent from version to version with minimal if any changes required to port applications to newer Oracle Responsys API versions. When an API version is to be deprecated, advance end-of-life notice will be given at least 9 months before support for the API version is ended. Oracle Responsys will directly notify customers using API versions planned for deprecation.

Web service standards compliance

The Oracle Responsys API was implemented in compliance with these specifications:

- Simple Object Access Protocol (SOAP) 1.1
  http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
- Web Service Description Language (WSDL) 1.1
  http://www.w3.org/TR/2001/NOTE-wsdl-20010315
- WS-I Basic Profile 1.1
  http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html

Oracle Responsys platform and data model overview

Oracle Responsys is a comprehensive on-demand marketing platform with a fully integrated suite of software applications—all built from the ground up on a single-instance, multi-tenant architecture.

Oracle Responsys Platform

Oracle Responsys platform currently offers the following on-demand applications:

- Campaign? for multichannel campaign management lets you efficiently create, test, execute, and measure high-volume, highly individualized marketing campaigns across touch-points for compelling ROI.
- Program? for dialogue and event-based marketing helps you orchestrate and automate intelligent, customer-driven dialogs at desired moments in the customer lifecycle for more relevant, profitable interactions.
- Team? for marketing process management is designed to help you plan, coordinate, and monitor marketing projects and resources for greater marketing efficiency and improved collaboration among geographically distributed marketing teams.
- Insight? for predictive analytics and contact optimization uses cutting-edge analytical models to identify your most relevant customer segments and produce contact strategies optimized for each segment.
- Connect? for data integration makes it easy to integrate Oracle Responsys with your enterprise or marketing information systems to better utilize marketing data and gain a complete view of customers at every interaction point.

Oracle Responsys Object Data Model

You can use the Oracle Responsys platform to create and manage a variety of objects to manage your marketing database and execute your marketing campaigns. The Oracle Responsys object model consists of these types of objects:

- Programs let you assemble multi-campaign dialogs.
- Campaigns help you execute email campaigns in batch launch or triggered modes.
- Forms enable you to collect data via web forms (not currently supported via the Oracle Responsys API).
- Documents consist of re-usable creative content that is available for use in any Campaign or Form.
- Data objects enable you to store and use data for a variety of purposes.
  - Lists and related objects (Filters, Proof Groups, Segmentations) store recipient audience records and are used primarily for campaign targeting and personalization.
  - Profile Extension Tables store additional information for each unique recipient in your profile list table.
  - Supplemental Tables and related objects (Filters, SQL object, Join objects) store miscellaneous data that can be used to define a multi-table relational schema for advanced levels of segmentation, targeting and message personalization.
  - Link Tables store campaign link tracking information.

The Oracle Responsys API provides control over many of these objects, allowing client applications to create, change, or remove these objects in a programmatic way to accomplish a variety of marketing automation goals.

Programs

Program objects define multi-step dialogs that involve a variety of campaign messaging and routing rules based on individual profile and behavioral attributes. Creation of an individual Program takes place in a visual, drag-and-drop user interface that is part of Program. The Oracle Responsys API can be used to trigger Custom Events which enter an individual into or affect the individual's routing in a program.

Campaigns

Campaign objects define the basic behavior of an email campaign in terms of audience, message, and settings.

- General properties include name, type (email or mobile), description, categorization, and other fields that identify the campaign.
- Audience selectors include a list, inclusion filters, exclusion filters, and suppression data.
- Message elements include From header, Reply-to header, Subject header, and HTML/Text message documents.
- Settings control tracking options, auto-close behavior, default variables, and so forth.

You can launch a campaign in bulk immediately or schedule it for a later launch. You can also trigger messages from a campaign on demand using form handler rules or program rules.

Forms

Form objects provide functionality for hosting web forms and collecting/processing the data that is submitted. You can use forms to gather customer preferences or for general purpose surveys. Data collected from forms can be merged into a list or supplemental table. Form responses can trigger follow-up emails and custom events that place the responder in a Program dialog.

Documents

Document objects contain the creative content used for campaigns and forms. The two types of the document object are HTML and text. For example, an email campaign usually consists of an HTML and a corresponding text document reference. The campaign handles the display of HTML-only, text-only, or multi-part emails automatically based on the recipient profile. Documents can be re-used across multiple campaigns and forms, copied, edited, and deleted via Campaign.

Lists and related objects

Lists are used to store audience database records—members of your audience might be leads, prospects, customers, contacts, consumers, or visitors, depending on your terminology. The standard set of fields in a list includes:

- Recipient ID (RIID), an internalOracle Responsys-assigned identifier that allows the behavior of individual recipients to be tracked over time.
- Email address, mobile number, postal address, which are standard contact channel fields
- Permission/Opt-in status fields for the various marketing channels (email, mobile, postal)
- Email format preference (HTML or text)
- Derived fields for ISP and domain
- Last modified and created timestamps

In addition, lists can have a number of custom, user-defined fields that you use to maintain a rich audience profile for targeting and personalization purposes.

» Note An account can have any number of lists, but it is recommended that a single central list is used for a given enterprise marketing objective. In some cases, it may make sense to have multiple lists, but use of multiple lists can generate duplicate identities for the same individual audience member.

These are the list-based objects:

- List filters are user-defined segments that contain a subset of the members of a list. You can use list filters to include or exclude members from any given campaign launch.
- List segmentations are a way of understanding how a list breaks down in terms of a given set of segments. For example, multiple purchasers, one-time purchasers, and non-purchasers.
- List seeds store records that share the same schema for a given list, but are used for testing and seeding of campaigns. These records do not represent real members (prospects, customers, and so forth).

Profile extension tables

One or more Profile Extension Tables can be associated with a Profile List. There must be a one-to-one relationship between a record in a Profile Extension Table and its parent Profile List. Profile Extension Tables provide an attractive and efficient way to organize and process audience data. Similar to data in Profile Lists, audience data in Profile Extension Tables can be used for segmentation and targeting in Filters as well as Programs.

Supplemental tables and related objects

As its name indicates, a supplemental table is a collection of database records that supplements a list with additional related information. The connections between a table and a list is made via a *data extraction key*, or key field, that is present in both the table and the list. Because you define the schema for any tables you create, you can use them for a wide variety of purposes, ranging from message personalization and dynamic content to storing form responses and campaign events.

There are several type of table-based data sources: tables, filters on tables, SQL views (on tables and/or lists), and joins on tables.

» Tip When you use tables to extend a list to represent a multi-table relational marketing database (where a variety of queries or joins could be made on the table), be sure to index your tables to reduce the performance impact associated with full table scans on tables being queried or joined.

Link Tables

Link Tables are used to store data about the links that are tracked for a campaign. The schema for a Link Table is fixed and consists of the following fields:

- LINK_NAME defines user-friendly name for the link.
- LINK_URL defines the destination URL for a tracked link.
- LINK_CATEGORY defines a category for links and is available for reporting.
- EXTERNAL_TRACKING defines optional parameters that can be appended to the query-string of the destination URL.

API Call Processing

Web Services API calls are processed synchronously. For most calls, you should receive a response shortly after Responsys finishes processing the call. However, some of the API calls trigger system actions that are performed after the system receives the API call and sends the positive response. If those attempts fail for some reason, you may have received a "true" response for the system receiving the API call, but the failure would be recorded elsewhere.

Examples:

- API calls that trigger messages requiring personalization processing. Responsys processes personalization asynchronously after the API call has returned a positive response. If the personalization fails, then you may receive a positive API response, even though the message was not sent to the recipient.
- Trigger custom event API calls. These calls do not actually send the email or mobile messages. The triggerCustomEvent API call merely sends a group of recipients (that is, enactments) to a Program. The Program subsequently uses its own logic to determine if those enactments will be added to campaigns within that Program. The campaigns ultimately send the email or mobile messages.

How Enactment Batching Affects Processing

Oracle Responsys enables cross-channel orchestrations with email, SMS, and Push. **If you plan to use the trigger custom event API call with cross-channel marketing programs, please review this section first.**

Responsys requires the Enactment Batching feature to be enabled when using trigger custom event with mobile app campaigns in Program. Otherwise, the mobile app campaign events in the program will not be processed. However, there are some trade-offs to consider before enabling the feature. When an account has Enactment Batching enabled, triggering a custom event cannot be used to perform near-real-time processing for ***any*** campaign type. Responsys will batch enactments together into a single enactment group before entering the enactments into a program. This results in ***at least*** a 10-minute delay between custom event triggering and entry into a Program. **If your account has Enactment Batching enabled and you need to send near-real-time messages, such as event reactions, then we highly recommend having your account enabled for the Real-time Events feature.**

The Real-time Events feature is intended for Responsys customers who use the Mobile App channel. Part of this feature enables you to create real-time custom events. Real-time custom events are a special type of custom event that override how Responsys handles enactments when the Enactment Batching feature is enabled. When a real-time custom event is triggered, Responsys handles the enactments in near real-time instead of batching them. This ensures that your customers receive the campaign messages (including Email, SMS, Push, and In-app) without the delay imposed by enactment batching. To have this feature enabled for your account, contact your Oracle Customer Success Manager. For more information, see the Defining Custom Event Types topic in the Oracle Responsys Help Center.

Access Controls

This section presents Oracle Responsys capabilities for controlling user access to APIs.

Organizational access control

When Organizational Access Control is enabled for an Oracle Responsys account, it will be enforced for all users in that account, including API users. This means that the API user's access to Oracle Responsys objects will be limited by the organizational units to which the user is assigned. Similarly, objects created through the API will inherit organizational membership of the API user. For the API user to access all objects within the account, that API user should be assigned to the Root node in the organizational hierarchy.

Organizational Access Control is configured through “Account | Manage Users | Organization Assignment”. Please contact your Oracle Responsys account administrator to set up access control.

Functional access control

API user's access level to a specific object is determined by functional roles that are assigned to that user. Functional Access Control and Organizational Access Control work together. Organizational Access Control determines whether the API user has access to a particular object, and Functional Access Control determines what operations the user can perform with that object.

Best practice recommendation is to use a dedicated user for API operations. API user should be assigned one or more functional roles (Campaign Web Services Manager, Folder Web Services Manager, Table Web Services Manager, Content Web Services Manager, or List Web Services Manager) to ensure adequate access level to the appropriate set of objects.

Functional Access Control is configured through “Account | Manage Users | Role Assignment”. Please contact your Oracle Responsys account administrator to setup access controls.

Login IP enforcement access control

Oracle Responsys enables customers to limit login access based on their defined range(s) of authorized login IP addresses. The system immediately denies any login attempts initiated outside of your authorized ranges of login IP addresses. These restrictions apply to the API user as well as to users logging in to the Responsys user interface.

To view the IP access list settings, a Responsys account administrator can log in and to go “Account | View login IP restrictions”.

Getting started with the Oracle Responsys API

This section contains general instructions for using the Oracle Responsys API as well as guidelines and sample code for using the Oracle Responsys API in a Java or C# application.

» **Note** We assume that you have a basic familiarity with software development, SOAP-based Web Services, and the Oracle Responsys platform and user interface.

To get started with the Oracle Responsys API:

1. Obtain the Oracle Responsys API WSDL that corresponds to the Oracle Responsys pod for your account:
   Interact 5: https://ws5.responsys.net/webservices/wsdl/ResponsysWS_Level1.wsdl
   or
   Interact 2: https://ws2.responsys.net/webservices/wsdl/ResponsysWS_Level1.wsdl
   or
   Interact 8: https://ws.rsys8.net/webservices/wsdl/ResponsysWS_Level1.wsdl
   or
   Interact 9: https://ws.rsys9.net/webservices/wsdl/ResponsysWS_Level1.wsdl
   Or your unique global routing endpoint. In the Responsys interface, select **Account > Global settings > Account configuration** and look for the endpoint in the **WS End Point** field.

2. Use the Oracle Responsys API WSDL to generate supporting code for creating SOAP calls on the Oracle Responsys API.

   Your development environment or programming language should provide the necessary support for accomplishing this step. The benefit of SOAP/WSDL-based APIs is that most programming languages provide support for managing SOAP requests and responses.

   NOTE: If the error File Not Found occurs while generating the stub using wsdl2java or a similar WSDL utility, the actual WSDL URL should be used as shown above instead of downloading the WSDL locally and generating the functions from there.

3. Ensure that your client systems use Transport Layer Security (TLS) version 1.2.

4. Establish a session with the Oracle Responsys Web Service, using one of the following methods of authentication:
   - Authenticate with username and password using the Login call
   - Authenticate with certificates using the authenticateServer and LoginWithCertificate calls.

   These calls return a session identifier.

5. Place the session identifier returned by Responsys into the SOAP header of all subsequent calls to the Oracle Responsys API to authenticate the client application.

6. Place a session cookie (JSESSIONID) on the client application after the first successful API call.

   This cookie should be persisted for the duration of the session. Make sure that your client accepts session cookies.

7. Use the available API calls to accomplish a desired goal, including:
   - Data API calls to create, modify or delete individual records.
   - Connect API calls to import or export data in bulk.
   - Campaign API calls to create or modify campaign definitions or launch campaigns.
   - Content API calls to create, modify or delete content documents.

» Note Some Oracle Responsys API calls have a maximum number of records that can be processed per invocation (triggerCustomEvent, plus all data source merge, retrieve, and delete calls). For example, the Oracle Responsys API limit for calls for triggering campaign messages and merging records into a list is 200 recipients or records. Therefore, you may need to execute these calls in a loop to process additional records during a given client session.

8 If your client application is inactive for longer than two hours and the session identifier becomes invalid, start a new session with a new Login call.

9 Use the Logout call to end the Oracle Responsys API session.

» Note You should explicitly log out before attempting a new login call, because there is a limit of 100 concurrent SOAP API sessions (by default) that you can create for each Oracle Responsys account.

Java Applications

These are general instructions for getting started with the Oracle Responsys API from a Java application.

To get started with a Java application:

1 Download the WSDL document. Name the downloaded file ResponsysWS.wsdl and place it in your project directory.

Responsys Support will provide the Oracle Responsys API URLs to you when your account is enabled for Oracle Responsys API access.

2 Use the Apache Axis2 WSDL2Java utility, as described on the Apache Axis2 web site, to generate Web Services API stub classes:

- %AXIS2_HOME%\bin\WSDL2Java -uri ResponsysWS.wsdl -u -d adb -s -p com.rsys.ws.client
- Assuming the following environment variables are defined:
  - AXIS2_HOME = C:\axis2-1.3 (or location of the Apache Axis2 Standard Distribution)
  - AXIS2_LIB = %AXIS2_HOME%\lib
  - AXIS2CLASSPATH = %AXIS2_LIB%\axis.jar;%AXIS2_LIB%\jaxrpc.jar;%AXIS2_LIB%\saaj.jar;%AXIS2_LIB%\commons-logging.jar;%AXIS2_LIB%\commons-discovery.jar;%AXIS2_LIB%\wsdl4j.jar

3 In your Java application, make sure that the generated Oracle Responsys API stub classes are available to your project build path.

4 Import the following WSDL2Java-generated packages or specific classes needed for your client application calls:

```
import com.rsys.ws.*;
import com.rsys.ws.client.*;
```

5 Instantiate a Oracle Responsys API service object:

```
service = new ResponsysWSServiceStub("...WS Endpoint URL...");
```

6 Maintain the JSESSIONID cookie between requests with the following statement:

```
service._getServiceClient().getOptions().setManageSession(true);
```

7 Instantiate a new Login request object and call the login method of the stub object:

```
Login login = new Login();
login.setUsername("username");
login.setPassword("password");
LoginResponse response = service.login(login);
```

8 Retrieve the sessionId string from the login result.

9 Submit this sessionId in the SOAP header for all following Oracle Responsys API calls.

10 Continue with client application logic.

11 End session by logging out when client application task is completed.

Java example

```
import com.rsys.ws.*;
import com.rsys.ws.client.*;
import java.rmi.RemoteException;

public class APITestLoginLogout {
   ResponsysWSServiceStub stub;
   SessionHeader sessionHeader;

   public static void main(String[] args) {
        APITestLoginLogout test = new APITestLoginLogout();
        test.login();
}

private void login() {
      try {
         stub = new ResponsysWSServiceStub("https://...WS Endpoint URL...");
         // maintain session between requests
         stub._getServiceClient().getOptions().setManageSession(true);
// CAUTION: It is important that the user session be maintained. Do no omit preceding step.
         Login login = new Login();
         login.setUsername("...username...");// substitute actual username for username
         login.setPassword("...password..."); // substitute actual password for password
         LoginResponse response = stub.login(login);
         String sessionId = response.getResult().getSessionId();
         System.out.println ("Login Result = " + sessionId);
         if (sessionId != null) {
            sessionHeader = new SessionHeader();
            sessionHeader.setSessionId(sessionId);
            // Set optional timeout to two minutes
            stub._getServiceClient().getOptions().setTimeOutInMilliSeconds(1000*60*2);
// CAUTION: It is important to set a timeout that is appropriate for the maximum expected duration of
    API calls
            ListFolders listFolders = new ListFolders();
            ListFoldersResponse listFoldersResponse = stub.listFolders(listFolders, sessionHeader);
            FolderResult[] folders = listFoldersResponse.getResult();
            if (folders != null) {
               System.out.println ("Folders length = " + folders.length);
               int i = 0;
               for (FolderResult folder : folders) {
               System.out.println ("Folder Name = " + folder.getName());
                  i++;
               }
            }
            LogoutResponse logoutResponse = stub.logout(new Logout(), sessionHeader);
            boolean loggedOut = logoutResponse.getResult();
            System.out.println("Logout Result = " + loggedOut);
         }
      } catch (AccountFault accountEx) {
            System.out.println ("Ex Code = " + accountEx.getFaultMessage().getExceptionCode());
            System.out.println ("Ex Msg  = " + accountEx.getFaultMessage().getExceptionMessage());
      } catch (UnexpectedErrorFault unexpectedEx) {
            System.out.println ("Ex Code = " + unexpectedEx.getFaultMessage().getExceptionCode());
```

```
            System.out.println ("Ex Msg  = " + unexpectedEx.getFaultMessage().getExceptionMessage());
        } catch (RemoteException remoteEx) {
            System.out.println ("Ex Msg = " + remoteEx.getMessage());
        }
    }
}
```

C# Applications

These are general instructions for getting started with the Oracle Responsys API from a C# application.

To get started with a C# application:

1. Download the WSDL document. Name the downloaded file ResponsysWS.wsdl.

   Responsys Support will provide the Oracle Responsys API URLs to you when your account is enabled for Oracle Responsys API access.

2. Generate the client-side code needed to support your client application's programmatic calls on the Responsys Web service.
   - Open the command window from the Visual Studio menu or include the .NET Framework's bin directory in path environment variable. Type the command WSDL ResponsysWS.wsdl
   - Copy the resulting C# file, ResponsysWSService.cs, to your project directory.
3. In your C# application, get a handle for the Web Service, and ensure the user session will be maintained. See example provided below.
4. Use the C# compiler to create an executable named fileName.exe, where fileName is the .CS file that contains the Main() method.

   ```
   csc *.cs
   ```
5. Be sure that csc.exe is in your path, usually:

   C:\WINDOWS\Microsoft.NET\Framework\v2.0.xxx\)

C# example

```
namespace WSCSharpClient {
    using System;
    using System.Net;
    using System.IO;
    using System.Xml;
    using System.Web.Services.Protocols;

    class TestResponsysWS {
        ResponsysWSService stub;
        bool loggedIn = false;
        SessionHeader sessionHeader;

        private bool login() {
            bool result = false;
            try {
                string url = "... WS Endpoint URL ...";
                Console.WriteLine("Web Services URL = " + url);

                string username = "username"; // substitute actual user name for username
                string password = "password"; // substitute actual password for password

                stub = new ResponsysWSService();
                stub.CookieContainer = new CookieContainer();
// Caution: It is important that the user session be maintained, so do not omit the preceding step.
                stub.Url = url;
                // Call the login method
                LoginResult loginResult = stub.login(username, password);
```

```
            string sessionId = loginResult.sessionId;

            if (sessionId != null) {
                // Create the sessionHeader object and set it to the stub.
                // The sessionHeader is passed to every other API call after the login.
                sessionHeader = new SessionHeader();
                sessionHeader.sessionId = sessionId;
                stub.SessionHeaderValue = sessionHeader;
// Caution: It is important to set a sessionHeader object to the stub as it is used in all the subsequent
    calls.
                sop("Setting the Client Timeout to 2 minutes");

                // Set timeout
                stub.Timeout = 1000 * 60 * 2;
// Caution: It is important to set a timeout that is appropriate for the maximum expected duration of API
    calls.
                loggedIn = true;
                result = true;
            }
        } catch  (System.Web.Services.Protocols.SoapException e) {
            Console.WriteLine("SoapException in login : " + e.Message);
            Console.WriteLine("SoapException in login : " + e.Detail.InnerText);
        } catch  (Exception e) {
            Console.WriteLine("Exception in login : " + e.Message);
        }
        return result;
    }
  }
}
```

Important .NET WSDL edits required

If you are using the Microsoft .NET WSDL, you must make a correction to the RecordData element in the ResponsysWS.wsdl file. This element contains an array of Record elements, each of which contains an array of Strings.

However, the Microsoft .NET wsdl.exe has a defect that affects arrays inside of other arrays. It creates the recordsField as a two-dimensional string array instead of an array of Record class. Furthermore, the Record class is not created at all in the ResponsysWSService.cs class. You can fix this by editing the ResponsysWSService.cs class to create a Record class and changing the two-dimensional string array in the RecordData class to an array of Record objects.

To make required .NET WSDL edits to the ResponsysWSService.cs class:

1 Create the following Record class.

```
/// <remarks/>
    [System.CodeDom.Compiler.GeneratedCodeAttribute("wsdl", "2.0.50727.42")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlTypeAttribute(Namespace="urn:ws.rsys.com")]
    public partial class Record {

        private string[] fieldValuesField;

        /// <remarks/>
        [System.Xml.Serialization.XmlElementAttribute("fieldValues", IsNullable=true)]
        public string[] fieldValues {
            get {
                return this.fieldValuesField;
            }
            set {
```

```
            this.fieldValuesField = value;
        }
    }
}
```

2 Change the string[][] recordsField in RecordData class to Record[] recordsField by replacing the contents of the RecordData class with this:

```
/// <remarks/>
[System.CodeDom.Compiler.GeneratedCodeAttribute("wsdl", "2.0.50727.42")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlTypeAttribute(Namespace="urn:ws.rsys.com")]
public partial class RecordData {
    private string[] fieldNamesField;
    private Record[] recordsField;
    /// <remarks/>
    [System.Xml.Serialization.XmlElementAttribute("fieldNames", IsNullable=true)]
    public string[] fieldNames {
        get {
            return this.fieldNamesField;
        }
        set {
            this.fieldNamesField = value;
        }
    }
    /// <remarks/>
    [System.Xml.Serialization.XmlElementAttribute("records", IsNullable=true)]
    public Record[] records {
        get {
            return this.recordsField;
        }
        set {
            this.recordsField = value;
        }
    }
}
```

Interact Calls, Types, Objects, and Result and Exception Codes

The Interact calls are divided into these categories:

The Interact also contains standard primitive types—boolean, string, int and long, and dateTime—as well as a collection of objects to be used with API calls.

In addition, the Interact provides result codes and exception codes divided into these categories:

Session Management API calls

The Session Management API calls are:

- Authenticate Using Username and Password (Login)
- Logout
- LoginWithCertificate
- AuthenticateServer

Authenticate Using Username and Password (Login)

Syntax:

```
LoginResult = service.login(string username, string password)
```

Usage

The first step for any client application is to establish a login session. This can be achieved using the login call.

When a client application invokes the login call, it passes a username and password as user credentials. Upon receiving the client application login request, the API authenticates these credentials, and returns a LoginResult object. This object can be inspected to retrieve a session token that is required for use in all subsequent API calls. After successfully completing the login call and retrieving the session token, a client application needs to set this session token in the SOAP header for subsequent calls as a means of authentication.

Session tokens expire automatically after two hours of inactivity. Client applications that make infrequent login calls should make explicit logout calls to prevent the accumulation of unnecessary open sessions. **By default, Oracle Responsys limits the number of concurrent API sessions that an account can initiate to 100 SOAP API sessions.** It is important to properly manage API sessions to avoid exceeding this limit. If the limit is reached, an error message will be returned, stating that the allowed number of concurrent sessions has been exceeded.

A JSESSIONID cookie is also set on the client application with the response from the login call. This cookie must be persisted for use in subsequent API calls in the session.

» Note If you are using either Axis2, C# or any other .Net language, the JSESSIONID is automatically captured and sent in subsequent requests. However, if you are not using one of these languages, you must capture the JSESSIONID and Path from the login response HTTP Headers and set them in a cookie in the HTTP headers of all subsequent requests until you log out. This will prevent errors.

Example

```
HTTP/1.1 200 OK
Date: Tue, 16 Nov 2010 14:52:14 GMT
Set-Cookie: JSESSIONID=C1DC1654EE6BBEEBE94043EE4D006F59.tmws2; Path=/tmws
Content-Type: text/xml;charset=UTF-8
Connection: close
Transfer-Encoding: chunked
```

Request Arguments

Name	Type	Description
username	string	User name for the Oracle Responsys account.
password	string	Password for the specified user.

Response

The login call returns a LoginResult object, which has the following property:

Name	Type	Description
sessionId	string	Unique Session ID associated with this session. Your client application needs to set this value in the session header of subsequent API calls.

Logout

Syntax

```
boolean  = service.logout()
```

Usage

Use the logout call to end an API session. The last step for any client application is to end a session by logging out. Note that sessions are terminated automatically after two hours of inactivity.

Request Arguments

None

Response

Name	Type	Description
result	boolean	Flag representing the success of a request to end the API session.

Authentication with Certificates (authenticateServer + loginWithCertificate)

Authentication with certificates is based on the use of a digital certificate in accordance with the X.509 standard for public key infrastructure (PKI). It is available for developers that require the security advantages of PKI over password-based authentication.

Authentication with certificates requires using both the authenticateServer and loginWithCertificate calls, which are described in the next two sections.

To develop a client application with this type of authentication, the Oracle Responsys account administrator must perform the following steps in Responsys:

1. Log into the Oracle Responsys user interface and navigate to the admin console.
2. Upload a digital certificate (client user public key).
3. Download the Interact server digital certificate (server public key).

These certificates will be used by the client application to log in with the authenticateServer and loginWithCertificate calls.

The client application establishes an authenticated session as follows:

- First, the client application uses the **authenticateServer** call (described in the next section) with a user name and client challenge.
- The server returns a server challenge (a byte array), an encrypted response to the client challenge (that is, the client challenge encrypted using the server certificate's private key), and a temporary session ID (authSessionId) for this authentication step.
- The client application confirms that the server is authentic by decrypting the client challenge returned from Responsys (using the server public key), and then comparing it with the original challenge sent by the client. If they match, the client application is communicating with a valid Responsys server. The client application prepares a response to the server challenge.
- The second step of the authentication involves calling **loginWithCertificate** with the response to the server challenge and the temporary session ID placed in the SOAP header.
- Responsys then authenticates these credentials, and returns a LoginResult object. This object can be inspected to retrieve a new session token that is required for use in all subsequent API calls.
- After successfully completing the loginWithCertificate call and retrieving the session token, a client application needs to set this session token in the SOAP header for subsequent calls as a means of authentication.

Session tokens expire automatically after two hours of inactivity. Client applications that make infrequent login calls should make explicit logout calls to prevent the accumulation of unnecessary open sessions. A limit is placed on the number of concurrent API sessions that an account can initiate. It is important to properly manage API sessions to avoid exceeding this limit. If the limit is reached, an error message will be returned, stating that the allowed number of concurrent sessions has been exceeded.

To authenticate using certificates:

1. Prepare a client challenge as a byte array.

» IMPORTANT The client should use the RSA algorithm for encryption and decryption, because that is the algorithm used by the server.

2 Call authenticateServer with an Oracle Responsys user name and the client challenge and receive a server challenge, an encrypted response to the client challenge, and a temporary session ID for this authentication process.

3 Validate the encrypted client challenge by decrypting with the server public key. **Stop the process if the server authenticity cannot be confirmed.**

4 Prepare a response to the server challenge by encrypting the server challenge with the client private key.

5 Call loginWithCertificate with the encrypted server challenge and the temporary session ID placed in the SOAP header.

6 The Responsys server will authenticate the client by decrypting the server challenge with the previously uploaded client public key.

7 Upon successful authentication, the Responsys server will respond with a LoginResult object from which a valid web services session ID can be retrieved for use in all subsequent API calls.

See Sample Code for Certificate Authentication (Java) on page 75 for sample Java client code for certificate authentication.

AuthenticateServer

Syntax

```
ServerAuthResult = service.authenticateServer(string username, byte[] clientChallenge)
```

Usage

Use the authenticateServer call to authenticate the Interact server and initiate a successful login to the Interact. The information returned from this API call can be used to successfully log in to the Interact with the loginWithCertificate call.

A client application can establish an authenticated session in two steps.

1 First, the client application uses the authenticateServer call with a user name and client challenge and then receives a server challenge, an encrypted response to the client challenge, and a temporary session ID for this authentication step. The client application confirms that the server is authentic and prepares a response to the server challenge.

2 The second step of the authentication involves calling loginWithCertificate with the response to the server challenge and the temporary session ID placed in the SOAP header.

» **Note** A JSESSIONID cookie is also set on the client application with the response from the authenticateServer call. This cookie must be persisted for use in subsequent API calls in the session.

Request Arguments

Name	Type	Description
username	string	User name for the Oracle Responsys account of interest.

Name	Type	Description
clientChallenge	byte[]	Client application challenge of the server which is used to confirm the authenticity of the server.

Response

The authenticateServer call returns a ServerAuthResult object, which has the following properties:

Name	Type	Description
authSessionId	string	Temporary session ID that should be placed in the SOAP header of the subsequent loginWithCertificate call.
encrytpedClientChallenge	byte[]	Response to the client challenge, represented by encrypting the client challenge with the server private key. Client applications should validate server authenticity by decrypting this value with the server public key (available through the Oracle Responsys user interface admin console).
serverChallenge	byte[]	Server challenge of client application authenticity. This challenge should be encrypted with the client private key and submitted with the loginWithCertificate call to authenticate the client application session.

LoginWithCertificate

Syntax

```
LoginResult = service.loginWithCertificate(byte[] encryptedServerChallenge)
```

Usage

Use the loginWithCertificate call with the authenticateServer call (described in the previous section) to establish a login session by authenticating with certificates.

Request Arguments

Name	Type	Description
encryptedServerChallenge	byte[]	Encrypted value of the server challenge. The server challenge is encrypted using the client private key that corresponds to a client public key certificate that was uploaded via the Oracle Responsys admin console as the means to authenticate Interact session requests.

Response

This call returns a `LoginResult` object, which has the following property:

Name	Type	Description
sessionId	string	Unique Session ID associated with this session. Your client application needs to set this value in the session header of subsequent API calls.

Folder Management API calls

The Folder Management API calls are:

- CreateContentLibraryFolder
- CreateFolder
- DeleteContentLibraryFolder
- DeleteFolder
- DoesContentLibraryFolderExist
- ListContentLibraryFolders
- ListFolders

CreateContentLibraryFolder

Syntax

```
HierarchyElement = service.createContentLibraryFolder (String folderName)
```

Usage

Use the create ContentLibraryFolder call to create a new empty folder in the Content Library.

Request Arguments

Name	Type	Description
folderName	string	The name of the folder to create.

Response

Name	Type	Description
result	HierarchyElement	The content library folder.

CreateFolder

Syntax

```
boolean = service.createFolder(string folderName)
```

Usage

Use the createFolder call to create a new empty folder in an Oracle Responsys account. This call returns a boolean value that indicates the success of the folder creation request.

Request Arguments

Name	Type	Description
folderName	string	Name of the folder to create.

Response

Name	Type	Description
result	boolean	Success flag folder creation.

DeleteContentLibraryFolder

Syntax

```
void service.deleteContentLibraryFolder (String folderName)
```

Usage

Use the deleteContentLibraryFolder call to delete a folder and its contents from the Content Library.

Request Arguments

Name	Type	Description
folderName	string	The name of the folder to delete.

Response

Name	Type	Description
void	N/A	Delete content library folder.

DeleteFolder

Syntax

```
boolean = service.deleteFolder(string folderName)
```

Usage

Use the deleteFolder call to delete a folder and its contents from an Oracle Responsys account.

Request Arguments

Name	Type	Description
folderName	string	Name of folder to delete.

Response

Name	Type	Description
result	boolean	Success flag for deletion of folder.

DoesContentLibraryFolderExist

Syntax

```
boolean = service.doesContentLibraryFolderExist (String path)
```

Usage

Use the doesContentLibraryFolderExist call to check whether a specific folder exists in the Content Library.

Request Arguments

Name	Type	Description
path	string	The name of the folder to check.

Response

Name	Type	Description
result	boolean	True if the folder exists.

ListContentLibraryFolders

Syntax

```
List<HierarchyElement> = service.listContentLibraryFolders(String startingPath, Boolean
   showTree)
```

Usage

Use the listContentLibraryFolders call to retrieve a listing of all Content Library folders.

Request Arguments

Name	Type	Description
startingPath	string	The starting parent folder.
showTree	boolean	True displays the full Content Library folder structure. If startingPath is specified, shows all children in the tree, not only the starting path's immediate child folders.

Response

Name	Type	Description
result	List<HierarchyElement>	List of Content Library folders.

ListFolders

Syntax

```
FolderResult[] = service.listFolders()
```

Usage

Use the listFolders call to retrieve a listing of all of the folders in an account.

Request Arguments

None

Response

The listFolders call returns an array of FolderResult objects. A FolderResult object has a single property.

Name	Type	Description
name	string	Folder name.

List Management API calls

The List Management API calls are:

- MergeListMembers
- MergeListMembersRIID
- DeleteListMembers
- RetrieveListMembers

MergeListMembers

Syntax

```
MergeResult[] = service.mergeListMembers(InteractObject list, RecordData recordData,
   ListMergeRule mergeRule)
```

Usage

Use the mergeListMembers call to insert new members or update existing member fields in a given List. Individual invocations of this API call are limited to 200 records. If you need to process more than 200 records, you should place multiple invocations.

» Note Using the OR logical operator will result in an error message.

Request Arguments

Name	Type	Description
list	InteractObject	List object.
recordData	RecordData	Array of RecordData objects that contain field and record data.

Name	Type	Description
mergeRule	ListMergeRule	Defines the merge rules for how to handle the record data.

Response

The MergeResult object that is returned from this call has the following properties:

Name	Type	Description
insertCount	long	Number of records inserted.
updateCount	long	Number of records updated.
rejectedCount	long	Number of records rejected.
totalCount	long	Number of records processed.
errorMessage	string	Error message if applicable.

MergeListMembersRIID

Syntax

```
RecipientResult [] = mergeListMembersRIID(InteractObject list, RecordData recordData,
   ListMergeRule mergeRule)
```

Usage

Use the mergeListMembersRIID call to insert new members or update existing member fields in a given List. Individual invocations of this API call are limited to 200 records. If you need to process more than 200 records, you should place multiple invocations.

» Note Using the OR logical operator will result in an error message.

Request Arguments

Name	Type	Description
list	InteractObject	List object.
recordData	RecordData	Array of RecordData objects that contain field and record data.
mergeRule	ListMergeRule	Defines the merge rules for how to handle the record data.

Response

The RecipientResult object that is returned from this call has the following properties:

Name	Type	Description
recipientId	long	Identifier of the record.

Name	Type	Description
errorMessage	string	Error message if applicable.

DeleteListMembers

Syntax

```
DeleteResult[] = service.deleteListMembers(InteractObject list, QueryColumn
   queryColumn, string[] idsToDelete)
```

Usage

Use the deleteListMembers call to delete members from a List by matching on RIID, CUSTOMER_ID, EMAIL_ADDRESS, or MOBILE_NUMBER fields. Individual invocations of this API call are limited to 200 records. If you need to process more than 200 records, you should place multiple invocations.

Request Arguments

Name	Type	Description
list	InteractObject	List object.
queryColumn	QueryColumn	One value from the QueryColumn list of RIID, CUSTOMER_ID, EMAIL_ADDRESS, or MOBILE_NUMBER. Note: Unlike the system field name, there is NO trailing underscore "_" in the QueryColumn name. For example, if using the Responsys ID, specify "RIID" for the queryColumn value, not the system field name "RIID_" (with trailing underscore).
idsToDelete	string[]	Values for the specified QueryColumn to match for deletion from the List.

Response

The DeleteResult that is returned from this call has the following properties:

Name	Type	Description
id	string	Identifier of the record that was deleted. The identifier value corresponds to the value of the queryColumn that was matched for the deleted record.
success	boolean	Flag indicating whether deletion request was successfully processed.
errorMessage	string	Error message if applicable.

RetrieveListMembers

Syntax

```
RetrieveResult = service.retrieveListMembers(InteractObject list, QueryColumn
   queryColumn, string[] fieldList, string[] idsToRetrieve)
```

Usage

Use the retrieveListMembers call to retrieve fields for individual List members. Individual invocations of this API call are limited to 200 records. If you need to process more than 200 records, you should place multiple invocations.

Request Arguments

Name	Type	Description
List	InteractObject	List object.
queryColumn	QueryColumn	One value from the QueryColumn match options: RIID, CUSTOMER_ID, EMAIL_ADDRESS, or MOBILE_NUMBER. Note: Unlike the system field name, there is NO trailing underscore "_" in the QueryColumn name. For example, if using the Responsys ID, specify "RIID" for the queryColumn value, not the system field name "RIID_" (with trailing underscore).
fieldList	string[]	Fields to retrieve from List member record.
idsToRetrieve	string[]	Values for the specified QueryColumn to match for retrieval from the List.

Response

The RecordData object that is returned from this call has the following properties:

Name	Type	Description
fieldNames	string[]	String array the names of fields returned.
records	Record[]	Record array of the record data returned. The order of the field values returned for each Record is the same order as the fieldNames array.

Table Management API calls

The Table Management calls are:

- CreateProfileExtensionTable
- CreateTable
- CreateTableWithPK
- DeleteProfileExtensionMembers
- DeleteTable
- MergeIntoProfileExtension
- MergeTableRecords
- MergeTableRecordsWithPK
- DeleteTableRecords
- RetrieveTableRecords
- RetrieveProfileExtensionRecords
- TruncateTable

CreateProfileExtensionTable

Syntax

```
boolean = service.createProfileExtensionTable(InteractObject PET, Field[] fields,
   InteractObject list)
```

Usage

Use the CreateProfileExtensionTable call to create a profile extension table with a user-defined schema for a specific profile list table.

Request Arguments

Name	Type	Description
PET	InteractObject	Profile Extension Table object.
fields	Field []	Fields to create. You can also specify data extraction keys via the fields array.
list	InteractObject	Profile list table to be used as parent of this profile extension table.

Response

Name	Type	Description
result	boolean	Success flag for profile extension table creation request.

CreateTable

Syntax

```
boolean = service.createTable(InteractObject table, Field[] fields)
```

Usage

Use the createTable call to create a table with a user-defined schema. Tables can be used in a variety of ways, ranging from use as a source of supplemental data to a List, related to the List through *data extraction key* field(s), as a lookup table for generating dynamic content in a campaign message, or as a form response table.

Request Arguments

Name	Type	Description
table	InteractObject	Table object.
fields	Field []	Fields to create. You can also specify data extraction keys via the fields array.

Response

Name	Type	Description
result	boolean	Success flag for table creation request.

CreateTableWithPK

Syntax

```
boolean = service.createTableWithPK (InteractObject table, Field[] fields, String[]
   primaryKeys)
```

Usage

Use this function to create a supplemental table with a user-defined schema and designate a set of one or more fields as the table's primary key.

Request Arguments

Name	Type	Description
table	InteractObject	Table object.
fields	Field[]	Fields to create. You can also specify data extraction keys via the fields array.
primaryKeys	String[]	An array containing the names of fields that define the primary key of the table.

Response

Name	Type	Description
result	boolean	Success flag for table creation request.

DeleteProfileExtensionMembers

Syntax

```
DeleteResult[] = service.deleteProfileExtensionMembers (InteractObject listExt,
    QueryColumn queryColumn, string[] idsToDelete)
```

Usage

Use the deleteProfileExtensionMembers call to delete members from a Profile Extension Table by matching on RIID, CUSTOMER_ID, EMAIL_ADDRESS, or MOBILE_NUMBER fields from the parent list. Individual invocations of this API call are limited to 200 records. If you need to process more than 200 records, you should place multiple invocations.

Request Arguments

Name	Type	Description
listExtension	InteractObject	Profile Extension object.
queryColumn	QueryColumn	One of the following values from the QueryColumn list: RIID CUSTOMER_ID EMAIL_ADDRESS MOBILE_NUMBER Note: Unlike the system field name, there is NO trailing underscore "_" in the QueryColumn name. For example, if using the Responsys ID, specify "RIID" for the queryColumn value, not the system field name "RIID_" (with trailing underscore).
idsToDelete	String[]	Values for the specified QueryColumn to match.

Response

The DeleteResult that is returned from this call has the following properties:

Name	Type	Description
id	String	Identifier of the record that was deleted. The identifier value corresponds to the value of the queryColumn that was matched for the deleted record.
success	boolean	Flag indicating whether the deletion request was successfully processed.
errorMessage	String	Error message, if applicable.

DeleteTable

Syntax

```
boolean = service.deleteTable(InteractObject table)
```

Usage

Use the deleteTable call to delete a table from your account.

Request Arguments

Name	Type	Description
table	InteractObject	Table object.

Response

Name	Type	Description
result	boolean	Success flag for table deletion request.

MergeIntoProfileExtension

Syntax

```
RecipientResult[] = service.mergeIntoProfileExtension(InteractObject profileExtension,
    RecordData recordData, QueryColumn queryColumn, boolean insertOnNoMatch,
    UpdateOnMatch updateOnMatch)
```

Usage

Use the MergeIntoProfileExtension call to insert or update records in a Profile Extension Table. Individual invocations of this API call are limited to 200 records. If you need to process more than 200 records, you should place multiple invocations.

Request Arguments

Name	Type	Description
profileExtension	InteractObject	profileExtension contains two fields: String folderName & String objectName. The objectName in this case is the name of the Profile Extension Table.
recordData	RecordData	Array of RecordData objects that contain field and record data.
matchColumn	QueryColumn	Column for which a match attempt should be attempted as part of the merge operation.
insertOnNoMatch	boolean	Indicates what should be done for records where a match is not found (true = insert / false = no insert).

Name	Type	Description
updateOnMatch	UpdateOnMatch	Controls how the existing record should be updated.

Response

A RecipientResult object having the following properties is returned from this call:

Name	Type	Description
recipientId	long	Identifier of the record.
errorMessage	string	Error message if applicable.

MergeTableRecords

Syntax

```
MergeResult[] = service.mergeTableRecords(InteractObject table, RecordData records,
    string[] matchColumnNames)
```

Usage

Use the mergeTableRecords call to insert or update records in a table. Individual invocations of this API call are limited to 200 records. If you need to process more than 200 records, you should place multiple invocations.

Request Arguments

Name	Type	Description
table	InteractObject	Table object.
records	RecordData	RecordData object that contains field and record data for the merge operation. Note: For supplemental tables, fieldNames specified in the RecordData object are case-sensitive. They must match their case as defined in the supplemental table.
matchColumnNames	string[]	Column for which a match attempt should be attempted as part of the merge operation. If there is a match for with an existing record, that record will be updated. If there is not a match, then a new record is inserted. Currently only a single match column can be used. So the length of the matchColumnNames string array is limited to one.

Response

A MergeResult object having the following properties is returned from this call:

Name	Type	Description
insertCount	long	Number of records inserted.
updateCount	long	Number of records updated.
rejectedCount	long	Number of records rejected.
totalCount	long	Number of records processed.
errorMessage	string	Error message if applicable.

MergeTableRecordsWithPK

Syntax

```
MergeResult[] = service.mergeTableRecordsWithPK (InteractObject table, RecordData
   recordData, boolean insertOnNoMatch, UpdateOnMatch updateOnMatch)
```

Usage

Use this function to update or insert data into a supplemental table that has a primary key.

Request Arguments

Name	Type	Description
table	InteractObject	Table object.
recordData	RecordData	Array of RecordData objects that contain field and record data. Note: For supplemental tables, fieldNames specified in the RecordData object are case-sensitive. They must match their case as defined in the supplemental table.
insertOnNoMatch	boolean	Indicates what should be done for records where a match is not found (true = insert / false = no insert).
updateOnMatch	UpdateOnMatch	Controls how the existing record should be updated.

» Note This API call doesn't have a match column because the primary key of the table is used as the match column. If a primary key is not defined for the table, an error message is returned.

Response

A MergeResult object having the following properties is returned from this call:

Name	Type	Description
insertCount	long	Number of records inserted.
updateCount	long	Number of records updated.
rejectedCount	long	Number of records rejected.
totalCount	long	Number of records processed.
errorMessage	string	Error message if applicable.

DeleteTableRecords

Syntax

```
DeleteResult[] = service.deleteTableRecords(InteractObject table, string queryColumn,
    string[] idsToDelete)
```

Usage

Use the deleteTableRecords call to delete records from a table. Individual invocations of this API call are limited to 200 records. If you need to process more than 200 records, you should place multiple invocations.

Request Arguments

Name	Type	Description
table	InteractObject	Table object
queryColumn	string	Column for which a match attempt should be attempted as part of the delete operation. If there is a match for with an existing record, that record will be deleted. If there is no match, then no record will be deleted and the success flag of the corresponding DeleteResult object will be set to false.
idsToDelete	string[]	Values for the specified QueryColumn to match for deletion from the table.

Response

The DeleteResult that is returned from this call has the following properties:

Name	Type	Description
id	string	Identifier of the record that was deleted. This identifier corresponds to the queryColumn value of the record.
success	boolean	Flag indicating whether the deletion request was successfully processed.

Name	Type	Description
errorMessage	string	Error message, if applicable.

RetrieveTableRecords

Syntax

```
RetrieveResult = service.retrieveTableRecords(InteractObject table, string queryColumn,
    string[] fieldList, string[] idsToRetrieve)
```

Usage

Use the retrieveTableRecords call to retrieve fields for individual table records. Individual invocations of this API call are limited to 200 records. If you need to process more than 200 records, you should place multiple invocations.

Request Arguments

Name	Type	Description
table	InteractObject	Table object.
queryColumn	string	Column name that will be queried for the idsToRetrieve values provided in this call. An index should be placed on the column used for retrieve queries.
fieldList	string[]	Fields to retrieve from table record.
idsToRetrieve	string[]	Values for the specified QueryColumn to match for retrieval from the List.

Response

The RecordData object that is returned from this call has the following properties:

Name	Type	Description
fieldnames	string[]	String array the names of fields returned.
Records	Record[]	Record array of the record data returned. The order of the field values returned for each Record is the same order as the fieldNames array.

RetrieveProfileExtensionRecords

Syntax

```
RetrieveResult = service.retrieveProfileExtensionRecords (InteractObject listExtension,
    QueryColumn queryColumn, String[] fieldList, String[] idsToRetrieve)
```

Usage

Use the retrieveProfileExtensionRecords call to retrieve fields for individual table records in a profile extension table (PET).

Request Arguments

Name	Type	Description
listExtension	InteractObject	Profile extension table object.
queryColumn	QueryColumn	Column name that will be queried for the idsToRetrieve values provided in this call. » Note Only the RIID column is supported at this time.
fieldList	string[]	Fields to retrieve from table record.
idsToRetrieve	string[]	Values for the specified QueryColumn to be matched when retrieving records from the table.

Response

The RecordData object that is returned from this call has the following properties:

Name	Type	Description
fieldnames	string[]	String array the names of fields returned.
Records	Record[]	Record array of the record data returned. The order of the field values returned for each Record is the same order as the fieldNames array.

TruncateTable

Syntax

```
boolean = service.truncateTable(InteractObject table)
```

Usage

Use the truncateTable call to remove all the records from a table.

Request Arguments

Name	Type	Description
folderName	string	Name of folder containing table to truncate.
tableName	string	Name of table to truncate.

Response

Name	Type	Description
result	boolean	Success flag for truncating a table.

Content Management API calls

The Content Management calls are:

- CopyContentLibraryItem
- CreateContentLibraryItem
- CreateDocument
- DeleteContentLibraryItem
- DeleteDocument
- GetContentLibraryItem
- GetDocumentContent
- GetDocumentImages
- MoveContentLibraryItem
- SetDocumentContent
- SetDocumentImages
- UpdateContentLibraryItem

CopyContentLibraryItem

Syntax

```
boolean = service.copyContentLibraryItem (String srcPath, String dstPath)
```

Usage

Use the copyContentLibraryItemcall to copy a Content Library item to a new location.

Request Arguments

Name	Type	Description
srcPath	string	Location from which to copy.
dstPath	string	Location to which to copy.

Response

Name	Type	Description
result	boolean	True if the item was copied.

CreateContentLibraryItem

Syntax

```
boolean = service.createContentLibraryItem (String folderName, String objectName,
    ItemData itemData)
```

Usage

Use the createContentLibraryItem call to create an item in the Content Library.

Request Arguments

Name	Type	Description
folderName	string	Folder in which to create the item.
objectName	string	Name of the item to create.
itemData	ItemData	The files to upload.

Response

Name	Type	Description
result	boolean	True if the item was created.

CreateDocument

Syntax

```
boolean = service.createDocument(String folderName, String documentName, String
    content, String charset)
```

Usage

Use the createDocument call to create new documents in an Oracle Responsys account. If the document contains relative references to images that should be hosted by Oracle Responsys, then the setDocumentImages call should be made to upload the corresponding image files.

For documents in the Content Library, a full Content Library folder path is required.

Request Arguments

Name	Type	Description
folderName	string	Folder in which to create the document.
documentName	string	Name of the document to create.
content	string	Text content of the document (including markup for HTML content).
charaset	string	Character set of document content.

Response

Name	Type	Description
result	boolean	Flag indicating success of create document request.

DeleteContentLibraryItem

Syntax

```
boolean = service.deleteContentLibraryItem (String folderName, String objectName)
```

Usage

Use the deleteContentLibraryItem call to delete an item from the Content Library.

Request Arguments

Name	Type	Description
folderName	string	Folder containing the item to delete.
objectName	string	Name of the item to delete.

Response

Name	Type	Description
result	boolean	True if the item was deleted.

DeleteDocument

Syntax

```
boolean = service.deleteDocument(String folderName, String documentName)
```

Usage

Use the deleteDocument call to delete a document from an Oracle Responsys account.

For documents in the Content Library, a full Content Library folder path is required.

Request Arguments

Name	Type	Description
folderName	string	Folder containing the document to delete.
documentName	string	Name of the document to delete.

Response

Name	Type	Description
Result	boolean	Flag indicating success of delete document request.

GetContentLibraryItem

Syntax

```
ItemData =  service.getContentLibraryItem (String folderName, String objectName)
```

Usage

Use the getContentLibraryItem call to retrieve the content of a Content Library item.

Request Arguments

Name	Type	Description
folderName	string	Folder containing the item to retrieve.
objectName	string	Name of the item to retrieve.

Response

Name	Type	Description
result	ItemData	The binary data.

GetDocumentContent

Syntax

```
ContentResult = service.getDocumentContent(String folderName, String
   documentName)
```

Usage

Use the getDocumentContent call to obtain the text/markup content of a document object.

For documents in the Content Library, a full Content Library folder path is required.

Request Arguments

Name	Type	Description
folderName	string	Folder containing the document.
documentName	string	Name of the document.

Response

A ContentResult object is returned. This object has the following properties.

Name	Type	Description
Content	string	Text content of document.
Format	ContentFormat	Type of content: HTML or TEXT.
characterEncoding	CharacterEncoding	Character set of document content.

GetDocumentImages

Syntax

```
ImageData[] = service.getDocumentImages(String folderName, String documentName)
```

Usage

Use the getDocumentImages call to retrieve the image file content for a document object.

For documents in the Content Library, a full Content Library folder path is required.

Request Arguments

Name	Type	Description
folderName	string	Folder containing the document.
documentName	string	Name of the document.

Response

Name	Type	Description
Result	ImageData[]	Array of ImageData objects corresponding to each image file to be uploaded. The ImageData object has a string property for the image name and a base64binary representation of the image content.

MoveContentLibraryItem

Syntax

```
boolean = service.moveContentLibraryItem (String srcPath, String dstPath)
```

Usage

Use the moveContentLibraryItem call to move a Content Libraray item to a new location.

Request Arguments

Name	Type	Description
srcPath	string	Location from which to move.
dstPath	string	Location to which to move.

Response

Name	Type	Description
result	boolean	True if the item was moved.

SetDocumentContent

Syntax

```
boolean = service.setDocumentContent(String folderName, String documentName,
    String content)
```

Usage

Use the setDocumentContent call to change the text content of a document object.

For documents in the Content Library, a full Content Library folder path is required.

Request Arguments

Name	Type	Description
folderName	string	Folder containing the document.
documentName	string	Name of the document.
content	string	Text content to set for existing document.

Response

Name	Type	Description
result	boolean	Flag indicating success of set content request.

SetDocumentImages

Syntax

```
CommonResult = service.setDocumentImages(String folderName, String
    documentName,ImageData[] imageData)
```

Usage

Use the setDocumentImages call to upload images files for a document.

For documents in the Content Library, a full Content Library folder path is required.

Request Arguments

Name	Type	Description
folderName	string	Folder containing the document.
documentName	string	Name of the document.
imageData	ImageData[]	Array of ImageData objects corresponding to each image file to be uploaded. The ImageData object has a string property for the image name and a base64binary representation of the image content.

Response

Name	Type	Description
result	boolean	Flag indicating success of set images request.

UpdateContentLibraryItem

Syntax

```
boolean = service.updateContentLibraryItem (String folderName, String objectName,
    ItemData itemData)
```

Usage

Use the updateContentLibraryItem call to update a jpg, gif, png, pdf, tif, or swf item in the Content Library.

Request Arguments

Name	Type	Description
folderName	string	Folder contatining the item to update.
objectName	string	Name of the item to update.
itemData	ItemData	The data to update.

Response

Name	Type	Description
result	boolean	True if the item was updated.

Campaign Management API calls

The Campaign Management calls are:

- GetLaunchStatus
- LaunchCampaign
- MergeTriggerEmail
- MergeTriggerSMS
- MergeTriggerSMS
- TriggerCustomEvent
- TriggerCampaignMessage

GetLaunchStatus

Syntax

```
LaunchStatusResult[] = service.getLaunchStatus(long[] launchIds)
```

Usage

Use the getLaunchStatus call to retrieve launch information for one or more launch identifiers.

Request Arguments

Name	Type	Description
launchIds	long[]	An array of launch identifiers which may have been retrieved and persisted by several possible previous API calls in the client application.

Response

An array of LaunchStatusResult objects is returned. The LaunchStatusResult object has the following properties:

Name	Type	Description
launchId	long	Launch identifier
launchState	string	Launch State: ■ PENDING ■ LAUNCHING ■ USER_PAUSE ■ USER_ABORT ■ SYSTEM_PAUSE ■ SYSTEM_ABORT ■ DONE
launchType	string	Launch Type: ■ PROOF ■ STANDARD
launchDate	dateTime	Timestamp for when launch was initiated.
campaign	InteractObject	Campaign object

LaunchCampaign

Syntax

```
LaunchResult = service.launchCampaign(InteractObject campaign, ProofLaunchOptions
    proofLaunchOptions, LaunchPreferences launchPreferences)
```

Usage

Use the launchCampaign to immediately initiate a campaign launch. A numeric launch identifier is returned from this call and allows for the monitoring of the launch status.

Request Arguments

Name	Type	Description
Campaign	InteractObject	Campaign object reference.
proofLaunch Options	ProofLaunch Options	For proof launches, specify several options as properties of the ProofLaunchOptions object: **proofEmailAddress**: comma separated email address(es) to send proof launches to **proofLaunchType**: ■ LAUNCH_TO_ADDRESS ■ LAUNCH_TO_PROOFLIST ■ LAUNCH_TO_ADDRESS_USING_PROOFLIST
launch Preferences	Launch Preferences	LaunchPreference object properties include: boolean enableLimit int recipientLimit boolean enableNthSampling int samplingNthSelection int samplingNthInterval int samplingNthOffset boolean enableProgressAlerts string progressEmailAddresses string progressChunk progressChunk is limited to one of the following values: ■ CHUNK_10K ■ CHUNK_50K ■ CHUNK_100K ■ CHUNK_500K ■ CHUNK_1M

Response

Returns a LaunchResult which contains the following properties:

Name	Type	Description
launchId	long	Launch identifier.

MergeTriggerEmail

Syntax

```
TriggerResult[] = service.mergeTriggerEmail(RecordData recordData,  ListMergeRule
   mergeRule,  InteractObject campaign,  TriggerData[] triggertData)
```

Usage

Use the mergeTriggerEmail function to merge member(s) into the profile list and subsequently trigger email message(s) to the merged member(s) all in a single call. Responsys email campaigns that already exist can be sent to up to 200 members of a profile list.

Request Arguments

Name	Type	Description
recordData	RecordData	Array of RecordData objects that contain field and record data.
mergeRule	ListMergeRule	Defines the merge rules for how to handle the record data.
campaign	InteractObject	Campaign name and folder.
triggerData	TriggerData[]	An array of TriggerData objects that consists of an OptionalData object array.

Response

The MergeTriggerEmail call returns an array of TriggerResult objects. This object has the following properties:

Name	Type	Description
recipientId	Long	Oracle Responsys internal recipient ID (RIID_) for the individual to whom the message was sent.
success	Boolean	Success flag for trigger message request.
errorMessage	String	NO_RECIPIENT_FOUND MULTIPLE_RECIPIENTS_FOUND

MergeTriggerSMS

Syntax

```
TriggerResult[] = service.mergeTriggeSMS(RecordData recordData, ListMergeRule
    mergeRule, InteractObject campaign, TriggerData[] triggertData)
```

Usage

Use the mergeTriggerSMS function to merge member(s) into the profile list and subsequently trigger SMS message(s) to the merged member(s) all in a single call. Responsys SMS campaigns that already exist can be sent to up to 200 members of a profile list.

Request Arguments

Name	Type	Description
recordData	RecordData	Array of RecordData objects that contain field and record data. MOBILE_NUMBER is required and must be specified in the E.164 format.
mergeRule	ListMergeRule	Defines the merge rules for how to handle the record data.
campaign	InteractObject	Campaign name and folder. NOTE: The Responsys user who creates the campaign must use the campaign template "Direct API Notification" and must activate it by clicking the Activate button on the Summary page of the SMS Campaign wizard. If the Responsys user chooses a different template, such as "Broadcast," then the system returns an "INVALID_CAMPAIGN" message.
triggerData	TriggerData[]	An array of TriggerData objects that consists of an OptionalData object array.

Response

The MergeTriggerSMS call returns an array of TriggerResult objects. This object has the following properties:

Name	Type	Description
recipientId	Long	Oracle Responsys internal recipient ID (RIID_) for the individual to whom the message was sent.
success	Boolean	Success flag for trigger message request.
errorMessage	String	NO_RECIPIENT_FOUND MULTIPLE_RECIPIENTS_FOUND

ScheduleCampaignLaunch

Syntax

```
boolean = service.scheduleCampaignLaunch(InteractObject campaign,
   ProofLaunchOptions proofLaunchOptions, LaunchPreferences launchPreferences,
   dateTime scheduleDate)
```

Usage

Use the scheduleLaunch call to schedule the launch of a campaign at some future point in time.

Request Arguments

Name	Type	Description
campaign	InteractObject	Campaign object reference.
proofLaunch Options	ProofLaunch Options	Leave null for standard launches. For proof launches, specify several options as properties of the ProofLaunchOptions object: **proofEmailAddress**: comma separated email address(es) to send proof launches to **proofLaunchType**: LAUNCH_TO_ADDRESS LAUNCH_TO_PROOFLIST LAUNCH_TO_ADDRESS_USING_PROOFLIST
launch Preferences	Launch Preferences	LaunchPreference object properties include: boolean enableLimit int recipientLimit boolean enableNthSampling int samplingNthSelection int samplingNthInterval int samplingNthOffset boolean enableProgressAlerts string progressEmailAddresses string progressChunk progressChunk is limited to one of the following values: ■ CHUNK_10K ■ CHUNK_50K ■ CHUNK_100K ■ CHUNK_500K ■ CHUNK_1M
scheduleDate	dateTime	Date and time for launch.

Response

Name	Type	Description
result	boolean	Flag for the success of the launch request.

TriggerCustomEvent

Syntax

```
TriggerResult[] = service.triggerCustomEvent(CustomEvent customEvent, RecipientData[]
    recipientData)
```

Usage

Use the triggerCustomEvent call to trigger a Custom Event for a recipient. The Oracle Responsys platform provides Custom Event listeners that will respond to a triggered Custom Event in several possible ways depending on the specific definition and use of Custom Events in your Oracle Responsys account. Some Custom Events provide an entry point into one or more Programs. Other Custom Events can be used for segmentation purposes. See the Oracle Responsys platform documentation for more information on the use of Custom Events.

A single triggerCustomEvent request is limited to 200 recipients. If you need to trigger a Custom Event for more than 200 recipients, then you should place multiple triggerCustomEvent requests.

» Note Sending duplicate names in the recipientData would result in an error message.

Request Arguments

Name	Type	Description
customEvent	CustomEvent	The CustomEvent to be triggered. The CustomEvent eventName or eventId property must be specified for this object.

Name	Type	Description
recipientData	RecipientData[]	An array of RecipientData objects that define the recipients for whom a custom event should be triggered. A RecipientData object consists of a Recipient object and an OptionalData object array. At least one of the following List member identifiers should be provided in the Recipient object (recipientId, emailAddress, customerId, or mobileNumber). If you specify more than one of these values, we process them in this order—recipientId, emailAddress, customerId, or mobileNumber—and we take the first non-null value. For example, if you specify emailAddress and customerId, we only take the emailAddress (unless there are no email addresses).

Response

The triggerCustomEvent call returns an array of TriggerResult objects. The TriggerResult object has the following properties.

Name	Type	Description
recipientId	long	Oracle Responsys internal recipient ID (RIID_) for the individual to whom the message was sent.
success	boolean	Success flag.
errorMessage	string	NO_RECIPIENT_FOUND MULTIPLE_RECIPIENTS_FOUND

TriggerCampaignMessage

Syntax

```
TriggerResult[] = service.triggerCampaignMessage(InteractObject campaign,
   RecipientData[] recipientData)
```

Usage

Use the triggerCampaignMessage call to send email messages to one or more recipients. A single triggerCampaignMessage request is limited to 200 recipients. If you need to trigger to a message to more than 200 recipients, then you should execute multiple triggerCampaignMessage requests.

» Note Sending duplicate names in the recipientData would result in an error message.

Request Arguments

Name	Type	Description
campaign	InteractObject	Campaign name.
recipientData	RecipientData[]	An array of RecipientData objects that define the recipients to whom a campaign message should be sent. A RecipientData object consists of a Recipient object and an OptionalData object array. NOTE: This call uses recipientData only to look up a recipient in the list. This means that if you want to change any data, for example, use a specific email format, you must update the user record before making this call. At least one of the following List member identifiers should be provided in the Recipient object (recipientId, emailAddress, customerId, or mobileNumber). If you specify more than one of these values, we process them in this order—recipientId, emailAddress, customerId, or mobileNumber—and we take the first non-null value. For example, if you specify emailAddress and customerId, we only take the emailAddress (unless there are no email addresses). The Recipient object List property is optional for this call since a valid campaign already has a reference to an existing List. The array of OptionalData objects define name/value pairs that can be used for dynamic content in the campaign message template.

Response

The triggerCampaignMessage call returns an array of TriggerResult objects. This object has the following properties.

Name	Type	Description
recipientId	Long	Oracle Responsys internal recipient ID (RIID_) for the individual to whom the message was sent.
success	Boolean	Success flag for trigger message request.
errorMessage	String	NO_RECIPIENT_FOUND MULTIPLE_RECIPIENTS_FOUND

Interact Data Types

The Interact uses the standard data types defined below. These data types conform to their specifications in the World Wide Web Consortium's publication "XML Schema Part 2: Data Types" (available at http://www.w3.org/TR/xmlschema-2). Data types are used as a standardized way to define, send, receive, and interpret basic data types in the SOAP messages exchanged between client applications and the Interact.

Type	Description
boolean	Boolean fields have one of these values: true (or 1), or false (or 0).
string	Character string data types contain text data. In some cases, strings are enumerated; that is, the text data values are restricted to a specific set of expected values.
int and long	Fields of these types contain integers (long ranges from 9223372036854775807 to -9223372036854775808 and int ranges from 2147483647 to -2147483648.
dateTime	Fields defined as dateTime data types handle date/time values (timestamps). Regular dateTime fields are full timestamps with a precision of one second.

Interact Objects

These are the Interact objects you can use.

- CharacterEncoding
- ContentFormat
- CustomEvent
- DeleteResult
- EmailFormat
- Field
- FolderResult
- ImageData
- InteractObject
- LaunchPreferences
- LaunchResult
- ListMergeRule
- LoginResult
- MatchOperator
- MergeResult
- OptionalData
- ProofLaunchOptions
- ProofLaunchType
- QueryColumn
- Recipient
- RecipientData
- RecipientResult
- Record
- RecordData
- ServerAuthResult
- TriggerData
- TriggerResult
- UnsubscribeOption
- UpdateOnMatch

CharacterEncoding

The CharacterEncoding is a string restricted to one of the values listed below.

Type	Values	
string	ISO_8859_1	SJIS
	windows_1257	euc_kr
	ISO_8859_2	koi8_r
	gb2312	ISO_8859_9
	big5	UTF_8
	ISO_8859_7	

ContentFormat

The ContentFormat is a string restricted to one of the values listed below.

Type	Values	
string	HTML	TEXT

CustomEvent

The CustomEvent object contains information needed for the triggerCustomEvent call.

Name	Type	Description
eventName	string	Name of the Custom Event.
eventId	long	Identifier for Custom Event. Either eventName or eventId of the Custom Event Type should be specified.

DeleteResult

The DeleteResult object represents the response from a delete request.

Name	Type	Description
Id	string	Identifier of the record that was deleted.
Success	boolean	Flag indicating whether the deletion request was successfully processed.
errorMessage	string	Error message, if applicable.

EmailFormat

The EmailFormat is a string restricted to one of the values listed below.

Type	Values	
String	TEXT_FORMAT	MULTIPART_FORMAT
	HTML_FORMAT	NO_FORMAT

Field

The Field object represents a field (or column) in a List or Table.

Name	Type	Description
fieldName	string	Name of field.
fieldType	FieldType	Data type of field.
Custom	boolean	Flag indicating whether this represents a custom field. This is a read-only variable that is used only in the describeObjects API.
dataExtractionKey	boolean	Flag indicating whether this field is a data extraction key.

FieldType

The FieldType is a string restricted to one of following values.

Type	Values	
String	STR500	NUMBER
	STR4000	TIMESTAMP
	INTEGER	

FolderResult

The Folder object has a single property that defines the name of a folder.

Name	Type	Description
Name	string	Folder name.

ImageData

The imageData object represents an image file.

Name	Type	Description
imageName	string	Name of image.
image	base64binary	base64binary representation of binary image content.

InteractObject

Name	Type	Description
folderName	string	Name of folder.
objectName	string	Name of object.

LaunchPreferences

The LaunchPreferences object defines the behavior of the launch.

Name	Type	Description
enableLimit	boolean	Enable limit for launch.
recipientLimit	int	Limit launch to a certain number of recipients.
enableNthSampling	int	Enable Nth sampling.
samplingNthSelection	int	Selection for Nth sampling.
samplingNthInterval	int	Interval for Nth sampling.
samplingNthOffset	int	Offset for Nth sampling.
enableProgressAlerts	boolean	Enable launch progress alerts.
progressEmailAddress	string	Email address to sent progress alerts.
progressChunk	string	Send progress alerts after the launch of a given number of recipients. progressChunk is limited to one of the following values: ■ CHUNK_10K ■ CHUNK_50K ■ CHUNK_100K ■ CHUNK_500K ■ CHUNK_1M

LaunchResult

The LaunchResult object contains information about a campaign launch.

Name	Type	Description
launchId	long	Launch identifier.

ListMergeRule

The ListMergeRule object represents the rules by which incoming List records are processed for merging into a List.

Name	Type	Description
insertOnNoMatch	boolean	Indicates what should be done for records where a match is not found (true = insert / false = no insert).
updateOnMatch	UpdateOnMatch	Controls how the existing record should be updated.
matchColumnName1	string	First match column for determining whether an insert or update should occur.
matchColumnName2	string	Second match column for determining whether an insert or update should occur (optional).
matchOperator	MatchOperator	Controls how the boolean expression involving the match columns is constructed to determine a match between the incoming records and existing records.
optinValue	string	Value of incoming opt-in status data that represents an opt-in status. For example, *1* may represent an opt-in status.
optoutValue	string	Value of incoming opt-out status data that represents an opt-out status. For example, *0* may represent an opt-out status.

Name	Type	Description
defaultPermissionStatus	enum	This value must be specified as either OPTIN or OPTOUT and would be applied to all of the records contained in the API call. If this value is not explicitly specified, then it is set to OPTOUT.
htmlValue	string	Value of incoming preferred email format data. For example, *H* may represent a preference for HTML formatted email.
textValue	string	Value of incoming preferred email format data. For example, *T* may represent a preference for Text formatted email.
rejectRecordIfChannelEmpty	string	String containing comma-separated channel codes that if specified will result in record rejection when the channel address field is null. Channel codes are 'E' (Email), 'M' (Mobile), 'P' (Postal Code). For example 'E,M' would indicate that a record that has a null for Email or Mobile Number value should be rejected. This parameter can also be set to null or to an empty string, which will cause the validation to not be performed for any channel, except if the matchColumnName1 parameter is set to EMAIL_ADDRESS or MOBILE_NUMBER. When matchColumnName1 is set to EMAIL_ADDRESS or MOBILE_NUMBER, then the null or empty string setting is effectively ignored for that channel.

LoginResult

The LoginResult object has a single property that defines the session ID for a client application session.

Name	Type	Description
sessionId	string	Valid session ID for use in subsequent API calls. This session ID should be placed in the SOAP header for subsequent calls.

MatchOperator

The MatchOperator is a string restricted to one of the values listed below.

Type	Values	
string	NONE	AND

MergeResult

The MergeResult object represents the response from a merge request.

Name	Type	Description
insertCount	long	Number of records inserted.
updateCount	long	Number of records updated.
rejectedCount	long	Number of records rejected.
totalCount	long	Number of records processed.
errorMessage	string	Error message if applicable.

OptionalData

The OptionalData object contains name/value pair data that can be used in a variety of ways ranging from optional campaign variables to Program enactment variables.

» Note To pass extended/accented characters in optionalData payload, they must be escaped as Unicode characters. For example, the euro symbol € is escaped as `\u20AC`, the yen symbol ¥ is escaped as `\u00A5`, an ü is escaped as `\u00FC`, an é is escaped as `\u00E9`, and the like. Otherwise, you may receive an INVALID_REQUEST_CONTENT error.

Name	Type	Description
Name	string	Name of variable.
Value	string	Value of variable.

ProofLaunchOptions

The.ProofLaunchOptions object defines how a proof launch should be conducted.

Name	Type	Description
proofEmailAddress	string	String of comma-separated email addresses.
proofLaunchType	ProofLaunchType	Object that defines the nature of the proof launch.

ProofLaunchType

The ProofLaunchType is a string restricted to one of the values listed below:

Type	Values
string	LAUNCH_TO_ADDRESS
	LAUNCH_TO_LIST
	LAUNCH_TO_ADDRESS_USING_LIST

QueryColumn

The QueryColumn is a string restricted to one of the values listed below. Note that there are no trailing underscores used in these values (“RIID”, not “RIID_” as used in the system field name)

Type	Values	
string	RIID	EMAIL_ADDRESS
	CUSTOMER_ID	MOBILE_NUMBER

Recipient

The Recipient object has the following properties. At least one of the Recipient identifiers should be used to uniquely target a recipient: recipientId, customerId, emailAddress, or mobileNumber.

Name	Type	Description
listName	string	Name of list for recipient.
recipientId	long	Internal Oracle Responsys ID (RIID_) for recipient.
customerId	string	Externally defined customer ID.
emailAddress	string	Email address.
mobileNumber	string	Mobile number.
emailFormat	EmailFormat	Format of message to deliver to the recipient (optional).

RecipientData

The RecipientData object has the following properties. It is used to represent a List member and a number of name/value pair parameters needed for triggering messages or custom events.

Name	Type	Description
recipient	Recipient	Identity of a List member.
optionalData	OptionalData[]	Optional name/value pair parameters associated with this List member. Note: To include extended/accented characters in the payload, they must be escaped as Unicode characters.

RecipientResult

The RecipientResult object has the following properties. It returns an array of RecipientResult objects that each contain a recipientID and an errorMessage.

Name	Type	Description
recipientId	long	Identifier of the record.
errorMessage	string	Error message if applicable.

Record

The Record object represents a record of data from a List or Table.

Name	Type	Description
fieldValues	string[]	A string array representing the values of fields in a record.

RecordData

The RecordData object represents a number of records of data from a List or Table.

Name	Type	Description
fieldNames	string[]	An array containing the field names in a record of data.
mapTemplateName	string[]	Optional parameter to use an existing data mapping to import data into a table.
records	Record[]	An array of Record objects which contain data from a List or Table.

ServerAuthResult

Name	Type	Description
authSessionId	string	Temporary session ID that should be placed in the SOAP header of the subsequent loginWithCertificate call.
encryptedClientChallenge	byte[]	Response to the client challenge, represented by encrypting the client challenge with the server private key. Client applications should validate server authenticity by decrypting this value with the server public key (available through the Oracle Responsys user interface admin console).
serverChallenge	byte[]	Server challenge of client application authenticity. This challenge should be encrypted with the client private key and submitted with the loginWithCertificate call to authenticate the client application session.

TriggerData

The TriggerData object defines an array of optional name/value pairs that can be used for dynamic content in the campaign message.

Name	Type	Description
optionalData	Optional Data[]	Array of OptionalData objects define name/value pairs that can be used for dynamic content in the campaign message. Note: To include extended/accented characters in the payload, they must be escaped as Unicode characters.

TriggerResult

The TriggerResult object defines the results from a trigger request for a campaign message or custom event.

Name	Type	Description
recipientId	long	Oracle Responsys internal recipient ID (RIID_) for the individual to whom the message was sent.
Success	boolean	Success flag.
errorMessage	string	NO_RECIPIENT_FOUND, MULTIPLE_RECIPIENTS_FOUND

UnsubscribeOption

The UnsubscribeOption is a string restricted to one of the values listed below.

Type	Values	
String	NO_OPTOUT_BUTTON	OPTOUT_FORM
	OPTOUT_SINGLE_CLICK	

UpdateOnMatch

The UpdateOnMatch is a string restricted to one of the values listed below.

Type	Values	
String	NO_UPDATE	REPLACE_ALL

Interact Result and Exception Codes

The Interact result and exception codes are divided into these categories.

The Campaign Management calls are:

- General result codes
- Merge and launch failure result codes
- Login failure result codes
- Access exception codes
- Data exception codes
- Campaign and launch exception codes
- TriggerCampaignMessage
- Trigger custom event exception codes
- Create and retrieve Oracle Responsys object exception codes
- General exception codes

Result Codes

General result codes

Code	Description
SUCCESS	Execution was successful.
FAILURE	Execution failed.

Merge and launch failure result codes

Code	Description
NO_RECIPIENT_FOUND	No recipient with the specified parameters is found.
MULTIPLE_RECIPIENTS_FOUND	Multiple recipients with the specified parameters were found.
RECIPIENT_STATUS_UNDELIVERABLE	The requested recipient's status is Undeliverable.
PROFILE_LIST_NOT_FOUND	No profile list with the specified parameter was found.
FOLDER_NOT_FOUND	No folder with the specified parameter was found.
PROFILE_LIST_NOT_FOUND_IN_FOLDER	No list in the folder with the specified parameters was found.

Login failure result codes

Code	Description
INVALID_PASSWORD	The password is invalid.
USER_BLOCKED	The user is blocked.
PASSWORD_EXPIRED	The password expired.
LOGINS_DISABLED	Logins are disabled for the user.

Code	Description
INVALID_AUTH_OPTION	An invalid authorization option.
CLIENT_CERTIFICATE_NOT_FOUND	The client certificate was not found.
CLIENT_CERTIFICATE_EXPIRED	The client certificate expired.
CLIENT_CERTIFICATE_NOT_YET_VALID	The client certificate is not yet valid.
SERVER_CERTIFICATE_NOT_FOUND	The sever certificate was not found.
SERVER_CERTIFICATE_EXPIRED	The server certificate expired.
SERVER_CERTIFICATE_NOT_YET_VALID	The server certificate is not yet valid.
PRIVATE_KEY_NOT_FOUND	The private key was not found.
SERVER_CHALLENGES_DO_NOT_MATCH	Server challenges do not match.
INACTIVE_ACCOUNT	The account is inactive.
MAX_CONCURRENT_SESSIONS_EXCEEDED	A new session cannot be opened because the maximum number of allowed sessions is currently open for the account.
MAX_LOGIN_FAILURES_EXCEEDED	The maximum number of unsuccessful login attempts was exceeded.
LOGIN_BLOCKED_TEMPORARILY	Login for the user is temporarily blocked.

Exception Codes

Access exception codes

Code	Description
API_DISABLED_FOR_USER	The API is disabled for the user.
INSUFFICIENT_ACCESS	The user does not have privileges for the call.
INVALID_USER_NAME	The user name is invalid.
INVALID_PASSWORD	The password is invalid.
INVALID_SESSION_ID	The session ID is invalid.
INVALID_SOAP_HEADER	The SOAP header is invalid.
PASSWORD_LOCKOUT	The user's password is locked.

Code	Description
PASSWORD_EXPIRED	The user's password has expired.
API_LIMIT_EXCEEDED	The request exceeded the maximum number of requests allowed.
API_BLOCKED	The <*function_name*> is currently not available to this user.
OPERATION_NOT_SUPPORTED	The requested operation is not supported.
INVALID_AUTHENTICATION_OPTION	An invalid authentication option.
AUTHENTICATION_FAILED	Authentication failed.
CLIENT_CERTIFICATE_EXPIRED	The client certificate expired.
CLIENT_CERTIFICATE_NOT_YET_VALID	The client certificate is not yet valid.
CLIENT_CERTIFICATE_NOT_FOUND	The client certificate was not found.
SERVER_CERTIFICATE_EXPIRED	The server certificate expired.
SERVER_CERTIFICATE_NOT_YET_VALID	The server certificate is not yet valid.
SERVER_CERTIFICATE_NOT_FOUND	The server certificate was not found.
SERVICE_UNAVAILABLE	The requested service is unavailable.

Data exception codes

Code	Description
INVALID_NUMBER	Invalid number format in a value.
INVALID_DATE	Invalid date format in a value.
INVALID_PARAMETER	General invalid parameter value.
INVALID_FIELD_NAME	The specified field is invalid or does not exist.
INVALID_OBJECT	The object is defined with invalid parameters/values.
RECORD_LIMIT_EXCEEDED	The requested number of records exceeds the maximum record limit.
RECORD_NOT_FOUND	The record was not found.

Campaign and launch exception codes

Code	Description
CAMPAIGN_NOT_FOUND	The specified campaign was not found. Ensure that your folder name and campaign name are correct, and that they follow correct case sensitivity.
CAMPAIGN_ALREADY_EXISTS	A campaign with the specified name already exists.
RECIPIENT_LIMIT_EXCEEDED	The allowable number of recipients is exceeded.
MAX_ATTACHMENT_SIZE_EXCEEDED	The maximum attachment size is exceeded.
PROOF_LIST_NOT_FOUND	The specified proof list was not found.
CAMPAIGN_LAUNCH_IN_PROGRESS	A campaign launch is in progress.
CAMPAIGN_NOT_LISTENING	The campaign is not active.
CAMPAIGN_LAUNCH_NOT_SCHEDULED	A campaign launch is not scheduled.
CAMPAIGN_LAUNCH_NOT_UNSCHEDULED	A campaign launch is not unscheduled.
CAMPAIGN_NOT_APPROVED_FOR_LAUNCH	The campaign is not approved for launch.
SCHEDULED_LAUNCH_NOT_FOUND	The specified scheduled launch is not found.
CAMPAIGN_IS_INVALID	The campaign is invalid.
MOBILE_CAMPAIGN_DISABLED_FOR_USER	Mobile campaigns are disabled for the user.

Trigger custom event exception codes

Code	Description
CUSTOM_EVENT_NOT_FOUND	The specified Custom event was not found.

Create and retrieve Oracle Responsys object exception codes

Code	Description
FOLDER_NOT_FOUND	The specified folder was not found.
FOLDER_ALREADY_EXISTS	The specified folder already exists.
NO_OPEN_LAUNCHES_FOR_THIS_ACCOUNT	No open launches exist for the account.
NO_LAUNCHES_FOR_THIS_CAMPAIGN	No open launches exist for the specified campaign.
NO_CAMPAIGNS_IN_THIS_FOLDER	No campaigns exist in the specified folder.
NO_OBJECTS_IN_THIS_FOLDER	No objects exist in the specified folder.
LIST_NOT_FOUND	The specified list was not found.
LIST_ALREADY_EXISTS	The list with the same name already exists.
LINK_TABLE_NOT_FOUND	The specified link table was not found.
LINK_TABLE_ALREADY_EXISTS	The specified link table already exists.
TABLE_ALREADY_EXISTS	The specified table already exists.
TABLE_NOT_FOUND	The specified table was not found.
OBJECT_NOT_FOUND	The specified object was not found.
OBJECT_ALREADY_EXISTS	The specified object already exists.
MULTIPLE_OBJECTS_FOUND	Multiple objects with the same name were found.
DOCUMENT_NOT_FOUND	The specified document was not found.
DOCUMENT_ALREADY_EXISTS	The specified document already exists.
IMAGES_NOT_FOUND	The specified images was not found.
PROFILE_EXTENSION_NOT_FOUND	The specified profile extension was not found.

General exception codes

Code	Description
UNEXPECTED_ EXCEPTION	An unexpected exception occurred.
UNRECOVERABLE_ EXCEPTION	HATM exception that means a failover is required. You need to re-login and try the call again.

Gateway server error when payload exceeds the maximum allowed size

If your payload exceeds the maximum allowed size, the gateway server will terminate the connection with the client application, and the client application will receive a `SocketException` error. Ensure that your payload size is 10 MB or less. One way to reduce the request payload size is by referencing HTML content instead of sending it as part of the payload.

Sample Code

» Tip If you copy samples out of this PDF document, examine the copy for the following potential issues and fix them: Incorrect characters (for example, “smart quotes”), hidden characters, and the document “footer” text containing the title and page number when you copy across multiple pages. We recommend using a text editor that has a “Show all characters” option, so that you can more easily detect these issues.

Sample Code for Handling Exceeded Account Limits

The following sections provide sample code in Java, C#, and PHP for handling the API_LIMIT_EXCEEDED error that is returned when the account limit for calling an API function is exceeded.

» Note This code example is shown as an example of how to use our API. You could follow a similar logic and use other Responsys API functions.

Sample Java code

```
private void listFolders() {
      try {
         if (!loggedIn) {
            if (!login()) {
               return;
            }
         }
         ListFolders listFolders = new ListFolders();
         ListFoldersResponse listFoldersResponse = stub.listFolders(listFolders, sessionHeader); // stub is
   generated by JAX-WS based client

         FolderResult[] folders = listFoldersResponse.getResult();
         if (folders != null) {
            System.out.println("****************************************");
            System.out.println("List Folders Successful");
            System.out.println("Folders length = " + folders.length);
            for (FolderResult folder : folders) {
               System.out.println("Folder Name = " + folder.getName());
            }
            System.out.println("****************************************");
         }
         else {
            System.out.println("****************************************");
            System.out.println("List Folders Failed");
            System.out.println("****************************************");
         }
      }
      catch (UnexpectedErrorFault unexpectedEx) {
         System.out.println("unexpectedEx listFolders");
         System.out.println("Exception Code = " + unexpectedEx.getFaultMessage().getExceptionCode());
         System.out.println("Exception Msg  = " +
   unexpectedEx.getFaultMessage().getExceptionMessage());
      }
      catch (RemoteException remoteEx) {
         System.out.println("remoteEx listFolders");
         System.out.println("Exception Msg = " + remoteEx.getMessage());
         if (remoteEx instanceof AxisFault) {
            AxisFault axisFault = (AxisFault) remoteEx;
            System.out.println("Fault detail element = "+ axisFault.getFaultDetailElement().getText());
         }
         if ("API_LIMIT_EXCEEDED".equalsIgnoreCase(remoteEx.getMessage())) {
            retryDelay();
```

```
            listFolders();
        }
    }
}

private void retryDelay() {
    int i = 0;
    while (i < 60) { //60 seconds delay
        try {
            System.out.print(". ");
            Thread.sleep(1000);
            i++;
        }
        catch (InterruptedException ex) {
        }
    }
}
```

Sample C# code

```
private void listFolders() {
        try {
            if (!loggedIn) {
                if (!login()) {
                    return;
                }
            }

            FolderResult[] folders = stub.listFolders(); // stub are generated by wsdl tool.
            if (folders != null) {
                Console.WriteLine("*************************************");
                Console.WriteLine("List Folders Successful");
                foreach (FolderResult folder in folders) {
                    Console.WriteLine("Folder Name = " + folder.name);
                }
                Console.WriteLine("*************************************");
            } else {
                Console.WriteLine("*************************************");
                Console.WriteLine("List Folders Failed");
                Console.WriteLine("*************************************");
            }
        } catch (System.Web.Services.Protocols.SoapException e) {
            Console.WriteLine("SoapException in listFolders : " + e.Message);
            Console.WriteLine("SoapException in listFolders : " + e.Detail.InnerText);
            if ("API_LIMIT_EXCEEDED".Equals(e.Message)) {
                Console.WriteLine("The API Limit has Exceeded");
                Thread.Sleep(60000);//need to add using System.Threading;
                listFolders();
            }
        } catch (Exception e) {
            Console.WriteLine("Exception in listFolders : " + e.Message);
        }
    }
```

Sample PHP code

```
function mlistFolders($client) {
    try {
        $listFoldersResult = $client - > listFolders();
        print('<pre>');
        print_r($listFoldersResult);
        print('</pre>');
    } catch (SoapFault $err) {
        if ($err - > faultstring == 'ConnectFault') {
            print "<br>ConnectFault Error";
            print "<br>Exception Message Detail: ".$err - > detail - > ConnectFault - > exceptionMessage.
            "<br>";
        } else if ($err - > faultstring == 'API_LIMIT_EXCEEDED') {
```

```
            print "<br>API LIMIT EXCEEDED";
            print "<br>Exception Message Detail: ".$err - > detail.
            "<br>";
            sleep(60); //60 secs sleep
            mlistFolders($client);
        } else {
            print "Other Exception Error: ".$err - > getMessage().
            "\n";
        }
    }
}
```

Sample Code for Certificate Authentication (Java)

The following code sample demonstrates client code for certificate authentication. See LoginWithCertificate on page 22 for details about how the certificate authentication process works.

```
private void loginUsingCertificate() {
      try {
         String endPointURL = "http://" + hostname + "/webservices/services/ResponsysWSService";
         System.out.println ("WEBSERVICES URL = " + endPointURL);
         stub = new ResponsysWSServiceStub(endPointURL);
         stub._getServiceClient().getOptions().setManageSession(true);

         String username = "apiusername"; // substitute with the API username

         AuthenticateServer authenticateServer = new AuthenticateServer();
         authenticateServer.setUsername(username);

         // Create a Random Number for client challenge.
         Random random = new Random();
         String clientChallengeStr = String.valueOf(random.nextLong());
         byte[] clientChallenge = clientChallengeStr.getBytes();
         authenticateServer.setClientChallenge(clientChallenge);
         System.out.println ("Client challenge created");

         /*****************************************************/
         // Set timeout
         stub._getServiceClient().getOptions().setTimeOutInMilliSeconds(1000*60*60);
         AuthenticateServerResponse response = stub.authenticateServer(authenticateServer);
         ServerAuthResult serverAuth = response.getResult();
         String authSessionId = serverAuth.getAuthSessionId();
         byte[] encryptedClientChallenge = serverAuth.getEncryptedClientChallenge();
         byte[] serverChallenge = serverAuth.getServerChallenge();
         /*****************************************************/

         // Validate the client challenge with Encrypted Client Challenge
         // Using server's public key.
         String serverCertName = "C:\\ResponsysServerCert.cer";
         File certFile = new File(serverCertName);
         if (!certFile.exists()) {
            System.out.println ("Server certificate doesn't exist in that location");
            return;
         }

         try {
            CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
            X509Certificate serverCertificate = (X509Certificate) certFactory.generateCertificate(new
   FileInputStream(certFile));

            Cipher decryptCipher = Cipher.getInstance("RSA");
            decryptCipher.init(Cipher.DECRYPT_MODE, serverCertificate.getPublicKey());
            byte[] decryptedClientChallenge = decryptCipher.doFinal(encryptedClientChallenge);

            // Compare the clientChallenge with decryptedClientChallenge.
            boolean serverValidated = Arrays.equals(clientChallenge, decryptedClientChallenge);
```

```
            if (serverValidated) {
                System.out.println ("Server validation is success ... proceeding further to login to
webservices");
            } else {
                System.out.println ("Server validation failed");
                return;
            }
        } catch (CertificateException ex) {
            System.out.println ("CertificateException : " + ex.getMessage());
            return;
        } catch (NoSuchAlgorithmException ex) {
            System.out.println ("NoSuchAlgorithmException : " + ex.getMessage());
            return;
        } catch (NoSuchPaddingException ex) {
            System.out.println ("NoSuchPaddingException : " + ex.getMessage());
            return;
        } catch (InvalidKeyException ex) {
            System.out.println ("InvalidKeyException : " + ex.getMessage());
            return;
        } catch (BadPaddingException ex) {
            System.out.println ("BadPaddingException : " + ex.getMessage());
            return;
        } catch (IllegalBlockSizeException ex) {
            System.out.println ("IllegalBlockSizeException : " + ex.getMessage());
            return;
        } catch (FileNotFoundException ex) {
            System.out.println ("FileNotFoundException : " + ex.getMessage());
            return;
        }

        // Get the private key of the client certificate
        PrivateKey privateKey = getPrivateKeyForClientCertificate();
        if (privateKey == null) {
            System.out.println ("Couldn't get private key from the client certificate");
            return;
        }

        // Encrypt server challenge using client's private key
        // and invoke the loginWithCertificate method with
        // authSessionId in the header.
        try {
            Cipher encryptCipher = Cipher.getInstance("RSA");
            encryptCipher.init(Cipher.ENCRYPT_MODE, privateKey);
            byte[] encryptedServerChallenge = encryptCipher.doFinal(serverChallenge);

            /*****************************************************/
            AuthSessionHeader authSessionHeader = new AuthSessionHeader();
            authSessionHeader.setAuthSessionId(authSessionId);
            LoginWithCertificate loginWithCertificate = new LoginWithCertificate();
            loginWithCertificate.setEncryptedServerChallenge(encryptedServerChallenge);
            LoginWithCertificateResponse loginResponse = stub.loginWithCertificate(loginWithCertificate,
authSessionHeader);
            LoginResult loginResult = loginResponse.getResult();
            String sessionId = loginResult.getSessionId();
            /*****************************************************/

            System.out.println ("**************************************");
            System.out.println ("Login Result = " + (sessionId != null));
            System.out.println ("**************************************");
            if (sessionId != null) {
                sessionHeader = new SessionHeader();
                sessionHeader.setSessionId(sessionId);

                System.out.println ("Setting the Client Timeout to 1 hour");

                // Set timeout
                stub._getServiceClient().getOptions().setTimeOutInMilliSeconds(1000*60*60);
                loggedIn = true;
            }
        } catch (InvalidKeyException ex) {
```

```
            System.out.println ("InvalidKeyException : " + ex.getMessage());
            return;
        } catch (NoSuchPaddingException ex) {
            System.out.println ("NoSuchPaddingException : " + ex.getMessage());
            return;
        } catch (NoSuchAlgorithmException ex) {
            System.out.println ("NoSuchAlgorithmException : " + ex.getMessage());
            return;
        } catch (IllegalBlockSizeException ex) {
            System.out.println ("IllegalBlockSizeException : " + ex.getMessage());
            return;
        } catch (BadPaddingException ex) {
            System.out.println ("BadPaddingException : " + ex.getMessage());
            return;
        }
    } catch (AccountFault accountEx) {
        System.out.println ("accountEx loginUsingCertificate");
        System.out.println ("Exception Code = " + accountEx.getFaultMessage().getExceptionCode());
        System.out.println ("Exception Msg  = " + accountEx.getFaultMessage().getExceptionMessage());
    } catch (UnexpectedErrorFault unexpectedEx) {
        System.out.println ("unexpectedEx loginUsingCertificate");
        System.out.println ("Exception Code = " + unexpectedEx.getFaultMessage().getExceptionCode());
        System.out.println ("Exception Msg  = " +
 unexpectedEx.getFaultMessage().getExceptionMessage());
    } catch (RemoteException remoteEx) {
        System.out.println ("remoteEx loginUsingCertificate");
        System.out.println ("Exception Msg = " + remoteEx.getMessage());
    }
}

/**
 * Get the PrivateKey of the Client Certificate
 * @return
 */
private PrivateKey getPrivateKeyForClientCertificate() {
    PrivateKey privateKey = null;
    try {
        String keyStoreName = "C:\\responsys\\ClientCert.cer";
        String keyAlias     = "clientcert";
        String keyStorePass = "clientcert";
        char[] passPhrase   = keyStorePass.toCharArray();

        KeyStore keyStore = KeyStore.getInstance("JKS");
        File certificateFile = new File(keyStoreName);
        if (!certificateFile.exists()) {
            return privateKey;
        }
        keyStore.load(new FileInputStream(certificateFile), passPhrase);

        KeyPair keyPair = getKeyPair(keyStore, keyAlias, passPhrase);
        if (keyPair != null) {
            privateKey = keyPair.getPrivate();
        }
    } catch (KeyStoreException ex) {
    } catch (IOException ex) {
    } catch (NoSuchAlgorithmException ex) {
    } catch (CertificateException ex) {
    }
    return privateKey;
}

/**
 * Get KeyPair from keystore for the given alias
 * @param keyStore
 * @param keyAlias
 * @param passPhrase
 * @return
 */
private KeyPair getKeyPair(KeyStore keyStore, String keyAlias,
        char[] passPhrase) {
```

```
    try {
        // Get private key
        Key key = keyStore.getKey(keyAlias, passPhrase);
        if (key instanceof PrivateKey) {
            // Get certificate of public key
            Certificate certificate = keyStore.getCertificate(keyAlias);

            // Get public key
            PublicKey publicKey = certificate.getPublicKey();

            // Return a key pair
            return new KeyPair (publicKey, (PrivateKey) key);
        }
    } catch (UnrecoverableKeyException e) {
    } catch (NoSuchAlgorithmException e) {
    } catch (KeyStoreException e) {
    } catch (Exception e) {
    }
    return null;
}
```