Openads Web Services API Tutorial

About this document

The Openads Ad Network service connects ad servers with ad networks, providing simplified configuration, distribution and management of online ad campaigns.

Purpose of this document

This document is designed to help developers use the Openads web services API. The tutorial includes:

- PHP example
- Java example
- PHP helper class example
- Java helper class examples

Further information

To use the web services API, it is always best to check Openads website for the latest versions of files and documents:

- XML-RPC files are at **https://developer.openads.org/browser/trunk/lib/xmlrpc**
- Helper files are at **https://developer.openads.org/browser/trunk/OA/Dll**
- API docs are at **http://api.openads.org**

Disclaimers

Copyright 2007 Openads Ltd.

No part of this guide shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without the express permission of Openads. Although every precaution has been taken in the preparation of these materials, no responsibility is assumed for errors or omissions, neither is any liability assumed for damages resulting from the use of the information contained herein.

Introduction

With the Openads web services API, you can manipulate objects in an Openads server that is located on a remote server. objects from a remote computer. manipulate with the basic entities of the system. You can add, modify, delete and return statistics for the following obects:

- Agency
- Advertiser
- Banner
- Campaign
- Publisher
- Zone

The API also includes web services for user authentication and session maintenance. Only users with administrator privileges can use the web services API.

Each web service has its own URL which you must specify to connect to that service. To perform operations on multiple entities, you must log on to each web service URL individually.

For all entities, you must log on to the relevant service to receive an ID for the session and you must include this session ID as the first parameter in all subsequent communications for the session. If you do not log off at the end of a session, the session will eventually timeout, but it is better practice to perform the log off using the logoff method of the log on service.

Many of the operations include routines to create appropriate data strucutres and to pass and session Ids to each method. A library of helper classes are provided which you can use to perform these routine operations.

This tutorial provides PHP and Java usage examples, as well as examples of using the related helper classes:

- PHP example
- Java example
- PHP helper example
- Java helper examples

PHP example

This example demonstrates PHP usage of the Openads web services API to perform simple manipulations of the agency entity. To enable these actions it is necessary to logon to the required web service to receive a session identifier.

Important: You must include the session ID in all messages for the session.

It is good practice to log off from the service when the actions are complete.

The example has the following sections:

- Set up the environment and parameters
- Log on to the XML/RPC service
- Add an agency to the database
- Modify an agency
- Delete an agency
- Log off from the XML/RPC server

1. Set the environment:

```
<?php

if (!@include('XML/RPC.php')) {
     die('Error: cannot load the PEAR XML_RPC class');
}

$xmlRpcHost = 'localhost';
$webXmlRpcDir = '/openads/www/api/v1/xmlrpc';

$logonXmlRpcWebServiceUrl = $webXmlRpcDir . '/LogonXmlRpcService.php';
$agencyXmlRpcWebServiceUrl = $webXmlRpcDir . '/AgencyXmlRpcService.php';

$username = 'admin';
$password = 'admin';
```

2. Define the XML-RPC response:

```
function returnXmlRpcResponseData($oResponse)
{
     if (!$oResponse->faultCode()) {
          $oVal = $oResponse->value();
          $data = XML_RPC_decode($oVal);
          return $data;
     } else {

     die('Fault Code: ' . $oResponse->faultCode() . "\n" .
         'Fault Reason: ' . $oResponse->faultString() . "\n");
     }
}
```

3. Log on and get the session ID:

```
$aParams    = array(
                    new XML_RPC_Value($username, 'string'),
                    new XML_RPC_Value($password, 'string')
              );
$oMessage   = new XML_RPC_Message('logon', $aParams);

$oClient    = new XML_RPC_Client($logonXmlRpcWebServiceUrl, $xmlRpcHost);

$oResponse = $oClient->send($oMessage);

if (!$oResponse) {
     die('Communication error: ' . $oClient->errstr);
}

$sessionId = returnXmlRpcResponseData($oResponse);

echo 'User logged on with session Id : ' . $sessionId . '<br>';
```

4. Add a new agency:

```
$oAgency    = new XML_RPC_Value(
            array(
                 'agencyName' => new XML_RPC_Value('agency name','string'),
                 'contactName' => new XML_RPC_Value
                                  ('contact name', 'string'),
                 ),
                 'struct'
                 );

$aParams     = array(
             new XML_RPC_Value($sessionId, 'string'),
             $oAgency
             );

$oMessage    = new XML_RPC_Message('addAgency', $aParams);

$oClient     = new XML_RPC_Client($agencyXmlRpcWebServiceUrl, $xmlRpcHost);

$oResponse   = $oClient->send($oMessage);

$agencyId    = returnXmlRpcResponseData($oResponse);

echo 'Agency with id: ' . $agencyId . ' added <br>';
```

5. Modify an agency:

```
$oAgency   = new XML_RPC_Value(
             array(
                'agencyId'   => new XML_RPC_Value($agencyId, 'int'),
                'agencyName' => new XML_RPC_Value(
                                    'modified agency name', 'string'),
                'contactName' => new XML_RPC_Value(
                                    'modified contact name', 'string'),
                ),
                'struct'
                );

$aParams   = array(
             new XML_RPC_Value($sessionId, 'string'),
             $oAgency
             );

$oMessage  = new XML_RPC_Message('modifyAgency', $aParams);

$oClient   = new XML_RPC_Client($agencyXmlRpcWebServiceUrl, $xmlRpcHost);

$oResponse = $oClient->send($oMessage);

echo 'Agency with id: ' . $agencyId . ' modified <br>';
```

6. Delete an agency:

```
$oAgency   = new XML_RPC_Value($agencyId, 'int');

$aParams   = array(
             new XML_RPC_Value($sessionId, 'string'),
             $oAgency
             );

$oMessage  = new XML_RPC_Message('deleteAgency', $aParams);

$oClient   = new XML_RPC_Client($agencyXmlRpcWebServiceUrl, $xmlRpcHost);

$oResponse = $oClient->send($oMessage);

echo 'Agency with id: ' . $agencyId . ' deleted <br>';
```

7. Log off from the session:

```
$aParams    = array(
                new XML_RPC_Value($sessionId, 'string')
                );

$oMessage   = new XML_RPC_Message('logoff', $aParams);

$oClient    = new XML_RPC_Client($logonXmlRpcWebServiceUrl, $xmlRpcHost);

$oResponse = $oClient->send($oMessage);

echo 'User with session Id : ' . $sessionId . ' logged off<br>';

?>
```

Java example

This example demonstrates use of Java with Apache libraries to perform simple manipulations of the agency entity. To enable these actions it is necessary to logon to the required web service to receive a session identifier.

Important: You must include the session ID in all messages for the session.

It is good practice to log off from the service when the actions are complete.

The example has the following sections:

- Set up the environment and parameters
- Log on to the XML/RPC service
- Add an agency to the database
- Modify an agency
- Delete an agency

Log off from the XML/RPC server

1. Set the environment:

```
package org.openads.samples;

import java.net.URL;
import java.util.HashMap;
import java.util.Map;

import org.apache.xmlrpc.XmlRpcException;
import org.apache.xmlrpc.client.XmlRpcClient;
import org.apache.xmlrpc.client.XmlRpcClientConfigImpl;

public class AgencySample {
     private final static String serverURL = "http://localhost";
     private final static String openadsDir = "/ openads";
     private final static String logonService =
                          "/www/api/v1/xmlrpc/LogonXmlRpcService.php";
     private final static String agencyService =
                          "/www/api/v1/xmlrpc/AgencyXmlRpcService.php";

     private final static String username = "admin";
     private final static String password = "admin";

     public static void main(String[] args) {
```

2. Create an XML-RPC client:

```
          final XmlRpcClientConfigImpl config = new
                                                XmlRpcClientConfigImpl();
```

3. Log on:

```
try {
     config.setServerURL(new URL(serverURL + openadsDir +
                                          logonService));

     XmlRpcClient client = new XmlRpcClient();
                           client.setConfig(config);

     String sessionId = (String) client.execute("logon",
                          new Object[]{username, password});
     System.out.println("User logged on with session Id:
                                              " + sessionId);
```

4. Set the URL for the agency service:

```
     config.setServerURL(new URL(serverURL + openadsDir
                                         + agencyService));
```

5. Create an agency object

```
     Map<String, Object> struct = new HashMap<String, Object>();
     struct.put("agencyName", "myAgency");
     struct.put("contactName", "Somebody");
     struct.put("emailAddress", "somebody@example.com");
     struct.put("username", "somebody");
     struct.put("password", "somepassword");
     Object[] params = new Object[] { sessionId, struct };
```

6. Add an agency:

```
     final Integer id = (Integer) client.execute
                                    ("addAgency", params);

     System.out.println("Agency with id: " + id + " added");

     if (id != null) {
```

7. Modify an agency:

```
          struct.clear();
          struct.put("agencyId", id);
          struct.put("emailAddress", "somebody@example.com");

          client.execute("modifyAgency", new Object[]
                                  { sessionId, struct });
          System.out.println("Agency with id: " + id +
                                           " modified");
```

8. Delete an agency:

```
          client.execute("deleteAgency", new Object[]
                                       { sessionId, id });
          System.out.println("Agency with id: " + id +
                                            " deleted");
     }
```

9. Log off from the session:

```
                config.setServerURL(new URL(serverURL + openadsDir +
                                                    logonService));

                client.execute("logoff", new Object[]{sessionId});
                System.out.println("User with session Id: " + sessionId +
                                                       " logged off ");
           }
           catch (XmlRpcException e) {
                System.out.println(" Fault Code: " + e.code +
                                   "\n Fault Reason: " + e.getMessage());
           }
           catch (Exception e) {
                e.printStackTrace();
           }
      }
}
```

PHP helper class example

The PHP helper library contains the openads-api-xmlrpc.inc.php file and individual helper files for each entity, such as the AgencyInfo.php file.

1. Set the environment

```
if (!@include('openads-api-xmlrpc.inc.php')) {
     die('Error: cannot load the OpenAds XML_RPC proxy class');
}
require_once 'AgencyInfo.php';

$xmlRpcHost = 'localhost';
$webXmlRpcDir = '/openads/www/api/v1/xmlrpc';

$username = 'admin';
$password = 'admin';
```

2. Log on to the XML-RPC service:

```
$service = new OA_Api_Xmlrpc($xmlRpcHost, $webXmlRpcDir, $username,
$password);

echo 'User logged on with session Id : ' . $service->sessionId . '<br>';

Add a new agency:
$oAgencyInfo = new OA_Dll_AgencyInfo();
$oAgencyInfo->agencyName    = 'my agency';
$oAgencyInfo->contactName   = 'Somebody';
$oAgencyInfo->emailAddress = 'somebody@example.com';

$oAgencyInfo->agencyId = $service->addAgency($oAgencyInfo);
echo 'Agency with id: ' . $oAgencyInfo->agencyId . ' added <br>';
```

3. Modify an agency:

```
$oAgencyInfo->agencyName  = 'modyfied agency name';
$oAgencyInfo->contactName = 'new contact name';
$service->modifyAgency($oAgencyInfo);

echo 'Agency with id: ' . $oAgencyInfo->agencyId . ' modified <br>';
```

4. Delete an agency:

```
$res = $service->deleteAgency($oAgencyInfo->agencyId);
echo 'Agency with id: ' . $oAgencyInfo->agencyId . ' deleted <br>';
```

5. Log off:

```
$res = $service->logoff();
echo 'User logged off <br>';
```

Java helper class examples

Two helper class libraries are available:

- Helper classes for Apache XML-RPC library version 2.0
- Helper classes for with Apache XML-RPC library version 3.0

Java with Apache XML-RPC library version 2.0

This example shows how to use the Java helper classes with the Apache XML-RPC library, version 2.0.

1. Import required classes, set up the session and log on:

```
import java.util.Hashtable;
import java.util.Map;

import org.apache.xmlrpc.XmlRpcException;
import org.openads.proxy.Agency;
import org.openads.proxy.OpenAdsApiXmlRpcProxy;

public class AgencySample {

     public static void main(String[] args) {

          OpenAdsApiXmlRpcProxy proxy = new
                                        OpenAdsApiXmlRpcProxy("localhost",
                                            "/openads/www/api/v1/xmlrpc");

          try {
               proxy.logon("admin", "admin");
               System.out.println("User admin logged in ");
```

2. Add a new agency:

```
               Map<String, Object> struct = new
                                            Hashtable<String, Object>();
               struct.put(Agency.AGENCY_NAME, "myAgency");
               struct.put(Agency.CONTACT_NAME, "Somebody");
               struct.put(Agency.EMAIL_ADDRESS, "somebody@example.com");
               struct.put(Agency.USERNAME, "somebody");
               struct.put(Agency.PASSWORD, "somepassword");

               final Integer agencyId = proxy.addAgency(struct);
               System.out.println("Agency with id: " + agencyId +
                                                    " added");
```

3. Modify an agency:

```
               struct.clear();
               struct.put(Agency.AGENCY_ID, agencyId);
               struct.put(Agency.EMAIL_ADDRESS, "somebody@example.com");

               proxy.modifyAgency(struct);
               System.out.println("Agency with id: " + agencyId +
                                                   " modified");
```

4. Delete an agency:

```
                proxy.deleteAgency(agencyId);
                System.out.println("Agency with id: " + agencyId +
                                                    " deleted");
```

5. Log off

```
                proxy.logoff();
                System.out.println("User admin logged off ");

          } catch (XmlRpcException e) {
          System.out.println(" Fault Code: " + e.code +
                             "\n Fault Reason: " + e.getMessage());
          } catch (Exception e) {
          e.printStackTrace();
          }
     }
}
```

Java with Apache XML-RPC library version 3.0

This example shows how to use the Java helper classes with the Apache XML-RPC library, version 3.0.

1. Import required classes, set up the session and log on:

```
import java.util.HashMap;
import java.util.Map;

Import required XML-RPC classes:
import org.apache.xmlrpc.XmlRpcException;
import org.openads.proxy.Agency;
import org.openads.proxy.OpenAdsApiXmlRpcProxy;

public class AgencySample {

     public static void main(String[] args) {

          OpenAdsApiXmlRpcProxy proxy = new
                                        OpenAdsApiXmlRpcProxy("localhost",
                                        "/openads/www/api/v1/xmlrpc");

          try {
               proxy.logon("admin", "admin");
                    System.out.println("User admin logged in ");
```

2. Add a new agency:

```
               Map<String, Object> struct = new HashMap<String, Object>();
               struct.put(Agency.AGENCY_NAME, "myAgency");
               struct.put(Agency.CONTACT_NAME, "Somebody");
               struct.put(Agency.EMAIL_ADDRESS, "somebody@example.com");
               struct.put(Agency.USERNAME, "somebody");
               struct.put(Agency.PASSWORD, "somepassword");

               final Integer agencyId = proxy.addAgency(struct);
               System.out.println("Agency with id: " + agencyId +
                                                    " added");
```

3. Modify an agency:

```
               struct.clear();
               struct.put(Agency.AGENCY_ID, agencyId);
               struct.put(Agency.EMAIL_ADDRESS, "somebody@example.com");

               proxy.modifyAgency(struct);
               System.out.println("Agency with id: " + agencyId +
                                                  " modified");
```

4. Delete an agency:

```
               proxy.deleteAgency(agencyId);
               System.out.println("Agency with id: " + agencyId +
                    " deleted");
```

5. Log off

```
                proxy.logoff();
                System.out.println("User admin logged off ");

            } catch (XmlRpcException e) {
            System.out.println(" Fault Code: " + e.code +
                               "\n Fault Reason: " + e.getMessage());
            } catch (Exception e) {e.printStackTrace();
            }
        }
}
```