

# Rollbase REST API Examples

Updated At: October 15, 2008

## CONTENTS

<b>REST AND PHP .....</b>	<b>2</b>
<b>Prerequisites.....</b>	<b>2</b>
<b>Custom REST functions in PHP .....</b>	<b>3</b>
<b>Using Rollbase REST API Methods.....</b>	<b>4</b>
Login .....	4
getObjectDef.....	5
search .....	6
getDataObj.....	6
getBinaryData .....	7
getDataField.....	7
getRelationships .....	8
getUpdated .....	8
delete.....	9

Introductory material and documentation on the Rollbase REST API can be found at: <https://www.rollbase.com/support/docs.shtml>

This document contains examples of using the Rollbase REST API in PHP applications. The API can just as easily be used in Java, .NET and other applications as necessary.

## REST and PHP

REST requests are essentially normal HTTP requests that work the same way your web browser makes requests to a web server. Therefore when using the Rollbase REST API in PHP, scripts simply mimic the way a web browser normally behaves. PHP, being a flexible and feature-rich language, provides a number of different mechanisms you can choose from to make your REST requests. Two common approaches are:

- `file_get_contents`
- `curl`

Depending on your web server environment, one or both of these functions may be desirable. `file_get_contents` is the simplest method and is usually preferable for very short scripts. Its main drawback is that it does not let you do HTTP POST calls. Because some Rollbase API methods require POST we recommend that you use `curl`. `curl` is a PHP extension that is widely used for making web services requests, which provides significant control over network transport options and is particularly convenient when making secure HTTPS requests.

Most Rollbase Web Services REST requests mimic HTTP GET requests, but a few use HTTP POST. Therefore, in this guide we provide sample code for both HTTP GET and HTTP POST requests where applicable.

### *Prerequisites*

Both `file_get_contents` and `curl` have configuration prerequisites.

`file_get_contents` is only available in PHP 4.3 and later versions. In general, we recommend that you run PHP version 4.3 or higher so avoid any issues. However, for security reasons your PHP installation must have `fopen wrappers` enabled in order to access URLs using `file_get_contents`. If `fopen wrappers` is not enabled, you will not be able to use `file_get_contents` for web services requests.

`curl` is a PHP extension that must be loaded in your PHP `php.ini` configuration file. Editing your `php.ini` file normally requires administrative privileges on your server so please contact your administrator if necessary.

To find out if `curl` is installed and active in your PHP environment, check whether `fopen wrappers` is enabled, and determine what version of PHP you are running, you can run the `phpinfo` function:

```
<?php
echo phpinfo();
?>
```

Note: To view your current fopen wrappers settings, look for the string `allow_url_fopen` in the output.

## ***Custom REST functions in PHP***

The following is a sample `restCall` method in PHP that can be used to make REST requests:

```
function restCall($request, $method, $pbody="") {

    //Initialize a curl session
    $session = curl_init($request);

    /* You then set any curl options that you might need. curl
    options are listed at
    http://us2.php.net/manual/en/function.curl-setopt.php. In this
    case, we tell curl that we don't want the HTTP headers returned,
    but we do want the request data returned: */

    // Tell curl not to return headers, but do return the response
    curl_setopt($session, CURLOPT_HEADER, false);

    curl_setopt($session, CURLOPT_RETURNTRANSFER, true);

    if($method == 'POST') { //For post method notify curl

        // Tell curl to use HTTP POST
        curl_setopt ($session, CURLOPT_POST, true);

        // Tell curl that this is the body of the POST
        curl_setopt ($session, CURLOPT_POSTFIELDS, $pbody);
    }

    //send request
    $response = curl_exec($session);

    //close session
    curl_close($session);

    //this is the response(xml)
    return $response;
}
```

Below is a sample `checkStatus` method in PHP that can be used to check if the status of the response is "ok":

```
function checkStatus($response) {
    if(strlen($response) < 0) {
        echo "No Response set";
        return;
    }
    $status = getNodeAsArray($response, "//@status");
    if(count($status) > 1) {
        echo "more than one status found";
        return false;
    }
}
```

```
if($status[0] == "ok")
    return true;

if($status[0] == "fail") {
    $error = getNodeAsArray($response, "//err");
    echo "Error in Response: $error[0]";
    return false;
}
}
```

Here the `getNodeAsArray()` custom method contains XML processing code that parses the XML response, finds the required node and returns the values in the nodes as an array for display purposes. To process XML in PHP one can use the DOM XML extension and XPATH extension. There are many other ways to parse XML in PHP. A discussion of XML parsing is beyond the scope of this guide. For more information see: <http://www.ibm.com/developerworks/library/x-xmlphp1.html>

## Using Rollbase REST API Methods

### Login

Login is the first REST call that should be made to Rollbase in order to establish an API session and retrieve a session ID, which is required as a parameter for all subsequent calls.

For example to login to a Rollbase account with username `TestUser` and password `mypass` a PHP 5 REST call will look like:

```
$uname = 'TestUser';
$password = 'mypass';
$base = ;

$request =
'http://www.rollbase.com/rest/api/login?loginName=' . $uname . '&password='
.$password;
$response= restCall($request, "GET");
```

Here we are using the above function `restCall` to make the request. The XML response can now be processed to obtain the session id.

Response with success:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
    <sessionId>rest-433533393144236162</sessionId>
</resp>
```

Response with error:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="fail">
    <err>Login session is not found</err>
</resp>
```

We can use our custom `checkStatus` method to check if the response contains the information we need:

```
if(checkStatus($response))
    $sessionId = getNodeAsArr($response, "//sessionId");
```

If the status is "ok", we then retrieve the `sessionId` from the XML and store it for making further REST calls.

Now that we have established a REST session with Rollbase, we can send and retrieve data. Let us now look at using other Rollbase REST methods in more detail.

## getObjectDef

This method retrieves the full object definition given the integration name of the target object. The result is an XML file, which can be parsed as necessary to find information required to access various object resources. (The schema definition describing the object definition XML format is available at: <http://www.rollbase.com/webapi/wsd/objDefDescription.xsd>)

```
$request =
'http://www.rollbase.com/rest/api/getObjectDef?sessionId= '.$sessionId.'
&objName=person';
$response= restCall($request, "GET");
if(checkStatus($response)) {
    //process response xml
}
```

Here `person` is the integration name for an object called `Person`. The following is an example of an object definition in XML format of an object definition:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
  <DataObjectDef objDefName="person" >
    <SingularName>Person</SingularName>
    <PluralName>Persons</PluralName>
    <Description>Person</Description>
    <DataFieldDefs>
      <DataFieldDef fieldName="mobilePhone" groupName="CONTACT"
        dataName="FieldString" isRequired="no" isReadOnly="no"
        maxLength="50" >
        <DisplayLabel>Mobile Phone</DisplayLabel>
        <Description>Mobile Phone</Description>
      </DataFieldDef>
    </DataFieldDefs>
    <RelationshipDefs>
      <RelationshipDef relName="person_comm" objDef1="person"
        objDef2="COMMLOG" />
    </RelationshipDefs>
  </DataObjectDef>
</resp>
```

This XML file can be parsed and information about the object such as the singular and plural name, field integration names, display names, etc can be easily retrieved. For example a field definition in this XML file is represented as:

```
<DataFieldDef fieldName="mobilePhone" groupName="CONTACT"
dataName="FieldString" isRequired="yes" isReadOnly="no"
maxLength="50" >
<DisplayLabel>Mobile Phone</DisplayLabel>
<Description>Mobile Phone</Description>
</DataFieldDef>
```

...where the `fieldName` attribute represents the integration name of this field, and the `<DisplayLabel>` and `<Description>` elements contain display information about this field.

## search

To retrieve IDs of all records that are accessible to the current user in a particular object or in all objects, we can use the `search` method. This method performs a full-text search throughout your Rollbase account:

```
$request =
'http://www.rollbase.com/rest/api/search?sessionId=.'.$sessionId.'&query
=john&objDefName=person;
$response = restCall($request, "GET");
if(checkStatus($response)) {
    //process the response xml
}
```

In this example a Rollbase account is searched for the string "john" in an object with integration name 'person'. The `$response` variable will now contain XML with all of the record IDs that match this request:

```
<resp status="ok">
  <ids>
    <id>314452</id>
    <id>128003</id>
  </ids>
</resp>
```

These IDs can be retrieved and stored as PHP variables for further processing.

## getDataObj

This method is used to retrieve individual records for the given record ID. We can perform this REST call as follows for our example person object:

```
$rid = 314452;

$request =
'http://www.rollbase.com/rest/api/getDataObj?sessionId=.'.$sessionId.'&i
d='.$rid;
$response = restCall($request, "GET");
if(checkStatus($response)) {
    //process response xml.
}
```

Here `$rid` represents the ID of the record that we want to retrieve. The resulting `$record` that we get here has the following structure:

- ID of the record retrieved
- Object definition name of the current object
- Array of fields

The XML response looks like:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
  <data id="131227" objName="person">
    <field name="id">131227</field>
    <field name="club_member">false</field>
    <field name="lastName">Smith</field>
    <field name="status">Created</field>
    <field name="updatedAt">20080111T143331Z</field>
  </data>
</resp>
```

## getBinaryData

This method is used to retrieve binary data (i.e. a file attachment) associated with a particular file field from a specific record.

```
$request =
'http://www.rollbase.com/rest/api/getBinaryData?sessionId= '.$sessionId.
'&id= '.$recId.'&fieldName= '.$fieldName;
echo "click the link below to download your file<br>";
echo "<a href='$request' target='_blank'> Open File </a>";
```

Here `$recId` is the ID of the target record and `$fieldName` is the integration name of that file field. When the above link is clicked the user will be able to view/download the file attachment. There is no XML output for this call.

## getDataField

This method is used to retrieve the value of a single field from a specific record.

```
$request =
'http://www.rollbase.com/rest/api/getDataField?sessionId= '.$sessionId.'
&id= '.$recId.'&fieldName= '.$fieldName;
$response = restCall($request, "GET");
if(checkStatus($response)) {
    //process response xml
}
```

Here `$recId` is the ID of the target record and `$fieldName` is the integration name of that field. The resulting XML looks like:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
  <field name="myemail">sam_tr@hotmail.com</field>
```

```
</resp>
```

## getRelationships

This method is used to retrieve all records that are related to a particular record given a particular relationship. For example, assume we want to retrieve all related records associated with a record with ID `$recId` in a relationship that has the integration name `$relName`:

```
$request =  
'http://www.rollbase.com/rest/api/getRelationships?sessionId='.$sessionId.'  
&id='.$recId.'&relName='.$relName;  
$response = restCall($request, "GET");  
if(checkStatus($response)) {  
    //process response xml  
}
```

Here `$recId` is the ID of the target record and `$relName` is the integration name of the relationship. The resulting XML looks like:

```
<resp status="ok">  
  <ids>  
    <id>314452</id>  
    <id>128003</id>  
  </ids>  
</resp>
```

## getUpdated

This method is used to retrieve an array of record IDs of a specific object type, which have either been created or updated within the given date/time interval.

```
$request =  
'http://www.rollbase.com/rest/api/getUpdated?sessionId='.$sessionId.'&f  
rom='.$dateFrom.'&till='.$dateTo.'&objName='.$objname;  
$response = restCall($request, "GET");  
if(checkStatus($response)) {  
    //process response xml  
}
```

The dates that are passed to this method are UTC times in 20080405T144800Z format. In PHP this can be achieved as follows:

```
dateTo = gmdate("Y-m-d", $inpDate)."T".gmdate("H:i:s", $inpDate)."Z";
```

The response XML looks like:

```
<resp status="ok">  
  <ids>  
    <id>314452</id>  
    <id>128003</id>  
  </ids>  
</resp>
```

## delete

This method is used to move a specified record to the Recycle Bin. The parameters required are session ID and record ID:

```
$request =  
'http://www.rollbase.com/rest/api/delete?sessionId='.$sessionID.'&id='.  
$rid;  
$response = restCall($request, "GET");  
if(checkStatus($response)) {  
    echo "successfully logged out";  
}
```

Here `$rid` is the ID of the record that is to be moved to the Recycle Bin.