Python

The Python SDK, not part of our core offering, provides a Python interface for the Gigya API. The library makes it simple to integrate Gigya services in your Python application. This document is a practical step-by-step guide for programmers who wish to integrate the Gigya service into their Python application. Follow the steps below to get started, and use the Library Reference while implementing.

Note: Gigya supports Python 2.7.x and 3.

Library Guide

Please follow these steps to integrate this library in your Python application:

1. Download the zip file from: Gigya Developer Downloads

   The file includes the GSSDK.py and the cacert.pem files. Extract both files to the same location.
2. Obtain Gigya's APIKey and Secret key.
3. Import GSSDK.py library into your application.
4. Log the user in.
5. Use Gigya's API - Send Requests.
6. Optional: incorporate security measures.

Obtaining Gigya's APIKey and Secret key

Making API calls requires an **API Key** and a **Secret Key** which are obtained from the Site Dashboard page on the Gigya website. The **Secret Key** must be kept secret and never transmitted to an untrusted client or over insecure networks. The **API Key** and the **Secret Key** are required parameter in each request (further ahead in this document you will find guidance for sending requests).

Importing the GSSDK.py Library into Your Application

To get started, you'll need to import Gigya Python SDK to your application:

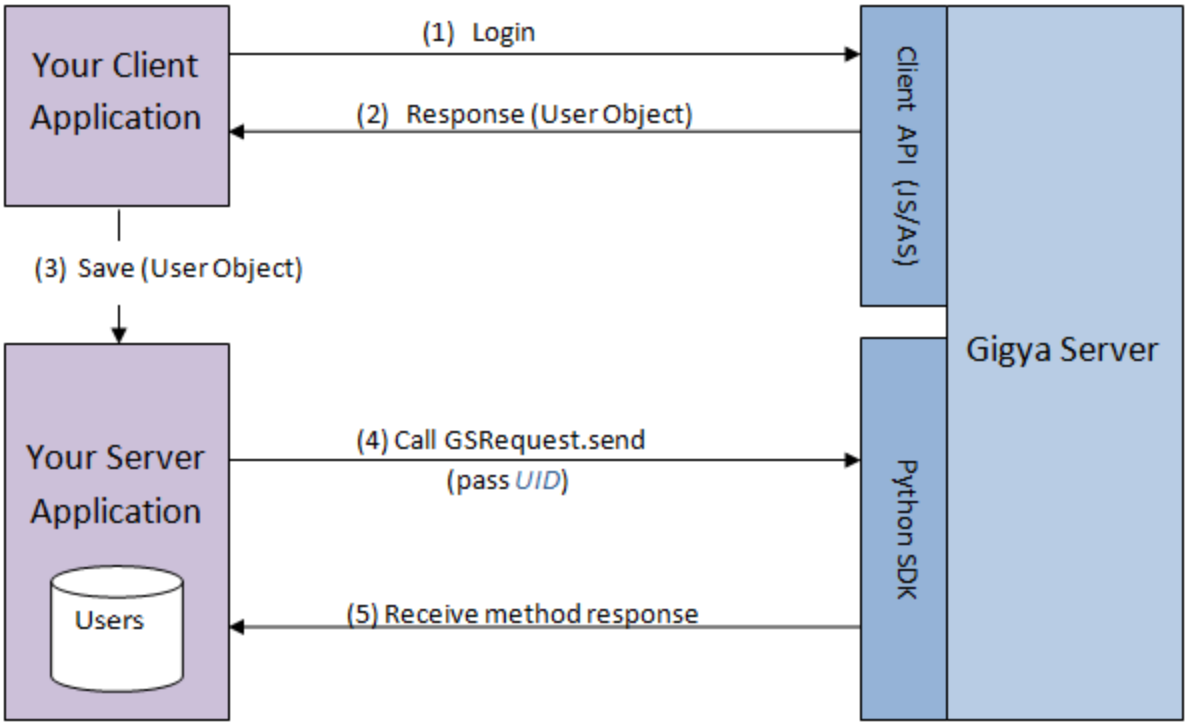
1. Copy the GSSDK.py file to your Python application path.
2. Import the GSSDK.py file into your application:

```
from GSSDK import *
```

You should now be able to use the SDK in your project.

Logging the User in

The first interaction with Gigya must always be logging in. If the user is not logged in, you cannot access their social profile nor perform social activities, such as setting their status. Sending requests requires an identified Gigya user (the identification of whom is performed using the **UID** p arameter) with an **active session**. A user session is created when a user logs in via the Gigya service. Log users in through your client application using our Web SDK methods: socialize.login, socialize.notifyLogin, or using our ready made Social Login UI.



To learn more about the login process, see Social Login.

Sending a Request

After you have logged in the user, you may use the GSRequest class to access the user profile and perform various activities. This is implemented using GSRequest's send method. The following code sends a request to set the current user's status to "I feel great":

```
# Define the API-Key and Secret key (the keys can be obtained from your
site setup page on Gigya's website).
apiKey = "PUT-YOUR-APIKEY-HERE"
secretKey = "PUT-YOUR-SECRET-KEY-HERE"

# Step 1 - Defining the request and adding parameters
method = "socialize.setStatus"
params={"uid":"PUT-UID-HERE", "status":"I feel great"}    # Set "uid" to
the user's ID, and "status" to "I feel great"
request = GSRequest(apiKey,secretKey,method,params)

# Step 2 - Sending the request
response = request.send()

# Step 3 - handling the request's response.
if (response.getErrorCode()==0):
    # SUCCESS! response status = OK
    print "Success in setStatus operation."
else:
    # Error
    print "Got error on setStatus: " + response.getErrorMessage()
    # You may also log the response: response.getLog()
```

Step 1: Defining the Request and Adding Parameters

Create a GSRequest instance:

```
method = "socialize.setStatus"
params={"uid":"PUT-UID-HERE", "status":"I feel great"}    # Set "uid" to
the user's ID, and "status" to "I feel great"
request = GSRequest(apiKey,secretKey,method,params)
```

The parameters of the GSRequest are:

1. apiKey
2. secretKey

> **Note:** For information about obtaining these keys, see above.

3. method - the Gigya API method to call, including namespace. For example: 'socialize.getUserInfo'. Please refer to the REST API reference for the list of available methods.
4. params - In this case the uid and status.

> **Note:** In the REST API reference you may find the list of available Gigya API methods and the list of parameters per each method.

Step 2: Sending the Request

Execute GSRequest's send method:

```
response = request.send()
```

The method returns a GSResponse object, which is handled in the next step.

Note: By default, requests to Gigya APIs are sent using the "us1.gigya.com" domain. If your site has been set up to use another of Gigya's data centers, you must specify that the request should be sent to that specific data center by adding the following line of code before calling the Send method:

```
request.setAPIDomain("<Data_Center>")
```

Where **<Data_Center>** is:

- **us1.gigya.com** - For the US data center.
- **eu1.gigya.com** - For the European data center.
- **au1.gigya.com** - For the Australian data center.
- **ru1.gigya.com** - For the Russian data center.
- **cn1.gigya-api.cn** - For the Chinese data center.

If you are not sure of your site's data center, see Finding Your Data Center.

See the GSRequest documentation for more information.

Step 3: Handling the Response

Use the GSResponse object to check the status of the response, and to receive response data:

```
if (response.getErrorCode()==0):
    # SUCCESS! response status = OK
    print "Success in setStatus operation."
else:
    # Error
    print "Got error on setStatus: " + response.getErrorMessage()
    # You may also log the response: response.getLog()
```

The GSResponse object includes data fields. For each request method, the response data fields are different. Please refer to the Gigya REST API reference for the list of response data fields per method.
For example - handling a socialize.getUserInfo response:
The response of 'socialize.getUserInfo' includes a 'user' object.

```
# Sending 'socialize.getUserInfo' request
params = {"uid", "PUT-UID-HERE"}  // set the "uid" parameter to user's ID
request = GSRequest(apiKey,secretKey,"socialize.getUserInfo",params)
response = request.send()

# Handle 'getUserInfo' response
if (response.getErrorCode() == 0):
    # SUCCESS! response status = OK
    nickname = response.getObject("nickname")
    age = response.getObject("age")
    print "User name: " + nickname + " The user's age: " + age
else:
    print "Got error on getUserInfo: " + response.getErrorMessage()
    # You may also log the response: response.getLog()
```

Optional - Incorporating Security Measures

Validating Signatures

> Signature validation is only necessary and supported when validating the signature of a response that was received on the client side and then passed to the server. Server-to-server calls do not contain the **UIDSignature** or **signatureTimestamp** properties in the response.

The Gigya service supports a mechanism to verify the authenticity of the Gigya processes, to prevent fraud. When Gigya sends you information about a user, your server needs to know that it is actually coming from Gigya. For that cause, Gigya attaches a cryptographic signature to the responses that include user information. We highly recommend validating the signature. The SigUtils class is a utility class for generating and validating signatures.

For example, Gigya signs the socialize.getUserInfo method response. The following code validates the signature received with the 'socialize.getUserInfo' method response:

```
# Handle 'socialize.getUserInfo' response
if (response.getErrorCode()==0):
    # SUCCESS! response status = OK
    # Validate the signature:
    valid =
SigUtils.validateUserSignature(response.getObject("UID"),response.getObjec
t("signatureTimestamp"),
        secretKey,response.getObject("UIDSignature"))
    if (valid):
        print "signature is valid"
    else:
  print "signature is not valid"
```

The parameters of the validateUserSignature method are:

1. UID - the user's unique ID
2. signatureTimestamp - The GMT time of the response in UNIX time format (i.e. the number of seconds since Jan. 1st 1970). The method validates that the timestamp is within five minutes of the current time on your server.
3. secretKey - The key to verification is your partner's "**Secret Key**". Your secret key (provided in BASE64 encoding) is located at the bottom of the Dashboard section on Gigya's website (Read more above).

4. UIDSignature - the cryptographic signature.

All the parameters, with the exception of the *secretKey*, should be taken from the 'User' object received with the 'getUserInfo' method response. The method returns a Boolean value, signifying if the signature is valid or not.

In a similar fashion, when using the 'getFriendsInfo' method, The method response include a collection of 'Friend' objects. Each Friend object will be signed with a cryptographic signature. To verify the signature of a friend object, please use the validateFriendSignature method.

Sending Requests over HTTPS

If you would like to use Gigya service over HTTPS, you will need to do the following:

Create a GSRequest object using the constructor that receives five parameters. The additional fifth parameter is a Boolean parameter named *use HTTPS*. Set this parameter to be ***true***.

Appendix - Publish User Action Example

The following code sample sends a request to publish a user action to the newsfeed stream on all the connected providers which support this feature.

The socialize.publishUserAction method has a complex parameter called userAction which defines the user action data to be published. To define the userAction parameter create a JSON object and fill it with data. Fill the object with data as shown in the example below:

```
# Publish User Action

# Defining the userAction parameter
userAction = {"title":"","userMessage":"This is my user
message","description":"This is my description",

"linkBack":"http://google.com",{\"src\":\"http://www.f2h.co.il/logo.jpg\",
\"href\":\"http://www.f2h.co.il\",\"type\":\"image\"}}

# Sending 'socialize.publishUserAction' request
params = {"userAction":userAction, "uid":"PUT-UID-HERE"}
request = GSRequest("PUT-YOUR-APIKEY-HERE", "PUT-YOUR-SECRET-KEY-HERE",
"socialize.publishUserAction", params)
response = request.send()
```

To learn more about publishing user actions, please read the Advanced Sharing guide.