# Happiness Meter for Mobile Apps

## *Mobile Web Widget Integration and Development Guide*

*Version 3.0*

## Table of Contents

Document Control ..... 3
Document History ..... 3
Distribution List ..... 3
Approval List ..... 3
1 Introduction ..... 4
1.1 About This Guide ..... 4
1.2 Purpose ..... 4
1.3 Prerequisites ..... 4
2 Sample Application ..... 5
2.1 How it works? ..... 5
2.2 Sample Application Building Blocks for ANDROID environment: ..... 10
Note: This section is to be referred to in conjunction with the 'Sample App' source code provided. ..... 10
2.3 Sample Application "iOS" ..... 12
Note: This section is to be referred to in conjunction with the 'Sample App' source code provided. ..... 12
4. Appendix A – Request Parameters ..... 16
5. Appendix B – Sample Request Parameters ..... 19
6. Appendix C - Security Requirements ..... 20

## Document Control

### Document History

| Date | Version | Author(s) | Description |
|---|---|---|---|
| 28/03/2016 | 1.0 | Tabish Shamim (DSG) | Created |
| 07/04/2016 | 2.0 | Tabish Shamim (DSG) | Added snapshots and iOS and Android specific steps |
| 24/01/2017 | 3.0 | Faizan Ali | Updated with Happiness Meter V2 changes |
| | | | |

### Distribution List

| Name | Title | Entity |
|---|---|---|
| Jameel Wasfat Ahmed | Smart Solutions Development Section Manager | |
| Hend Saqer Al Nuaimi | Senior eServices Provisioning Officer | |
| Ahmad Ali Shobaki | Principal Systems Developer | |

### Approval List

| Date | Name | Title | Signature |
|---|---|---|---|
| | | | |
| | | | |

# 1 Introduction

## 1.1 About This Guide

This document describes and explains the necessary development activities required by Service Providers (Government Departments) to integrate with the new “Happiness Meter Systems” by adding the Happiness Mobile Web Widget into their native apps (iOS or Android).

The audience of this guide are Developers, Software Specialist and Software Application vendors who are assigned to integrate Service Providers’ existing mobile apps with the Happiness Meter System using the Mobile Web Widget developed by Dubai Smart Government.

## 1.2 Purpose

This guide helps the implementer to understand the required development actions and steps required to integrate with the Happiness Meter System with the department’s mobile apps (iOS or Android).
Any number of mobile apps can be integrated with the Dubai Smart Government provided solution, provided the steps in the following sections are adhered to.

## 1.3 Prerequisites

Before proceeding with this Development Guide you should have a good knowledge of the target platform (i.e. either Android or iOS development platform).

You should be familiar with mobile apps development on the said platforms with basic understating to the HTTP "Hyper Text Transfer Protocol" and Servlets invoking.

# 2 Sample Application

## 2.1 How it works?

This section describes the workflow and how the sample application is working.

**Note:** To facilitate rapid integration of the mobile app widget developed by DSG into the department mobile apps, DSG has developed a demo app to show how the integration would be done, and how the screens would look like when deployed. **This section of the document mainly describes the functionality from the provided 'sample app' perspective, but the concepts and functionality should be the same when integrated with the target mobile app of the department.**

There are 3 scenarios in which the user can submit the votes from their mobile apps:

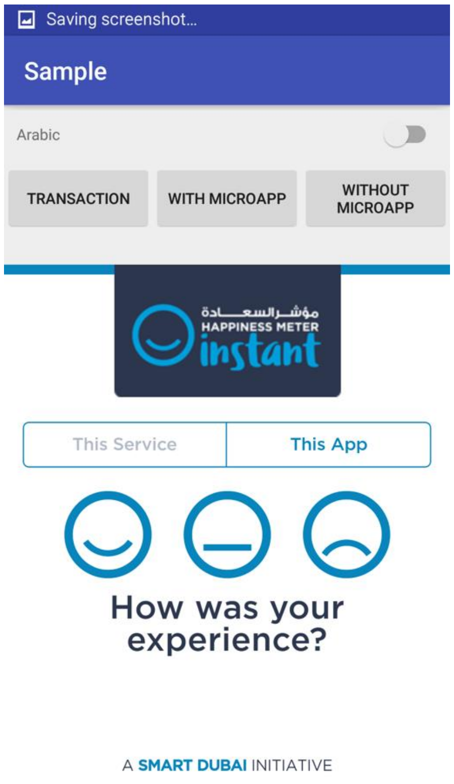
1. **Application Level Votes:** Vote submitted for the overall application ('This App' as in the snapshots).
2. **Application Level but 'microApp' votes:** Vote submitted for the microApp (or specific section of the App as desired by the Service Provider).
3. **Transaction votes:** Vote submitted for a transaction

Out of the above 3 kinds of votes, '1' and '3' have been available in the Web based votes which are already implemented by most of the departments. The only new kind of vote possible from the mobile apps would be the 2rd one i.e. enabling to submit vote for a specific section / microApp of the department.

The Sample app just contains 3 buttons to mimic submission of each of the above 3 kinds of votes. Please refer further explanations below:

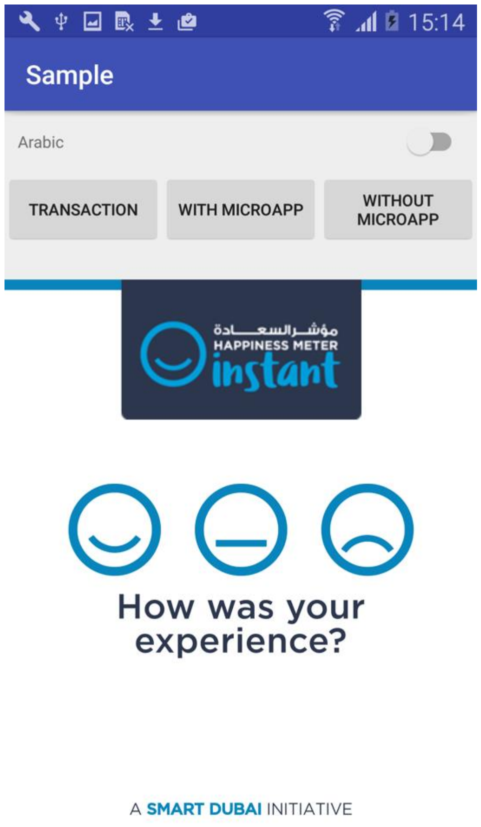
1. **Invocation of the Happiness Mobile Web Widget:** Once the end user clicks on the "Happiness Gadget Button" (i.e. _this button must be added to all pages in the department app where feedback is desired_), a modal dialogue window will appear under which the survey/happiness form will be opened.

2. **Vote submitted for the application / microApp:** (User selects on 'This App' or the microApp name. User selection gets highlighted in 'blue' color – the unselected option is 'grey' in color):

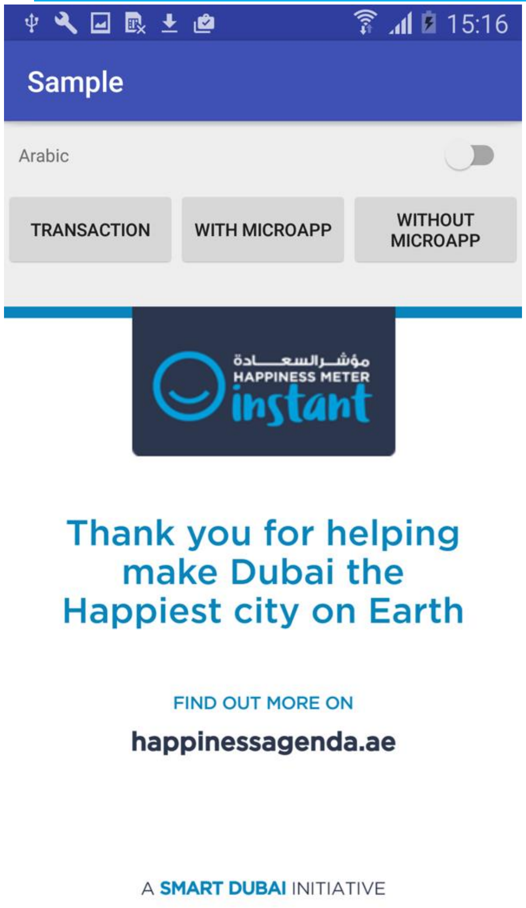


3. **Vote submitted for a transaction:** (user will not have option to select 'This App' or microApp. This is by design, since at this point user is providing his feedback for the transaction just completed, and not the app or microApp).

   **Note:** The happiness widget would automatically popup at the completion of a transaction, unlike the 'Application' level votes, where the user has to click on the Happiness icon to bring the popup.



4. **Thanks page:** Once the user select the desired face icon based on his experience, following screen will show a thanks message.

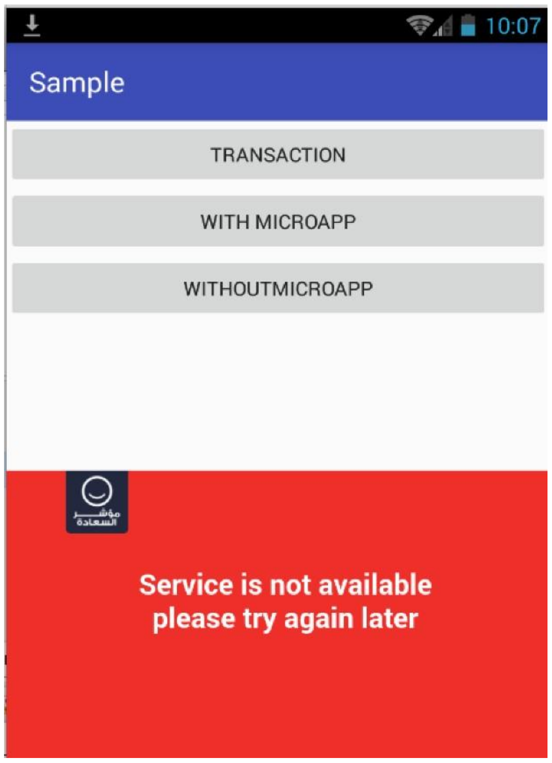


Thank you for helping make Dubai the Happiest city on Earth

FIND OUT MORE ON
happinessagenda.ae

A SMART DUBAI INITIATIVE

5. **Error page:** If, for some reason, there is an error on the server side (e.g. because of wrong timestamp or validation failed), an error page would be displayed:



Service is not available
please try again later

**Notes:**

1. Please note that user can navigate out of the survey dialog by clicking any place outside the shown dialog.
2. Also, once the 'thanks' message is displayed (after vote submission), the survey dialog popup would close by itself after a preconfigured time (currently 5 seconds) on the server side.
3. For the 'Thanks' page above, departments would need to provide their transparent images in the specified size so that the same can be configured on DSG side):

   **Image size: 200 X 64 pixels (background should be transparent)**

4. We suggest that the 'Happiness gadget' button (that the user clicks to open Happiness popup) should always be placed at a permanent position (preferably at the bottom). (For a reference implementation, you can refer to the 'Dubai Now' app which is available from Google Playstore and Apple AppStore)

5. **What happens when the user submits a 'microApp' vote?**

   Normal application level votes (i.e. when the user clicks on 'This App') are saved in the database based on the JSON received – refer to an example of JSON in subsequent section. In this case, the 'application_id' is saved as the application_id received from the service provider. However, in case the user submits a vote for the microApp, DSG contatenates the value of the microApp with the application_id (separated by colon ‘:’), and then saves the resulting application_id in the database. E.g.:

**‘This App’ Vote submitted:**

- Application_id : ‘RTAApplicationXYZ’
- Value of ‘microApp’ parameter:  ‘’” (empty)
- Value of ‘microAppDisplay’ parameter: “” (empty)
- Application_id saved for the votes:  ‘RTAApplicationXYZ’

**‘microApp’ Vote submitted:**

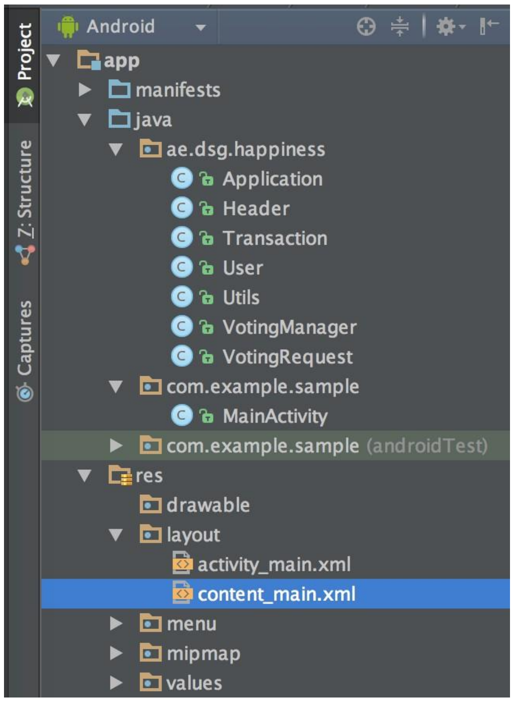
- Application_id : ‘RTAApplicationXYZ’
- Value of ‘microApp’ parameter:  ‘NOLPayment’
- Value of ‘microAppDisplay’ parameter: “Pay NOL”
- Application_id saved for the votes:  ‘RTAApplicationXYZ’: NOLPayment’

This would enable the departments to track later (through the inquiry service) which votes have been submitted for the App itself, and which ones for the specific microApps.

## 2.2 Sample Application Building Blocks for ANDROID environment:

**Note:** This section is to be referred to in conjunction with the 'Sample App' source code provided.

The Sample Application is an Android Application that contains the following structure, we are using android studio to build this sample application and this is how the project is shown the Project Explorer view in Android Studio.



```
app
  manifests
  java
    ae.dsg.happiness
      Application
      Header
      Transaction
      User
      Utils
      VotingManager
      VotingRequest
    com.example.sample
      MainActivity
    com.example.sample (androidTest)
  res
    drawable
    layout
      activity_main.xml
      content_main.xml
    menu
    mipmap
    values
```

Following are the main building blocks:

### 2.2.1 Main Activity

The Main Activity is the starting activity of the application; this is the activity where the end web view is added in order to access Happiness Meter URL.

In order to add this step to your application:

- Copy the 'ae.dsg.happiness' package to your source code.
- Add a 'web view' to your view where you want to show Happiness Meter.
- Copy the 'load' method from the Main Activity and access the web view from your view in the first line.

- Set the rest of the parameters accordingly and request Voting Manager to load Happiness Meter URL.

- Call the load method with appropriate type of request to load.

- The minimum height of the web view is should be 430dp in order to display the Happiness Meter correctly.

- Change the HAPPINESS_URL in the VotingManager.java class with either the production URL or the QA URL depending on which environment you are connecting to.

## 2.3 Sample Application "iOS"

**Note:** This section is to be referred to in conjunction with the 'Sample App' source code provided.

iOS sample app is build using Xcode version 7.3. Just click on **xcproject** file to open the project.

### 2.3.1 View Controller

View Controller is the starting screen of the application. This is the view controller where the web view is added in order to access Happiness Meter URL.

In order to add this step to your application:

- Copy the 'DSGHappiness' folder to your source code. Copy all imports as done in viewController.
- Add a 'UIWebView' to your view where you want to show Happiness Meter. Bind the web view delegate to viewController and assign web view as property in viewController.
- Copy the 'logRequestWithVotingManager' function from the ViewController and access the web view from your view in the first line.
- Set the rest of the parameters accordingly as done in "didLoad", "actionLog" and "logrequestWithVotingManager" function in viewController and request Voting Manager to load Happiness Meter URL.
- Note that request_type is being set according to the button tag numbers in actionLog function. Developer can set the request_type explicitly as its defined as enum.
- Call the logRequestWithVotingManager method with appropriate type of request to load. All the parameters and their types are defined in the document and in code its decided according to the request_type enum.
- The minimum height of the web view should be 260 pixels in order to display the Happiness Meter correctly.
- Change the requestUrl in the VotingManager.m class initWithParams function with either the production URL or the QA URL depending on which environment you are connecting to.
- Kindly note that comments are provided with class definitions and the implementation of code. Kindly follow all the comments and instructions to ensure the desired code

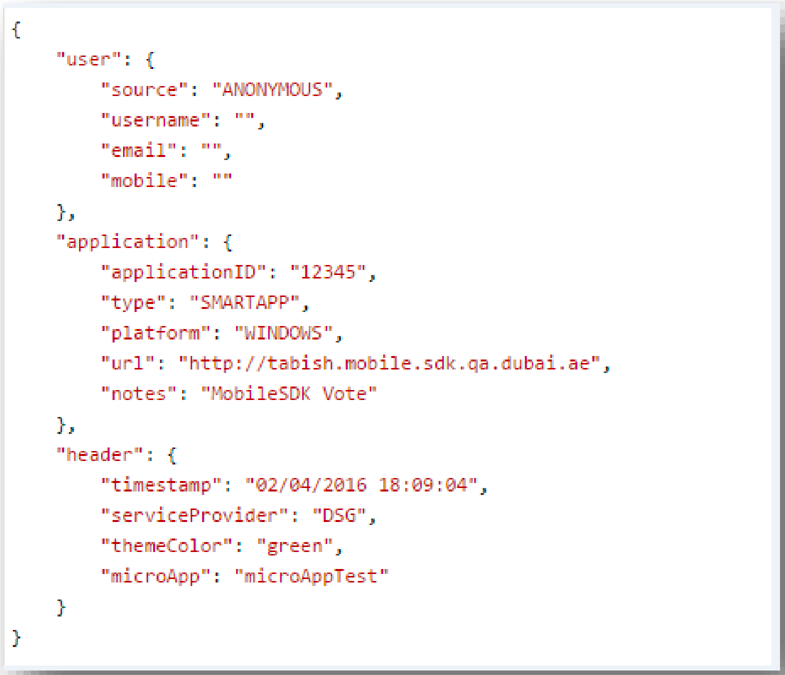
execution results.

Note: Recommended minimum height of the web view is 400px.

## Additional notes about the implementation:

1- By default happiness feedback will be at application level if you want to do transaction based feedback, you have to create the JSON request to have a Transaction object rather than an application object. Examples of the JSON submitted from the department side for Application and Transaction votes are as following:

**Application Vote Example:**

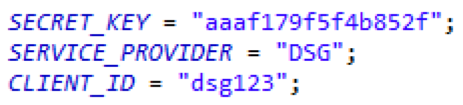


```
{
    "user": {
        "source": "ANONYMOUS",
        "username": "",
        "email": "",
        "mobile": ""
    },
    "application": {
        "applicationID": "12345",
        "type": "SMARTAPP",
        "platform": "WINDOWS",
        "url": "http://tabish.mobile.sdk.qa.dubai.ae",
        "notes": "MobileSDK Vote"
    },
    "header": {
        "timestamp": "02/04/2016 18:09:04",
        "serviceProvider": "DSG",
        "themeColor": "green",
        "microApp": "microAppTest"
    }
}
```

**Transaction Vote Example:**

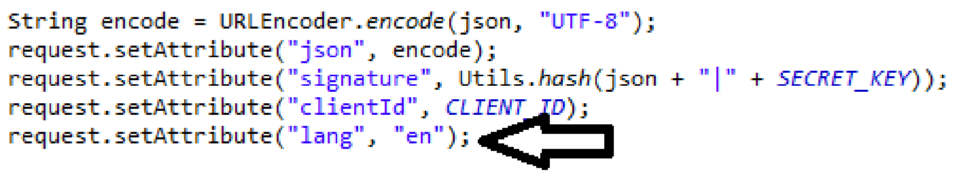
```
{
  "user":{
    "source":"ANONYMOUS"
  },
  "header":{
    "timestamp":"24\/01\/2017 09:41:03",
    "serviceProvider":"DSG",
    "themeColor":"#029cdf"
  },
  "transaction":{
    "transactionID":"Happiness Vote 1485250863375",
    "gessEnabled":"true",
    "serviceCode":"2952",
    "serviceDescription":"Demo Transaction",
    "channel":"SMARTAPP",
    "notes":"MobileSDK Vote"
  }
}
```

2- The following values would need to be changed in your code, based on the key information received from DSG:



```
SECRET_KEY = "aaaf179f5f4b852f";
SERVICE_PROVIDER = "DSG";
CLIENT_ID = "dsg123";
```

**3-** You can change the text language by setting either **ar** or **en:**



```
String encode = URLEncoder.encode(json, "UTF-8");
request.setAttribute("json", encode);
request.setAttribute("signature", Utils.hash(json + "|" + SECRET_KEY));
request.setAttribute("clientId", CLIENT_ID);
request.setAttribute("lang", "en");
```

4- The following request parameters need to be populated for sending to the server (DSG) side servlet:

| Parameter Name | Description |
|---|---|

| | |
|---|---|
| json_payload | Request payload in JSON (i.e. this value must be URL encoded) |
| client_id | Service provider code (will be provided by DSG) |
| signature | Calculated signature for the request |
| lang | This will be the survey language accepted values are (**en** \| **ar**) |
| random | 16 digit random number (i.e. see below sample code)<br><br>(I.e. please refer to Appendix C - Security Requirements) |
| timestamp | System Timestamp (DD/MM/YYYY HH24:MI:SS) always in UTC time zone<br><br>(i.e. please refer to Appendix C - Security Requirements) |
| nonce | Secure value calculated for each request (i.e. please refer to Appendix C - Security Requirements) |
| themeColor | If the department wishes to override the themeColor for the popup widget, they can send the color in #hashCode format. |
| microApp | When the vote is submitted for a microApp rather than for the main application, this parameter needs to be sent.<br><br>**Note:** microApp is concatenated to application_id before saving in the database. Departments should always provide English value of the 'microApp'. |

| | |
|---|---|
| microAppDisplay | This is the parameter for displaying the microApp name on the Happiness popup tab. It may be the same as 'microApp' name or different.<br><br>**Note:** 'microAppDisplay' can be in English or Arabic. E.g. while rendering the Arabic Happiness, you should send 'microAppDisplay' as the Arabic name, whereas 'microApp' should be the English name. |

# 4. Appendix A – Request Parameters

The following table provides a description of all parameters.

| Key Name | Value |
|---|---|
| serviceProvider | DSG will provide the value |
| client_id | DSG will provide the value |
| PostURL Production | https://happinessmeter.dubai.gov.ae/HappinessMeter2/MobilePostDataService |
| PostURL QA | https://happinessmeterqa.dubai.gov.ae/HappinessMeter2/MobilePostDataService |
| SecretKey | DSG will provide the value |
| serviceCode | DSG will provide the value |
| channel | SMARTAPP |
| source | User Source Any of<br>LOCAL \| MYID \| ANONYMOUS<br>Where LOCAL to be used with Departments Local User Profile. |
| serviceDescription | Provide the detail of your service |
| applicationID | Provide the name of your application |
| gessEnabled | Indicates if the department service is exist on GESS system or not<br><br>Allowed Values {true, false} |
| Applicationtype | Provide the type of your application as:<br><br>**SMARTAPP** |
| platform | Any of IOS\|ANDROID\|BLACKBERRY\|WINDOWS \|OTHERS<br><br>*Filed Required when type is SMARTAPP |
| Applicationurl | Web application URL |
| version | Application version |
| mircoApp | Name of the microApp for which the vote is submitted.<br><br>If the vote is submitted for the overall application (i.e. the user clicks on 'This App', then 'microApp' value would be empty string). |

| | |
|---|---|
| mircoAppDisplay | Display name of the microApp for which the vote is submitted (if lang set to ‘ar’, then microAppDisplay value must be Arabic).<br><br>If the vote is submitted for the overall application (i.e. the user clicks on ‘This App’, then ‘microAppDisplay’ value would be empty string). |

## 5. Appendix B – Sample Request Parameters

The following table shows a sample values for each request parameter

| Parameter Name | Sample Value |
|---|---|
| json_payload | `{ "user": {"source":"ANONYMOUS"}, "transaction": {"transactionID":"12345", "gessEnabled":"false", "serviceCode":"", "serviceDescription":"demo transaction", "channel":"SMARTAPP"}, "header": {"timestamp":"22/01/201508:19:06", "serviceProvider":"DSG", "themeColor":"green", "microApp":"nameOfMicroApp"} }` |
| client_id | dsg123 |
| signature | b13e873f22e537f7978ef420f16d864a40633200ca45bb8797d058c9953c87f868 aa20ce287f399f8bf16086f87bb0eb680a8900d5890896672104facc40d6fc |
| lang | en |

# 6. Appendix C - Security Requirements

DSG will provide the following credential for every service provider; the Service provider will use these credentials from Happiness Index Web gadget and Smart Application SDK.

- **Service Provider Code**
- **Username**
- **Service Provider Secret**

The following http headers should be passed to access the Happiness Index APIs

1. **Timestamp**, The client's current system timestamp using the following format, DD/MM/YYYY HH24:MI:SS (i.e. UTCtimezone)

2. **Random**, this is a 16 Hexadecimal digits String

3. **Nonce**, This header is used to prevent reply attacks and is calculated by using the following formula
   Nonce=HASH("SHA512", Random header| Timestamp header| Service Provider Secret)

4. **Signature** , This header is used to assure data integrity and that Message body was not tempered during transmission "Man in the middle attack" and calculated by the using the following formula
   Signature= HASH("SHA512", Message Body | Service Provider Secret)

## Example

A client with the following credentials "Provided by DSG to the client" want to invoke the API on 30/12/2014 12:25:11

**Service Provider Code:** DEWA
**Username (i.e. Client_ID):** dewa_user_happiness_index
**Service Provider Secret:** 123FA34CD1223

### 7.1 How to calculate the Timestamp http header

The Timestamp http header is equal to the client current system date

Timestamp: 30/12/2014 12:25:11

### 7.2 How to calculate the Random http header

The Random http header is a 16 Hexadecimal digits random string

Random: random (“HEX”, 16 digits)
Random: CD35545FDAB33CDF

## 7.3 How to calculate the Nonce http header

Nonce: HASH(“SHA512”, Random header| Timestamp header| Service Provider Secret)

Nonce: HASH(“SHA512”,”CD35545FDAB33CDF|30/12/2014 12:25:11|123FA34CD1223”)

Nonce:
6283DA2466B9C4A7085CD7A6BF7942AD7B854F33233132AB4734A2B48DB8F22368993
373479308642F39B004B7B010A124A844A2CF04DC5B0C0859B0F6F8C5DB

## 7.4 How to calculate the signature http parameter

Signature: HASH(“SHA512”, Message Body | Service Provider Secret)