API MONETIZATION PLATFORM VODAFONE INDIA 1.0

Agile Service Enablement 2.0

OneAPI Common Information Guide

Document Version 3.0

This OneAPI Common Information Guide has been prepared for Vodafone India.

Copyright © Intel Corporation 2014; all rights reserved.

Document Revision History

Rev #	Date	Description
1.0	7 March 2013	Initial issue of the OneAPI Common Information Guide.
2.0	27 June 2014	Added POL-018 for UUP lookup error.
3.0	24 October 2014	Restructured and revised. Removed POL-018 – replaced by SVC0004. Updated Address Normalization section, updating its service error reference from SVC0002 to SVC0004. Added POL-008 and details of HTTP response codes returned for policy errors.

Table of Contents

1 Introduction 4
1.1 Audience 4
1.2 Requirements 4
2 Developer Access 5
2.1 Using Basic Authentication 5
2.2 Using REST APIs from a Browser 5
2.3 Using REST APIs from an Application 5
3 Response Codes & Exceptions 7
3.1 Response Codes 7
3.2 REST Exceptions 7
3.2.1 Service Exceptions 8
3.2.2 Policy Exceptions 9
4 Address Normalization 11
4.1 Address Normalization Errors 11

1 Introduction

This guide provides information that is common to all the OneAPI services offered on the API Monetization Platform (AMP). It describes how to access the services and lists details and possible solutions to response codes and exceptions that may be encountered.

1.1 Audience

The intended audience for this document is application developers who wish to use the services offered on AMP to develop their own APIs, which they can subsequently deploy for use by their customers.

1.2 Requirements

The following table is a summary of the requirements needed to use the AMP APIs.

1	The **service endpoints** made available to you when you registered with your service provider.
2	The **API documentation** specific to the APIs you wish to use.
3	Knowledge of an **application programming language**. Your service provider may provide you with client libraries and bindings.
4	To receive SMS sent to your application by end users, you will need to obtain a registrationId (e.g. short code or similar identifier) to identify your application to the network, which ensures correct routing. For more information on this, contact your service provider.

2 Developer Access

This section provides the detailed information required to successfully access the services available on AMP.

2.1 Using Basic Authentication

In order to use the various AMP services, basic authentication is required, i.e. an application username ID and password. The information below is applicable to all of the APIs.

In all cases, attempting to use the service APIs without basic authentication credentials, results in the following HTTP error:

```
HTTP/1.1 401 Unauthorized
```

For information on obtaining authentication details, refer to the 'Developer Quick Start Guide', which is available from the Resources section on the Partner Portal.

2.2 Using REST APIs from a Browser

Enter the API URL in a browser. An example is shown below:

http://myhost:8181/cxf/sms/2_0/smsmessaging/outbound/tel%3A%2B5141516242/requests

Enter the application username and password when prompted to access the required certificate from the server.

2.3 Using REST APIs from an Application

From within a Java application, code similar to the following is required to set up basic authentication:

import java.net.HttpURLConnection;
import org.apache.geronimo.mail.util.Base64;
final static String username = "partner";
final static String password = "partnerpassword";
final static String url = "http://your-endpoint-here";
 	
final static String credentials = username + " : " + password;
final static String authHeaderValue = new String(Base64.encode(credentials.getBytes()));
 	
HttpURLConnection con = (HttpURLConnection)new URL(url).openConnection();
	Replace "username" and "password" with your application username and password.

con.setRequestProperty ("Authorization", "Basic " + authHeaderValue);	

3 Response Codes & Exceptions

3.1 Response Codes

HTTP response codes are used to indicate:

- **200** – Success!
- **400** – Bad request; check the error message for details. Generally returned for service exceptions.
- **401** – Authentication failure, check your authentication details
- **403** – Forbidden; please provide authentication credentials. Generally returned for policy exceptions.
- **404** – Not found: mistake in the host or path of the service URI, including a problem with the MSISDN supplied within the URL path
- **405** – Method not supported: for example you mistakenly used a HTTP GET to create an SMS instead of a POST
- **500** – The server encountered an unexpected condition. This could be incorrect authentication details or limited user permission
- **503** – Server busy and service unavailable. Please retry the request.

For more details, refer to http://www.ietf.org/rfc/rfc2616.txt.

3.2 REST Exceptions

The **requestError** object contains either a **serviceException** or a **policyException:**

- A **serviceException** describes the reason why the service cannot accept the request; in the example below, because of an invalid event ID.
- A **policyException** object means that the request syntax is valid, however an operator policy has been broken, e.g. you are requesting to charge an amount that exceeds a limit that the operator has set.

The two error types use the same body structure, which includes the following pairs:

- messageId pair – to denote the error type, for example, SVC0001, POL0001
- text pair – to describe the error, including the error code where applicable, for example POL-014
- variables pair – to indicate any specific cause of the error, where applicable. The variables relate to the parameter placeholder(s).

Examples are copied below, with details of the exceptions and error codes in the next sections.

Service Exception Example

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Headers: {Content-Type=[application/json], Date=[Thu, 27 Jun 2013 16:15:28 GMT]}
```

```
{
"requestError" : {
  "serviceException" : {
   "messageId" : "SVC0002",
   "text" : "Invalid input value [DWAP_VCU0001_PPU_BAR] for parameter [code]"
  }
 }
}
```

Policy Exception Example

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
Headers: {Content-Type=[application/json], Date=[Thu, 28 Jun 2014 16:15:28 GMT]}
{
 "requestError" : {
  "policyException" : {
   "messageId" : "POL0001",
   "text" : "A policy error occurred. Error code is POL-014: Destination White List is enforced and address is not in
Destination White List.",
   "variables" : [ "POL-014", "Destination White List is enforced and address is not in Destination White List." ]
  }
 }
}
```

3.2.1 Service Exceptions

The following are service exception types which may be thrown when an operation fails:

Table 1: Service Exceptions

Error	Explanation
SVC0001 – Service error occurred	A service-related error has occurred as a result of a client invocation on the service. This category can be used for implementation-specific errors. Contact the support team.
SVC0002 – Invalid input value	An input parameter value is not of the expected type. Check the parameter types and re-submit your request.
SVC0004 – Invalid address(es)	The address, to which the request was sent, does not exist, or the address type is unknown, or the associated subscriber profile cannot be found, or the subscriber is not active. This may be returned

	for Address Normalization errors. Update your request and re-submit. See more details on Address Normalization and reason texts returned for errors in section 4.
SVC0270 – Charge failed	This will be accompanied by the text: 'Charging operation failed, the charge was not applied.' This could be due to, for example, the transaction not being found or payment not allowed, in relation to Advice of Charge. Contact the support team.

3.2.2 Policy Exceptions

A policy exception means that the request syntax is valid, but an operator policy has been broken. It is denoted by the messageId value and defined as follows:

- **POL0001**: Policy error occurred. This exception may be thrown to indicate a fault relating to a policy associated with the service. This category can be used for implementation-specific errors which will be included in the text and variables details of the response. In general, HTTP response code 403 is thrown, unless listed otherwise.

Examples of common policy error codes are listed below:

Table 2: Common Policy Error Code Examples

Error	Explanation
POL-006: TPA exceeded its maximum allowed rate of transactions	The maximum rate of transactions is exceeded. Ensure that the rate of your requests is within the limits set up in your SLA, e.g. 10 TPS (Transactions Per Second).
POL-008: TPA is invalid	The Third Party Application authentication details are incorrect. Check your basic authentication username and password are correct and re-submit your request. • HTTP response code 401 (Unauthorized) is returned if no credentials are provided. • HTTP response code 403 (Forbidden) is returned if incorrect credentials are provided.
POL-010: Subscriber target not authorized	Authorization from the subscriber has not been obtained through the Consent service. Please request authorization for this subscriber through the Consent Service.
POL-014: Destination White List is enforced, and address is not in Destination White List	A white list is enforced and the number is not in the white list. Check your SLA details.

POL-016: Max Requests is enforced, and max requests has been exceeded	The maximum number of requests for this service is exceeded. Contact the support team.
POL-017: Operation is not allowed	The method/operation is not supported in your current SLA. Check your SLA and use a method that is supported.
POL-020: Max Message Length is enforced, and max message length has been exceeded	A maximum message length policy is in place and you have exceeded this. Check you SLA for the maximum message length, update your message and re-submit your request.
POL-021: Min Message Length is enforced, and message length is less than min allowed	A minimum message length policy is in place and your message length is less than this minimum. Check you SLA for the maximum message length, update your message and re-submit your request.
POL-022: Receipting is enforced, and receipting has not been enabled	A receipt has been requested but it is not enabled for this service. Remove the receipt request and re-submit your request.
POL-028: Destination Black List is enforced, and address is in Destination Black List	A black list is enforced and the number is in the black list. Check you SLA details.
POL-030: Sender Black List is enforced and address is in Sender Black List	Sender address blacklisting is enforced for messaging services, where blacklisted addresses are omitted. This error occurs when *all addresses* on the request exist on the configured sender blacklist.
POL-031: Sender White List is enforced and address is not in Sender White List	Sender address whitelisting is enforced for messaging services, where non-whitelisted addresses are omitted. This error occurs when *none of the addresses* in the request exist on the configured sender whitelist.
POL-040: Max Destination Addresses is enforced and maximum destination addresses has been exceeded	A maximum destination address limit is enforced and it has been exceeded. Check your SLA for the limit and re-submit your request.
POL-051: The callback endpoint parameter is not prefixed by HTTPS	The policy requires callback endpoints to use HTTPS; the delivery receipt URI in the request does not start with HTTPS. Resubmit using HTTPS.

4 Address Normalization

This section provides outline information on Address Normalization.

The address normalization logic is handled partly by ASE custom code, and partly by Google Phone Number Library. The rules are complex, combining a number of factors such as address type, value length, prefixes, etc.

When the Address Normalization Policy is applied to a *service*, an address received by the service is first inspected for address type, by using the following rules applied in the order set out below.

1. If the supplied address is null, empty or contains only whitespace then it is of type **BLANK**.
2. If the supplied address starts with **tel:+**, **tel:** or **+** and then contains only digits (or spaces or hyphens) followed by an optional extension (e.g. **;ext=123**) then it is an international number of type **TELEPHONE**.
3. If the supplied address starts with **short:** or contains only digits, and has a length between 5-8, then it is of type **SHORT_CODE**.
4. If the supplied address starts with **sip:** then it is of type **SIP**.
5. If the supplied address starts with **acr:**, then it is of type **ACR**.

Addresses falling into the category TELEPHONE are processed by the normalization algorithm determined by the country code and the numbering plan in operation in that country, by referring to the Google Phone Number library.

It is not possible to provide general examples, as the TELEPHONE rules vary greatly from country to country, but you can check an address for address type validity by using the tool available on the Google Phone Number Library website at *https://code.google.com/p/libphonenumber/*.

Addresses not falling into the category TELEPHONE are processed by other algorithms implemented for the service.

4.1 Address Normalization Errors

When an address cannot be normalized by any of the rules, a SVC0004 error is returned:

```
{
 "requestError" : {
  "serviceException" : {
   "messageId" : "SVC0004",
   "text" : " No valid address(es). Invalid address found: [tel:+9198250429999]. Reason: [MSISDN fails normalizer rules – too long!]"
  }
 }
}
}
```

The reason text begins with 'MSISDN fails normalizer rules' and continues with 'invalid country code!', 'too long!' or 'too short!' as applicable. Other possible reason texts, and HTTP response codes returned are listed below:

Table 1: SVC0004 Reason Texts

Reason	Reason Text Returned	HTTP Response Code when the MSISDN is in the message body	HTTP Response Code when the MSISDN is in the URI path
subscriber not found	Subscriber Profile not found!	400	404
unknown address type	Address type is unknown!	400	404
invalid country code	MSISDN fails normalizer rules - invalid country code!	400	404
too long	MSISDN fails normalizer rules - too long!	400	404
too short	MSISDN fails normalizer rules - too short!	400	404
not in an active state (suspended)	Subscriber is not active!	400	**400**
MSISDN does not contain the configured number of digits, (default value: 10)	Invalid address length!	400	404

.

End of Document