API MONETIZATION PLATFORM 4.0/4.1

AGILE SERVICE ENABLEMENT 1.0

Aepona v3.0 Consent REST

Document Version1.1

Document Properties

Document ID	API_Aepona3-0_Consent-REST
Document Version	1.1
Approval Date	7 May 2013
Originator	Luigi Cirillo
Approver	Jim Beggs
Version Information	v1.1, following initial issue with AMP 4.0/4.1 v1.0 in Nov 2012

Copyright

Table of Contents

1 Consent REST Overview

The Consent interface allows an application to request the consent of a subscriber to perform actions against that subscriber, e.g. to obtain the subscribers location or send the subscriber an SMS or MMS.

In addition the API allows direct management of subscriber consent via the 'Deposit API' methods: create, update and delete consent, that may be used when the application is trusted to manage consent, e.g. via its own subscriber dialogue mechanism.

Access to these methods for applications will be dependent on "operation set" policy. This policy defines a set of allowed operations, so any invoked service method must be within this set, in order to be completed. Enablers for consents are managed at the application level, so enabler name is not supported in requests but will be working behind the scenes for all those configured.

Please note that throughout this document, the examples may be shown WITHOUT URL encoding for readability purposes.

! Throughout this document, the examples may be shown WITHOUT URL encoding for readability purposes, e.g. if the address "tel:+123456789" is in the URL example, this should be encoded as "tel%3A%2B123456789", where the character ":" is "%3A" and the character "+" is "%2B"

! Data types indicated as XSD in parameter tables refer to the standard XSD type. Relevant information may be obtained from http://www.w3.org/2001/XMLSchema.

2 Authentication

A server side certificate is required plus HTTP Basic Authentication.

For more information, refer to the "Developer Access' section in the 'OneAPI v2.0 Common Information Guide'.

3 Methods

Consent may only be accessed via the REST API (described in this document). The following methods are available:

- Create (Deposit) Subscriber Consent- section

- Update Subscriber Consent previously deposited via the create operation - section
- Delete Subscriber Consent previously deposited via the create operation - section
- Request Consent from Subscriber- section

- Query Subscriber Consent Status - section

> ! The Consent API supports application/x-www-form-url encoded for POST operations. The response content type is application/XML.

3.1 URIs

The URIs of the resources are as follows:

- Creating (depositing) a consent for a subscriber

  https://{serverRoot}/PrivacyService/rest_v3_0/sms

- Updating a consent previously deposited by the create operation

  https://{serverRoot}/PrivacyService/rest_v3_0/sms/{address}&{channel}&{status}&{expiryTime}

- Deleting a consent previously deposited

  https://{serverRoot}/PrivacyService/rest_v3_0/sms/{address}&{channel}

- Requesting the consent of a subscriber

  https://{serverRoot}/PrivacyService/rest_v3_0/sms

- Querying the status of the subscriber consent

  https://{serverRoot}/PrivacyService/rest_v3_0/sms/{address}

The variables used in request URLs are described in the relevant sections, except for the common variable {serverRoot}:

Name	Description
{serverRoot}	Server base url: hostname+base path. Base path is optional. *example.com* is used in the examples in this document.

4 Create (Deposit) Subscriber Consent

This method allows the application to 'deposit' a consent that is managed by a mechanism outside of TWS.

4.1 Request

```
POST https://example.com/PrivacyService/rest_v3_0/sms
HTTP/1.1

Accept: application/xml
Content-type: application/x-www-form-urlencoded
address=tel:+447990123456&
operation=createConsent&
channel=IVR&
status=ALLOWED&
expiryTime=2000
```

4.1.1 Request Parameters

Table 1: Create Subscriber Consent - Request Parameters

Parameter	Data Type	Description	Optional
address	xsd:anyURI	The URI of the subscriber to deposit the consent for.	No
operation	xsd:string	Must be 'createConsent' to distinguish from the requestConsent POST method.	No
channel	xsd:string	Must be one of the values checked on "Valid Channel Policy" (See Table 10 for details)	No
status	status	The consent status. (See Table 9 for details).	No
expiryTime	xsd:int	The number of hours until the consent expires.	No

4.2 Response

```
HTTP/1.1 204 No Content
```


5 Update Subscriber Consent

Allows the application to update a consent previously deposited via the create operation.

5.1 Request

```
PUT
https://example.com/PrivacyService/rest_v3_0/sms/?address=tel:+447990123456&channel=EMAIL&status=DENIED&expiryTime=2000

HTTP /1.1

Host: www.example.com
Accept: application/xml
Content-type : application/x-www-form-urlencoded
```

! Query strings should be URL encoded. The address parameter is converted to tel%3A%2B447990123456.

5.1.1 Request Parameters

Table 2: Update Subscriber Consent - Request Parameters

Parameter	Data Type	Description	Optional
address	xsd:anyURI	The URI of the subscriber consent is being updated for.	No
status	status	The consent status. (See Table 9 for details).	No
expiryTime	xsd:int	The number of hours until the consent expires.	No
channel	xsd:string	Must be one of the values checked on "Valid Channel Policy" (See Table 10 for default values)	No

5.1.2 Response

```
HTTP/1.1 204 No Content
```


6 Delete Subscriber Consent

Allows the application to delete a consent previously deposited via the create operation.

6.1.1 Request

```
DELETE
https://example.com/PrivacyService/rest_v3_0/sms/?address=tel:+447990123456&channel=SMS

Accept: application/xml
```

! Query strings should be URL encoded. The address parameter is converted to tel%3A%2B447990123456.

6.1.2 Request Parameters

Table 3: Delete Subscriber Consent - Request Parameters

Parameter	Data Type	Description	Optional
address	xsd:anyURI	The URI of the subscriber the consent is being deleted for.	No
channel	xsd:string	Must be one of the values checked on "Valid Channel Policy" (See Table 10 for default values)	No

6.2 Response

```
HTTP/1.1 204 No Content
```


7 Request Consent from Subscriber

Allows the application to seek consent from a subscriber for various actions, for example, to access the subscriber's location or to be able to send an SMS to the subscriber.

7.1 Request

```
POST https://example.com/PrivacyService/rest_v3_0/sms
HTTP/1.1

Accept: application/xml
Content-type: application/x-www-form-urlencoded
Authorization: Basic<base64 encoded application credentials>

address=tel:+447990123456&
operation=requestConsent&
callbackUrl= http://www.partnersite.com/privacyReceiver
```

! Query strings should be URL encoded. The address parameter is converted to tel%3A%2B447990123456 and the callback URL parameter is converted to http%3A%2F%2Fwww.partnersite.com%2FprivacyReceiver.

7.1.1 Request Parameters

Table 4: Request Subscriber Consent - Request Parameters

Parameter	Data Type	Description	Optional
address	xsd:anyURI	The URI of the subscriber to whom you are seeking consent.	No
operation	xsd:string	This parameter is used only to distinguish this operation from the createConsent operation, using both the POST request method. It may be absent, or anything other than 'createConsent'. The parameter isn't used or traced.	Yes
callbackUrl	xsd:anyURI	The URL to be invoked when the subscriber has given/withheld their consent.	No

7.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Consent status="PENDING"
channel="SMS"/>
```

7.2.1 Response Parameters

Table 5: Request Subscriber Consent - Response Parameters

Parameter	Data Type	Description	Optional
status	status	The consent status. (See Table 9 for details). Please note that expired consents won't be returned.	No
channel	xsd:string	The consent creation channel	No

8 Query Subscriber Consent Status

Allows the application to query the status of a previous consent request.

8.1 Request

```
GET
https://example.com/PrivacyService/rest_v3_0/sms/?address=tel%3A%2B164738
47077

HTTP/1.1
Accept: application/xml
Content-Type: application/xml
Authorization: Basic<base64 encoded application credentials>

address=tel:+447990123456
```

8.1.1 Request Parameters

Table 6: Query Subscriber Consent - Request Parameters

Parameter	Data Type	Description	Optional
address	xsd:anyURI	The URI of the subscriber to whom you are seeking consent.	No

8.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Consent status="PENDING
channel="SMS"/>
```


8.2.1 Response Parameters

Table 7: Query Subscriber Consent - Response Parameters

Parameter	Data Type	Description	Optional
status	status	The consent status. (See Table 9 for details).	No
channel	xsd:string	The consent creation channel (See Table 10 for details)	No

9 Consent Callback

When the subscriber gives their consent or withholds it, your service at the callback URL (which you specified in your original privacy request) is invoked. Your service must respond with "`204 No Content`".

9.1 Request

```
POST /<privacyReceiver>
HTTP/1.1

Accept: application/xml
Content-type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<privacyReceipt>
     <subscriber>tel:+447990123456</subscriber>
     <status>ALLOWED</status>
</privacyReceipt>
```

9.1.1 Request Parameters

Table 8: Consent Callback Request - Parameters (consentEventType)

Parameter	Data Type	Description	Optional
subscriber	xsd:anyURI	The address (URI) of the subscriber who responded to your privacy notification.	No
status	status	Specifies if the subscriber has granted access or denied access. (See Table 9 for details).	No

9.2 Response

```
204 No Content
```

9.2.1 Response Parameters

N/A

10 Status Responses

The following Consent Response statuses may be returned:

Table 9: status types

Status	Description
PENDING	The subscriber has been notified by text message that the application wishes to use their number. They have not yet replied and the request has not yet expired.
ALLOWED	The subscriber has replied to the notification text message and has chosen to allow the application to use their number.
DENIED	The subscriber has replied to the notification text message and has chosen not to allow the application to use their number.
EXPIRED	The subscriber has not replied to the notification text message, and the time allowed for replying has passed or the consent session has expired. You can request again if the status is expired.

11 Channel Types

The following are the default channels that may be used:

Table 10: channels

Status	Description
EMAIL	The subscriber has submitted his consent via EMAIL
IVR	The subscriber has submitted his consent via Interactive Voice Response
SANDBOX	Sandbox purposes
SMS	The subscriber has submitted his consent via Short Message
UNKNOWN	The subscriber has submitted his consent via any other way
WAP	The subscriber has submitted his consent via WAP
WEB	The subscriber has submitted his consent via Web Form.

12 HELP & INFO Message Support

12.1 Help & Info Policy

An application can choose to support HELP & INFO messages from the subscriber or rely on more generic HELP & INFO messages configured through the Consent service.

To support HELP & INFO messages an application must request the administrator to enable the "Help Info Messages" Policy.

If an application chooses not to independently support HELP & INFO messages, upon receiving a HELP & INFO messages, a HELP or INFO is returned via SMS directly to the subscriber. This is not impacting the consent request but simply provides automatically an SMS send service.

If an application chooses to support HELP & INFO messages and a HELP or INFO message is received from a subscriber, an event will be published to the original callbackUrl appended with a "/keyword". If the original callbackUrl was http://callbackUrl/appName, the HELP or INFO event will be published to http://callbackUrl/appName/keyword. This doesn't impact the consent request and no SMS service is provided.

```
Host: www.partnersite.com
Accept: text/plain
Content-type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<privacyReceipt>
     <subscriber>tel:+447990123456</subscriber>
     <messageType>messageTypeHelp</messageType>
     <message>HELP blah, blah, blah</message>
</privacyReceipt>
```

13 Response Codes & Exceptions

13.1 Response Codes

HTTP response codes are used to indicate:

- **200** – Success!
- **400** – Bad request; check the error message for details
- **401** – Authentication failure, check your authentication details
- **403** – Forbidden; please provide authentication credentials
- **404** – Not found; mistake in the host or path of the service URI or consent not found, check the error message for details

- **500** – The server encountered an unexpected condition. It could be incorrect authentication details or limited user permission
- **503** – Server busy and service unavailable. Please retry the request.

Those apply to all methods. For more details on the response codes, refer to http://www.ietf.org/rfc/rfc2616.txt.

13.2 Exceptions

```
HTTP/1.1 403 Forbidden
Content-Type: application/xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<error> A policy error occurred. Error code is POL-014: Destination White
List is enforced and address is not in Destination White List.</error>
```

This section lists the available error codes, the possible reasons why the exception may have occurred, and possible solutions.

13.2.1 Service Exceptions

The following exceptions may be thrown when an operation fails:

Table 11: Service Exceptions

Error	Explanation	Applicability
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.	All requests
SVC0002 – Invalid input value	An input parameter value is not of the expected type. Check the parameter types and re-submit your request.	All but Delete Subscriber Consent and Query Subscriber Consent.
SVC0004 – No valid address(es)	The requested terminal device address does not exist. Use an address that exists.	All requests

13.2.2 Policy Exceptions

A policy exception means that the request syntax is valid, however an operator policy has been broken.

POL0001 - Policy error occurred

The above exception may be thrown to indicate a fault relating to a policy associated with the service.

It is a generic error that can be sub-categorized for the implementation-specific errors following:

Table 12: Policy Error Codes

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.
POL-014: White List is enforced, and address is not in White List	A white list is enforced and the number is not in the white list. Check your SLA details.
POL-015: Black List is enforced, and address is in Black List	A black list is enforced and the number is in the black list. Check you SLA details.
POL-016: Max Requests is enforced, and max requests has been exceeded	The maximum number of requests for this service is exceeded. Check you SLA details.
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.

14 Consent Sandbox

The Consent service contains a sandbox for use in testing applications using consent, available at http://<domain>/PrivacyService/rest/sandbox. This currently uses built in responses and does not make use of the sandbox data service.

While the application is in Sandbox state, the Consent Endpoint will be directed at the Consent Sandbox service. When the application transitions to Production state, the Consent Endpoint will point to the Production Service.

The behaviour of the sandbox is covered in the following subsections.

14.1 Create Consent

Creates consent using the specified request parameters.

All statuses are expired and removed after 5 minutes.

14.2 Update Consent

Updates (existing) consent using the specified request parameters.

14.3 Delete Consent

Deletes (existing) consent using the specified request parameters.

14.4 Request Consent

1. All requests will return an initial PENDING status.
2. A secondary status is scheduled to return after a number of minutes defined by the last digit of the supplied subscriber number.
   - If the last digit of the supplied subscriber number is even, an ALLOWED status will be returned to the callback URL.
   - If the last digit of the supplied subscriber number is odd, a DENIED status will be returned to the callback URL.
3. All statuses are expired and removed after 5 minutes.

14.5 Query Consent

This returns the current status according to the rules above for requesting consent.

If the requested subscriber does not have a consent state associated, a 'Consent Not Found' error will be returned.

15 References

For WSDLs or XSDs, SSL certificates, API documentation and code samples, please refer to http://developer.aepona.com

For data types details, please refer to http://www.w3.org/2001/XMLSchema.

For "server side certificate" and "HTTP Basic Authentication", please refer to the "Developer Access' section in the 'OneAPI v2.0 Common Information Guide'.

For HTTP response codes details, please refer to http://www.ietf.org/rfc/rfc2616.txt.

