TECHNOLOGY

AMEX API Management Gateway
OAuth Integration for External Client

Integration Architecture & Services

Version 1.0

Table of Contents

Document History

Revision #	Reviewer	Review Date	Change Detail
0.1	Prahalad Tarigopula	12/15/2015	First draft creation.
0.2	Gary St. Lawrence	1/18/2016	T&C review of template and format.
0.3			Manager review of document.
0.4	Prahalad Tarigopula	01/25/2016	Updated with addition of error codes.
0.4	Srinidhi	04/14/2016	Updated with Refresh token recommendation.
0.5	Srinidhi		Added XML support for access token.
1.0	Deborah Reece	6/6/2017	Re-formatted. Reviewed, and revised document provided by Aju.

AMEX API Management Gateway – Overview

The AMEX API Management Gateway supports the OAuth (Open Authorization) protocol security model. The OAuth framework enables the client application to obtain limited access to server resources on behalf of the resource owner.

OAuth orchestrates approval interaction between the resource owner and the HTTP service. OAuth provides an access token that authorizes the sharing of specific account information. The client uses this access token to access the resource hosted by the resource server.

In order to access APIs, the Partner must complete the onboarding process. After the application is provisioned and registered in the API Management Gateway, the Partner receives a unique client ID key and secret for authenticating API access with OAuth.

API Gateway Onboarding – Application Registration

The Amex API Management platform facilitates on-boarding of the Partner onto the API Management Gateway through the following steps:

- The Partner successfully completes on-boarding.
- A client ID key and secret are generated and shared with the Partner via a secure email.
- The Partner utilizes the provided client key and secret to get the access token used to consume APIs through the gateway with OAuth.

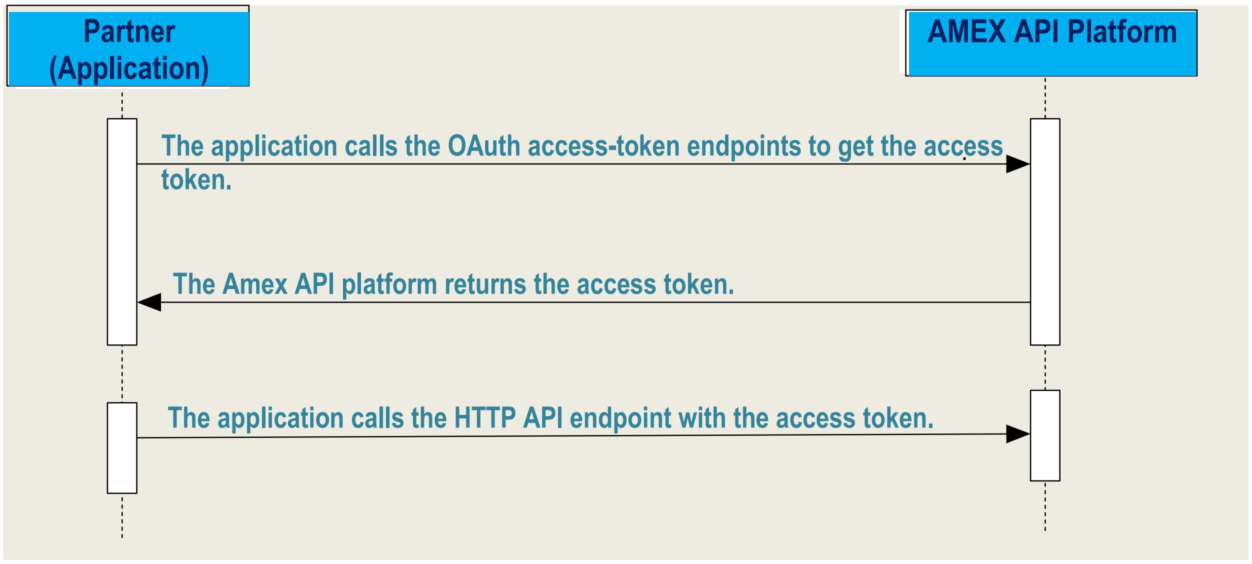
OAuth Access Token – API Flow

Access to Amex APIs is obtained through the following steps:

- The Partner application utilizes the client ID and secret to call the OAuth API access token endpoints and obtain the access token.
- The API Management platform returns the access token to the Partner application.
- The Partner application uses the Amex issued access token to call the HTTP access endpoints for the desired API.

NOTE: The access token has a limited life-span and must be used within the validity period (i.e., two hours). It’s the Partner’s responsibility to track the validity of the access token. If the access token expires, then the Partner can request a refresh token to gain access to the API.

Figure 1. API Flow – Access Token


Partner (Application)
AMEX API Platform
The application calls the OAuth access-token endpoints to get the access token.
The Amex API platform returns the access token.
The application calls the HTTP API endpoint with the access token.

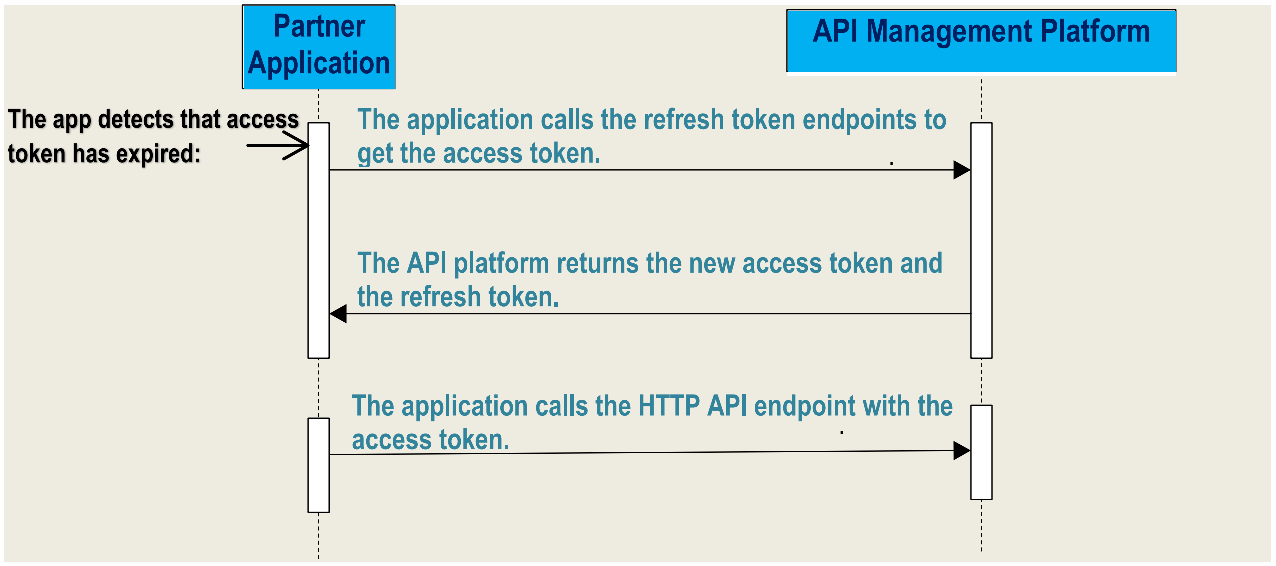
Refresh Token – API Flow

- After successful authentication if the Partner application does not use the access token to call the API within two hours, then the authorization server issues a refresh token in the response payload for the access token.
- The Partner then uses the refresh token to call the OAuth API refresh endpoints and receive a new access token.

Note: The refresh token must not be sent to the resource server, and is only intended for use with the authorization server to obtain the new access token.

- If the Partner application needs to keep the refresh token for a longer duration (e.g., the OAuth secret assigned to the application), the refresh token should be stored in a secure data store.
- The refresh token must be used only in the Authorization Code grant type (i.e., 3-legged OAuth flow), not in the Client Credentials or Password grant types.
- The Partner application extends the access token life using the following steps:
  - The application detects that the access token has expired.
  - The application calls the OAuth refresh token endpoints.
  - The API Management platform returns the new access token and the refresh token.
  - The application calls the HTTP API endpoints with the access token.

Figure 2. API Flow – Extend Lifetime of Access Token


Partner Application
API Management Platform
The app detects that access token has expired:
The application calls the refresh token endpoints to get the access token.
The API platform returns the new access token and the refresh token.
The application calls the HTTP API endpoint with the access token.

API Management Platform – OAuth Endpoints

Figure 3. OAuth Endpoints

Token Type	Grant Type	Access and Refresh Token Endpoints
mac	Password (Resource Owner Password) Client Credentials (Application/ App credentials) Authorization Code (Code provided by Authorization Server)	<u>QA</u> • **Access token:** https://api.qa.americanexpress.com/apiplatform/v2/oauth/token/mac • **Obtain access token by refresh token:** https://api.qa.americanexpress.com/apiplatform/v1/oauth/token/refresh/mac <u>**Production**</u> • **Access token:** https://apigateway.americanexpress.com/apiplatform/v2/oauth/token/mac • **Obtain access token by refresh token:** • https://apigateway.americanexpress.com/apiplatform/v1/oauth/token/refresh/mac

Authentication

The Partner must perform the authentication steps listed in the sections below to access the API Management Gateway.

Call OAuth Token Endpoint – Get Access Token

- The Partner application must call the OAuth token endpoint to get the access token.
- The access token will be in JSON format (by default); however, the consumer can get the access token in "XML" format by passing one of the following HTTP headers:
  - "`Accept=application/xml`"; or
  - "`Accept=text/xml`"

Example: Call Access Token Endpoint to Get Access Token

- **URL (QA):** https://api.qa.americanexpress.com/apiplatform/v2/oauth/token/mac
- **Header:**
  - **Content-Type**= "application/x-www-form-urlencoded"
  - **Authentication**= "MAC id="client id value",ts="time stamp generated by client(in unix epoch time format)",nonce="unique identifier string",mac="request MAC generated using HMAC SHA256 algorithm"
  - **X-AMEX-API-KEY**= "apikey assigned to your application"

Figure 4: Authorization Header Parameters

Field Name	Mandatory	Field Description	Field Type	Field Values
id	Yes	Client ID Value	String	da170c7d-c773-4553-8c8e-c3bc6ee44c7b
ts	Yes	Client Generated Time Stamp (Unix Epoch format)	String	1365602280
nonce	Yes	Unique Identifier String	String	1365602280:AMEX
mac	Yes	Request MAC	String	Request MAC generated using HMAC Sha256 algorithm

Figure 5: Post Body for Grant Type "client_credentials"

Field Name	Mandatory	Field Description	Field Type	Field Values
grant_type	Yes	Defines the Input Grant Type	String	**Client_credentials**
scope	Yes	Scope Values	String	**Example:** **Name READ. AMEX will share this value string prior to the connectivity test.**

Figure 6: Post Body for Grant Type "password"

Field Name	Mandatory	Field Description	Field Type	Field Values
grant_type	Yes	Defines the Input Grant Type	String	**client_credentials**
scope	Yes	Scope Values	String	**Example:** **Name READ. Amex will share this value string prior to the connectivity test.**
userName	Yes	User ID	String	**AMEX will share this value string prior to the connectivity test.**
userPassword	Yes	User Password	String	**AMEX will share this value string prior to the connectivity test.**

Figure 7: Post Body for Grant Type "authorization_code"

Field Name	Mandatory	Field Description	Field Type	Field Values
grant_type	Yes	Defines the Input Grant Type	String	Client_credentials
scope	Yes	Scope Values	String	Example: Name READ. Amex will share this value string prior to the connectivity test.
code	Yes	Authorization Code Value	String	Authorization Code received from the authorization server after the user is authenticated.
redirect_url	Yes	Registered Redirect URL	String	Redirect URL of the Partner. If Redirect URL has query parameters, then the Partner must verify the Redirect URL with AMEX team.

Mac Signature Generation Steps

1. Use the HMAC SHA256 algorithm.
2. Construct the base string using the parameters below in this exact order. After each parameter enter the newline"\n" character(%x0A):
   - `client_id`
   - `ts (timestamp in unix epoch format)`
   - `nonce`
   - `grant_type`
3. Use `client_secret` as the key.
4. Perform base 64 encoding on the output raw data.

Example: Base String

If the client_id="543b5bc5-c4a2", ts="137131201", nonce="137131201" and grant_type="authorization_code" *(\n is only for display purposes)* parameters are used, then the base string would be:

```
543b5bc5-c4a2\n
137131201\n
137131201\n
authorization_code\n
```

Example: Authentication Header

If the above base string and key (client secret)="9985-4321-21" are used, then the authentication header would be: "Authentication" => mac id="543b5bc5-c4a2",ts= "137131201",nonce= "137131201", mac="eQXz94uPpGh1sw3dSi7sQt085Us=".

Example: Message Template Headers

Header:

- **Content-Type:** application/x-www-form-urlencoded
- **Authentication:** Mac id="aabc17cb-89a3-4beae987d030132efb0", ts="1366711099",nonce="1366711099:AMEX",MAC="RUNXQ XRKeitOTERtRHhEcHdsUzl0ZkQ3aU5zPQ=="
- **Post Body:** grant_type=client_credentials&scope=name

Figure 8: Output Response

Field Name	Field Description	Field Type	Field Values
access_token	Access Token	String	Example: 66ee3e2f-5313-4df8-84ad-22640bb33d5f
token_type	Type of Token (Bearer or MAC Token)	String	Example: mac
expires_in	Validity of Token (in seconds)	Integer	Example: 3600
refresh_token	Refresh Token	String	Example: c1d37d08-2bbd-4baa-843a-985a67427a56
scope	Scope	String	Example: READ

Field Name	Field Description	Field Type	Field Values
mac_key	MAC Key	String	Example: caeeaeeb-408a-4d15-ad9f-a9f9d812538b
mac_algorithm	MAC Algorithm	String	hmac-sha-256

Sample Code: Output Format

```
{
 "access_token":"66ee3e2f-5313-4df8-84ad-22640bb33d5f",
 "token_type":"mac",
  "expires_in":3600,
  "refresh_token":"c1d37d08-2bbd-4baa-843a-985a67427a56",
  "scope":" READ",
  "mac_key":"caeeaeeb-408a-4d15-ad9f-a9f9d812538b",
  "mac_algorithm":"hmac-sha-256"
}

If access token is requested as XML then output format will be
<oauth2_token>
   <access_token>66ee3e2f-5313-4df8-84ad-22640bb33d5f
</access_token>
   <token_type>mac</token_type>
   <expires_in>300</expires_in>
   <refresh_token>c1d37d08-2bbd-4baa-843a-
985a67427a56</refresh_token>
   <scope>READ</scope>
   <mac_key> caeeaeeb-408a-4d15-ad9f-a9f9d812538b</mac_key>
   <mac_algorithm>hmac-sha-256</mac_algorithm>
</oauth2_token>
```

Call API/Resource Endpoint with Access Token – Access Resource

- The Partner application must call the API endpoint using the required request and access token.
- The API access token will be in JSON format (by default); however, the consumer can get the access token in XML format by passing one of these HTTP headers:
  - "`Accept=application/xml`"; or
  - "`Accept=text/xml`"

Example: Authorization Header for OAuth Validation

Header:

- **Content-Type**= "application/x-www-form-urlencoded"
- **Authorization**= "MAC id=" access token value",ts="time stamp generated by client(In unix epoch time format)",nonce="unique identifier string",mac="request MAC generated using HMAC SHA256 algorithm"

Figure 9: OAuth Authorization Header

Field Name	Field Description	Field Type	Field Values
id	Access Token Value	String	Example: da170c7d-c773-4553-8c8e-c3bc6ee44c7b
ts	Client Generated Time Stamp (Unit Epoch format in MS Capture timestamp as "long" to avoid truncating)	String	Example: 1444917586626
nonce	Unique Identifier String	String	Example: 1444917586626 (uuid):AMEX
mac	Request MAC	String	Example: Request mac generated using HMAC Sha256 algorithm

OAuth Authorization Header Attributes

The authorization header is the critical component in OAuth validation. The header is a combination of various static and runtime attributes. The attributes below are required to form the authorization header:

- **ts** – Timestamp generated by the client *(in Unix epoch format*).
  - Sample Value: "1444917586626"
- **Nonce** – Unique Identifier generated by the client.
  - Sample Value: "1365602280799:AMEX"*(in "uuid:AMEX" format).*
- **HTTP Method** – HTTP method.
  - Sample Value: **POST**

- **Resource Path** – Resource path of the API being called.
  - Sample Value: `/v1/api/resource`
- **Host** – Host name.
  - Sample Value: `api.qa.americanexpress.com`
- **Port** – Port Number.
  - Sample Value: `443`
- **Mac-key** – Mac-key is sequence of runtime parameter, containing ts, nonce, method, resource path, host, port.
  - Sample Value: `14217729451421772945:AMEXPOST/v1/apis/getme api.qa.americanexpress.com443`
- **Mac-signature** – Mac-signature is HMAC SHA256 algorithm on Mac-key, digitally signed by "access-token" and base64 encoded (to transmit on wire).
  - Sample Value: `q8Y5wnSyTwa1Oi9O4vpu/bCQOmY=`
- **Authorization Header** – MAC authorization token is passed to the resource endpoint in the HTTP header. The Authorization Header consists of the Mac-key and Mac-signature.
  - Sample Value:
    - **Authorization:** `MAC id="V32hkKG", ts="1444917586626",nonce="1444917586626:AMEX",mac="q8Y5wnSyTwa1Oi9O4vpu/bCQOmY="`

Sample Code: Generate MAC Token (Java)

```
String baseString = tsn+nonce+method+resPath+host+port+ext;
// cerate signature
SecretKeySpec key = new SecretKeySpec(mac_key.getBytes("UTF-8"),
"HmacSHA256");
Mac mac = Mac.getInstance("HmacSHA256");  mac.init(key);
byte[] bytes = mac.doFinal(baseString.getBytes("UTF-8"));
String signature = new
String(Base64.encodeBase64(bytes)).replace("\r\n",
"");
String mac_auth_header = "MAC
Id=\""+access_token+"\",ts=\""+ts+"\",nonce=\""+nonce+"\",mac=\""+s
ignature+"
\""
```