**Open API Partner Integration Security Pattern - Oauth Access Token Only**

**Overview**

This solution pattern covers the scenario of integrating a new partner or ABC app to consume backend Restful service.

In this integration pattern, the API provider leverages Layer7 API Platform gateway to manage the API and to securely enable access to the backend service, for a scalable number of consumers/developers.

The solution pattern addresses the following topics:

* Secure communication between consumer's applications to DFS Open APIs gateway.
* Secure communication between DFS Open APIs gateway and DFS service provider.
* Consumer's application's authentication.
* Consumer's application's authorization.
* Carrier authenticated identifier to the subsequent process layers.
* Highly sensitive data encryption (field level encryption).

**Reference Architecture**

This solution pattern is part of the [**System of Engagement Security Architecture**](https://dlife.discoverfinancial.com/docs/DOC-11311). (link to DLife to be updated)

**Solution Pattern Diagrams and Tables**



**Figure 1 - Integration and Interaction Diagram**



**Figure 2 - Client Application Registration Process**



**Figure 3 - Athenication & Authorizatin using OAuth v2 Client Credentials**

**Description of Solution Pattern Components**

This solution pattern aims that DFS partners consume the restful services provided by DFS and involved the following entities. Figure 1 shows the components integration and interactions.

Client Application - The DFS partners' application is called client application.

(I) Here are the activities, functions, or roles played by **the client application**:

* The client application has to be registered via API Portal in order to access DFS provided Open API.
* The client application must communicate with DFS Open APIS gateway over HTTPS protocol.
* The client application must authenticate to DFS Open APIS gateway by using Oauth v2 Client Credentials grant type.
* The client application must submit Oauth v2 access token (Bearer type) in order to access the restful services provided by DFS via Open APIs gateway.
* The client application must encrypt any highly sensitive data before sending them to DFS restful services.

For data confidentiality and integrity, the Load balancer at the data center hosting DFS Open APIs gateway must route client application requests to the gateway nodes over HTTPS protocol.

(II) DFS **Open APIs gateway** plays the following roles:

* Authenticate the client application. If the client application is authenticated, issue an Oauth v2 access token to the client application.
* Control which DFS Open APIs the client application can access[^1].
* Mediate the client application request.
* Route the client application request to the DFS backend services.
* Pass the authenticated client application identifier to the DFS backend services by using JWT token.
* Communicate with DFS load balancer over HTTPS with mutual authentication.

Load balancer at DFS data center routes traffic originated from DFS Open APIs gateway to DFS web server instances over HTTPS with mutual authentication.

DFS web servers at DFS data center routes traffic originated from DFS load balancer to DFS Application servers which host DFS backend restful services over HTTPS with mutual authentication.

(III) DFS **backend restful service provider** plays the following roles:

* Provide API functions exposed via DFS Open APIs gateway, by implementing the required business functions
* Verify the client application identifier carried by JWT token in HTTP header HTTP\_AUTH\_TOKEN.
* Enforce access control required by business functions.
* Decrypt the highly sensitive data sent & encrypted by the client application.

The restful service must ensure that the authentication/authorization logic (e.g. JWT verification):

Is guaranteed to execute prior to any business logic. Is packaged with the application logic and runs within the same memory and class loader (application cannot run without the security module)

This could be accomplished by implementing JEE Servlet Filter or similar techniques, to encapsulate security logic while providing the ability to properly package the security module with the business logic. EA provides a common JWT verification utility for this purpose to help developing your application security module.

**Highly sensitive data encryption** - Any data that's classified as 'highly sensitive' must be encrypted with separate key (FLE - field level encryption) as an *additional* security measure. Follow the up-to-date security recommendation regarding encryption mechanism.

As of the writing of this document, the field level encryption can be done in symmetric or asymmetric mode.

If the size of the data to be encrypted is within the performance acceptable range, the PKI encryption may apply directly; or use PKI encryption to exchange the encryption key which encrypts the actual field level data. With PKI encryption, do not share the public key among different vendors. For json payload, it is recommended to follow JSON Web Encryption (JWE) specification to do field level encryption.

In general, the symmetric encryption outperforms the asymmetric encryption. If use symmetric-key cipher algorithm, e.g. AES-256 with random initialization vector for performance reason, the symmetric key is shared among the client application and DFS backend service provider and MUST be stored such that only the client instance and the DFS backend service provider has access to it.

**JWT Token** - is used to carry the client application id and its issuer from the DFS Open APIs gateway to the DFS backend service provider. The JWT token is set as HTTP header HTTP\_AUTH\_TOKEN of the request to the DFS backend service provider. The JWT token header is {"typ":"JWT","alg":"HS256"}; its claims set is {"iss":"apis.ABC.com","exp":"xxx","sub":"YYY"} where XXX is the token expiration time and YYY is the corresponding client application identifier (at minimum) and may add additional attributes to the JWT claim as needed. The digest algorithm HS256 used here is HMAC-SHA256 which requires sharing a key between DFS Open APIs gateway and DFS backend service provider. The shared key can be issued to each DFS service provider or to the pair of the client application and service instances. It will be determined based on service use cases.

You may contact EA for a common JWT verification utility that you can leverage to code the above security logic.

**Reusable API implementation** - The FLE key must be unique per each App. In order to reuse the same Restful Service to support multiple consumer Apps, these values must not be hardcoded within the application logic. The application must also support more than one key at a time to facilitate no down time during key update/rotation period.

The sections below describe the interaction flow of client application registration, authentication, authorization, request routing over Open API gateway.

**Register Client Application**

Here are the steps to register the application. Figure 2 shows client application registration process flow.

* On the API Portal, an organization (e.g. ABC Network, Apple, Paypal, etc...) is created and has been assigned a set of APIs.
* The organization administrator (Business) registers or invites a developer from the DFS partner.
* The developer of the DFS partner logs into the API Portal and creates a client application. During the creation of the client application, the client application selects which APIs under the organization to be used, make the application type as confidential, and provide the oauth scope of the application.
* The API provider realizes an application is pending for approval and verify the application, such as what APIs to be used by the application, what API plans are used by the application, and if the required oauth scope is allowed. After all of the pending-approval application details are verified, the API Provider approves the application.
* Once the application is approved, the application will have the right scope, client application id and client application secret assigned. The partner of the client application must keep the client application secret in safe place because the client application secret is the credential used to authenticate the application.

Now, the client application registration procedure is completed.

**Authenticate & Authorize Client Application**

For the client application to access DFS APIs, it must authenticate to DFS Open APIs gateway. This solution pattern uses industry standard Oauth v2 Client Credentials grant type to authenticate the client application. Oauth v2 Client Credential grant type is a two steps process - first authenticate the client application by verifying the client id and client secret to issue an access token, secondly the client uses the issued access token to access the DFS service.

**Authenticate & Authorize using Oauth v2 Client Credentials**

* The client application is authenticated and authorized by implementing industry standard Oauth v2 Client Credential grant type. Figure 3 shows authentication, authorization, data integrity/confidentiality using Oauth v2 Client Credentials. First the client application obtains the access token by using the following steps:
* The client application sends authentication request to the token end point of DFS Open APIs gateway over HTTPS with HTTP Action verb POST, e.g. [https://dfs-openApi-gateway-host[:port]/auth/oauth/v2/token?grant\_type=client\_credentials&scope=xxx](https://dta.discoverfinancial.com/communities/api-guild/solutions/openapi-partner-integration-solution-Oauth-Access-Token-Only) where the scope value xxx is the one registered along with the application registration process; the http request has to set HTTP Authorization header as Basic base64Encode(client application Id:client application secret). DFS Open APIs gateway receives the request and follows Oauth v2 Client Credential grant type protocol to authenticate the application, i.e. validate the request having the right client application id, its corresponding application secret, and the matched application scope. If the application is authenticated, the gateway issues the access token and sends it in HTTP response with json payload {"access\_token":"03b55f50-f72d-4693-9df0-b1c2cfc0dc59","token\_type":"Bearer","expires\_in":3600,"scope":"XXX"}where the access token is a unique random number, e.g. 03b55f50-f72d-4693-9df0-b1c2cfc0dc59, the token type is oauth Bearer type, the scope is the approved scope for the authenticated application, and the token expires in a certain amount of seconds (in this example, the token expires in one hour.).

Once the client application successfully obtains the access token, the application can use it to access DFS open API in the following steps:

* The client application constructs request body or query parameters. If the request body or query parameters contain highly sensitive data (e.g. PAN or SSN), the client application has to encrypt those sensitive data before placing them in the request body or query parameters by using a pre-established cryptographic mechanism with DFS service (API) provider. The client application constructs HTTP authorization header with value "Bearer access token" where the access token is obtained from the previous step. The client application sends the HTTP request with the constructed body/query parameters and HTTP headers over HTTPS to DFS Open APIS gateway.
* The DFS Open APIs gateway validates the access token submitted by the incoming request and also validates the scope associated with the access token. If both are valid, the DFS Open API gateway generates a JWT token which has header {"typ":"JWT","alg":"HS256"} and claims set is { "iss":"apis.ABC.com","exp":"xxx","sub":"client-application-id"} where the client-application-id is the identifier of the client application which sends the request. The JWT token signature is generated by applying the client application secret to its header & claims using HMAC-SHA256. The DFS Open APIS gateway sets the JWT token to the value of the HTTP header HTTP\_AUTH\_TOKEN of the request sent to the DFS backend service. The gateway sends HTTP request which has original request body or query parameter plus HTTP\_AUTH\_TOKEN header to the DFS backend service over HTTPS. The DFS Open API gateway communicates with DFS backend service using HTTPS protocol along with client-server mutual authentication.
* The DFS backend service authorizes the request based on provided JWT token carried in HTTP request header HTTP\_AUTH\_TOKEN. If the JWT token is validated and the request is authorized, decrypts the sensitive data included in the request by using the cipher algorithm and key agreed with the client application. The serviceconducts the business functions and constructs the service response. If the response contains highly sensitive data, the DFS service encrypts them by using the cipher algorithm and key agreed with the client application. The DFS service sends the response back to the DFS Open API gateway.
* The DFS Open API gateway receives the response from DFS backend service and forwards it to the client application. If needed, the DFS Open API gateway can mediate the response from DFS backend service before forwarding it to the client application, such as adding new header or transferring some response fields to different formats but not touching the encrypted data fields.
* The client application receives the HTTP response from DFS Open API gateway over HTTPS.

**Technology and Product listing**

Oauth v2 Client Credentials refers to specification at [http://tools.ietf.org/html/rfc6749](https://dta.discoverfinancial.com/communities/api-guild/solutions/openapi-partner-integration-solution-Oauth-Access-Token-Only)

JWT token refers to specification at [http://tools.ietf.org/html/draft-ietf-oauth-json-web-token-23](https://dta.discoverfinancial.com/communities/api-guild/solutions/openapi-partner-integration-solution-Oauth-Access-Token-Only)

**Related Guidelines and Best Practices**

* Open API Service Provider and Developer Guide
* Rest Security Solution Pattern
* REST design guidelines and best practices

**Glossary**

**Note: Also refer to the**[**DFS Glossary\*\***](http://dtoday.discoverfinancial.com/glossary/)

**Revision History and Contributors**

| Revision Date | Approved By | Author | Changes | Revision |
| --- | --- | --- | --- | --- |
| 06/10/2014 |  | Hao Chen |  | 0.1 |
| 07/07/2014 |  | George Mitry |  | 0.2 |
| 07/15/2014 |  | George Mitry/ Hao Chen |  | 1.0 |
|  |  |  |  |  |

**Reviewers/Contributors:**

| Name | Department/Contribution |
| --- | --- |
| Sean Cocat | Information Security |
| Mohammad khan | Information Security |
| ARB Board Members | Architecture Review Board |
| ISTR SRB | Information Security Team Review |
|  |  |

[1]: This pattern addresses application's authorization by the Open API gateway (for user's authorization refer to REST Security Solution Pattern, which includes (coarse-grained url-based as well as fine-grained authorization)