B2B Okta/AWS Cognito - Authentication & Authorization Flow

Overview

This documentation is to help APICoE team learn how works the Apigee Proxy **EITS-Authorization-Oauth2-V2**.

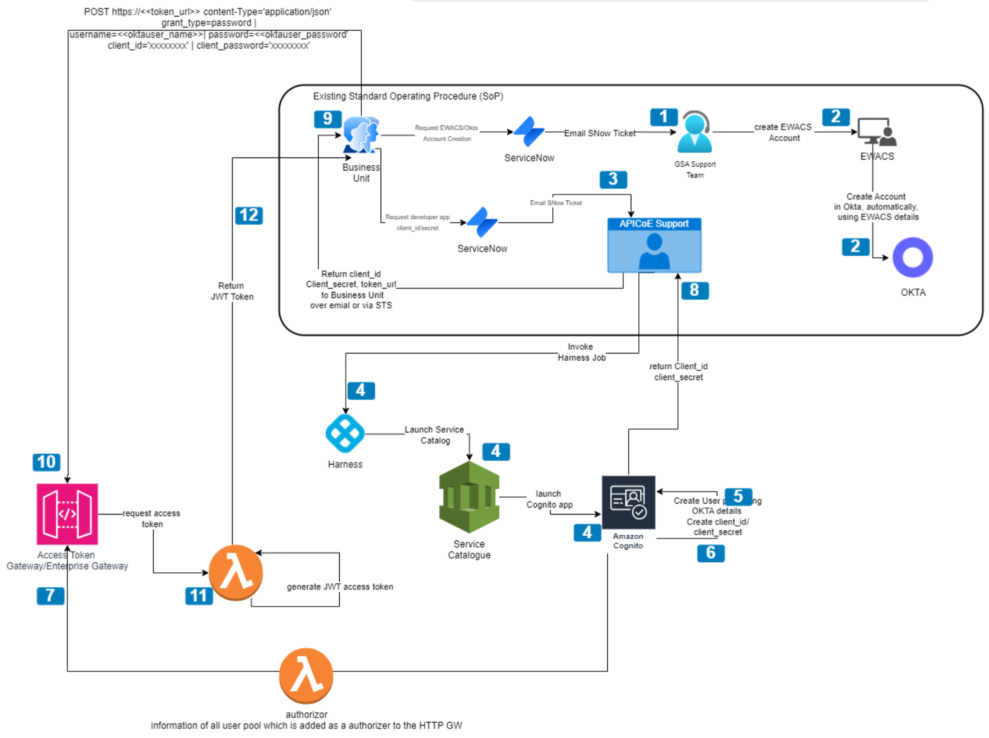
Flow Diagram

Onboarding a new B2B user

Pre-requisites

- The Business User is already vetted.

1	Business Unit will raise a ServiceNow request to onboard a new B2B user
2	The GSA Support team will then go ahead and create an user profile on OKTA
3	Business Unit will request client_id/client_secret, token_url for the developer app from APICoE team
4	APICoE team will Launch Cognito Client App via Service catalog
5	Create Client App in B2B-Cognito user pool based on B2B user profile details defined using drupal forms.
6	Client_id/client_secret is generated for the Cognito Client app
7	APICoE team maps Cognito Client application with respective API GW
8	AWS Cognito returns 'Client_id/client_secret' to APICoE team
9	APICoE team shares client_id/client_secret, token_url with Business Unit
10	Business user uses token_url to invoke Access Token Gateway using 'grant_type=password', Okta user/password
11	Access Token Gateway sends the 'access token request' to Enterprise Gateway
12	Enterprise Gateway returns the 'JWT access token' back to the Business user


POST https://<<token_url>> content-Type='application/json'
grant_type=password |
username=<<oktauser_name>>| password=<<oktauser_password'
client_id='xxxxxxxx' | client_password='xxxxxxxx'
Existing Standard Operating Procedure (SoP)
9
Business Unit
Request EWACS/Okta Account Creation
ServiceNow
Email SNow Ticket
1
GSA Support Team
create EWACS Account
2
EWACS
Create Account in Okta, automatically, using EWACS details
2
OKTA
3
Email SNow Ticket
Request developer app client_id/secret
ServiceNow
APICoE Support
8
Return client_id Client_secret, token_url to Business Unit over emial or via STS
12
Return JWT Token
Invoke Harness Job
return Client_id client_secret
4
Harness
Launch Service Catalog
4
Service Catalogue
launch Cognito app
4
Amazon Cognito
5
Create User p[illegible]ng OKTA details
Create client_id/ client_secret
6
10
Access Token Gateway/Enterprise Gateway
request access token
7
11
generate JWT access token
authorizor
information of all user pool which is added as a authorizer to the HTTP GW

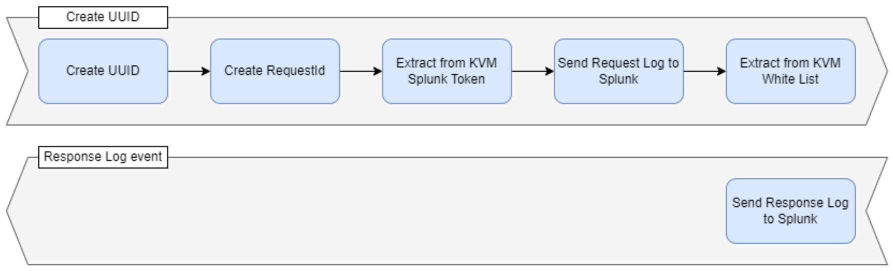
Origin (EWACS) & OKTA Details (for step 2) for Service Now ticket	
Section	**Select/Include**
Region	<<Region Name>>
Description	Request EWACS account created in EWACS Production
Details	First Name: Last Name: Email: DL or team name Telephone User ID User Role: End User EWACS Product Name (to be assigned to the account) List of IPs (as per IP restriction requirement)
Assignment Group	Select "EITS IAM SSO Requests" for US account Select "EITS IAM EWACS Requests" for UK account

Developer App Creation (For step 3) for Service Now ticket	
Section	**Select/Inclue**
Region	Select Appropriate Region
Request Type	Select "Credentials"
Organization	Create appropriate organization
Environment	Select "Production'
Details	Apigee Developer App Creation: API Product Name (please be sure to provide the AWS Cognito Product name and not the EWACS product name. Even though they sometimes are the same, please be sure to validate it): Username (please provide EWACS created username): Email Address (please provide the email address linked to the EWACS username): Account Owner First and Last Name (Please provide as stated in EWACS username): *** If the client needs these credentials sent to an email address other than the one linked to the user, please be sure to state it in

Micro level flow diagram

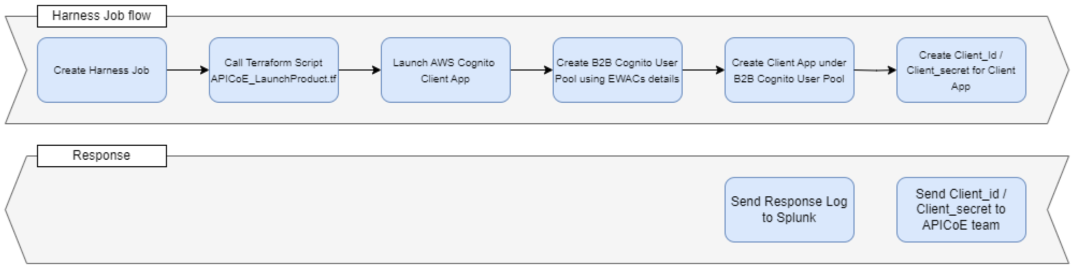
Splunk Integration for Logging

This will invoke a UUID for Splunk to log events


Create UUID
Create UUID
Create RequestId
Extract from KVM Splunk Token
Send Request Log to Splunk
Extract from KVM White List
Response Log event
Send Response Log to Splunk

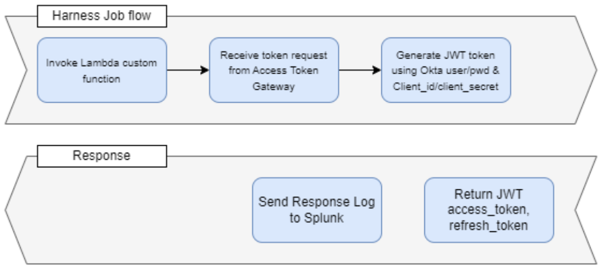
Harness Job flow:

This will trigger a 'APICoE_launchCognitoproduct.tf' that will launch the AWS Cognito Client App, create user pool, client_id/client_secret and send the client_credentials to the APICoE team

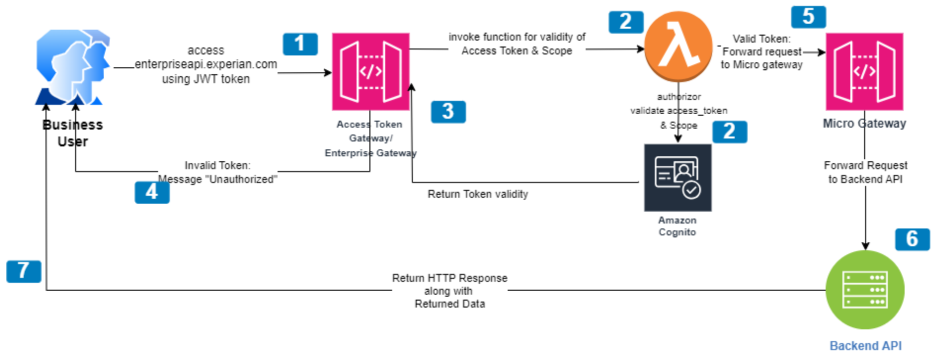

Harness Job flow
Create Harness Job
Call Terraform Script APICoE_LaunchProduct.tf
Launch AWS Cognito Client App
Create B2B Cognito User Pool using EWACs details
Create Client App under B2B Cognito User Pool
Create Client_Id / Client_secret for Client App
Response
Send Response Log to Splunk
Send Client_id / Client_secret to APICoE team

Lambda Custom Token generator:

This will triggered by the a 'APICoE_launchCognitoproduct.tf' that will launch the AWS Cognito Client App, create user pool, client_id/client_secret and send the client_credentials to the APICoE team


Harness Job flow
Invoke Lambda custom function
Receive token request from Access Token Gateway
Generate JWT token using Okta user/pwd & Client_id/client_secret
Response
Send Response Log to Splunk
Return JWT access_token, refresh_token

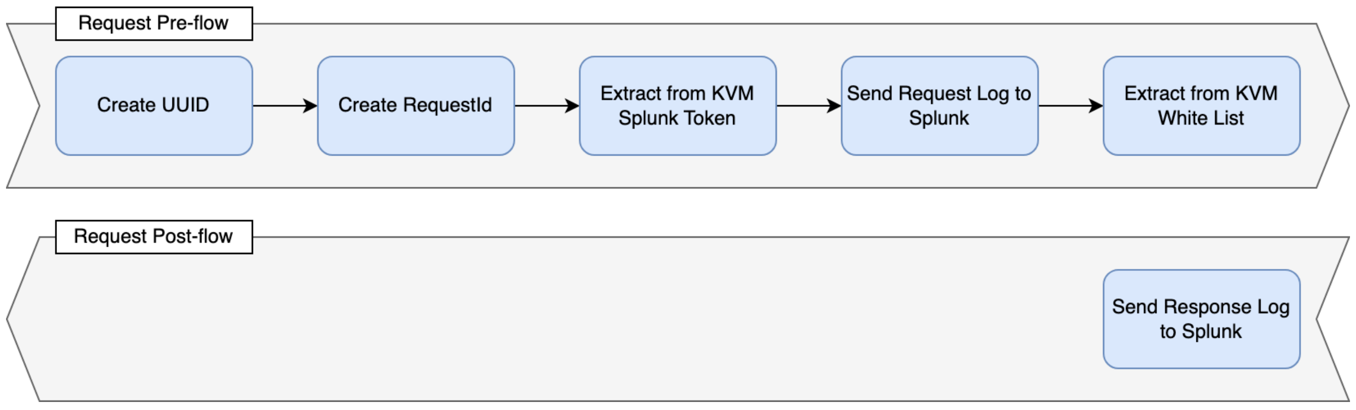
B2B User API Invocation Flow


1
2
5
access enterpriseapi.experian.com using JWT token
invoke function for validity of Access Token & Scope
Valid Token: Forward request to Micro gateway
Business User
Access Token Gateway/ Enterprise Gateway
3
authorizor validate access_token & Scope
2
Micro Gateway
Invalid Token: Message "Unauthorized"
4
Return Token validity
Amazon Cognito
Forward Request to Backend API
6
7
Return HTTP Response along with Returned Data
Backend API

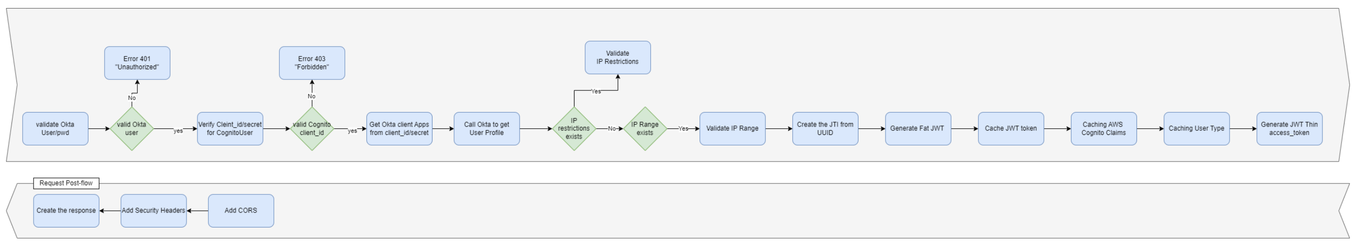
EWACS & OKTA Details	
Section	**Select/Include**
Region	<<Region Name>>
Description	Request EWACS account created in EWACS Production
Details	First Name: Last Name: Email: DL or team name Telephone User ID User Role: End User EWACS Product Name (to be assigned to the account) List of IPs (as per IP restriction requirement)
Assignment Group	Select "EITS IAM SSO Requests" for US account Select "EITS IAM EWACS Requests" for UK account

1	Access API using "enterpriseapi.experian.com" with JWT access_token passed as a request header
2	Validate Access token that API Gateway had generated and shared it with AWS Cognito, during B2B onboarding
3	AWS Cognito returns token_validity to Enterprisegateway
4	API Gateway receives token_validity response. If token is invalid, message "Unauthorized" is sent back to the Business User
5	If the API token is valid, forward API request to relevant micro gateway
6	Micro gateway forwards the API rquest to invoke the Backend API
7	Backend Gateway return HTTP Response along with Returned Data

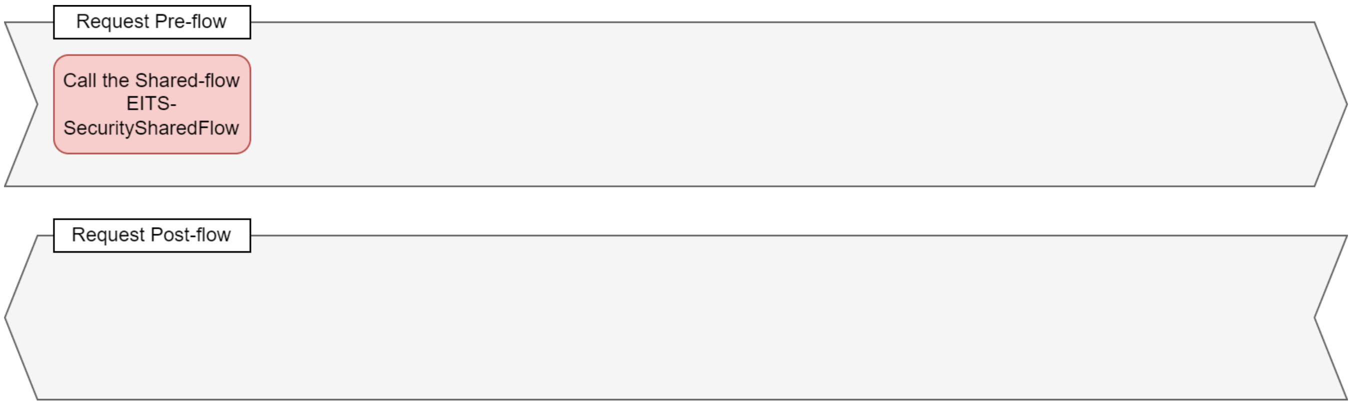
The diagram below enlists the components/policies applicable for any API endpoint requested.


Request Pre-flow
Create UUID
Create RequestId
Extract from KVM Splunk Token
Send Request Log to Splunk
Extract from KVM White List
Request Post-flow
Send Response Log to Splunk

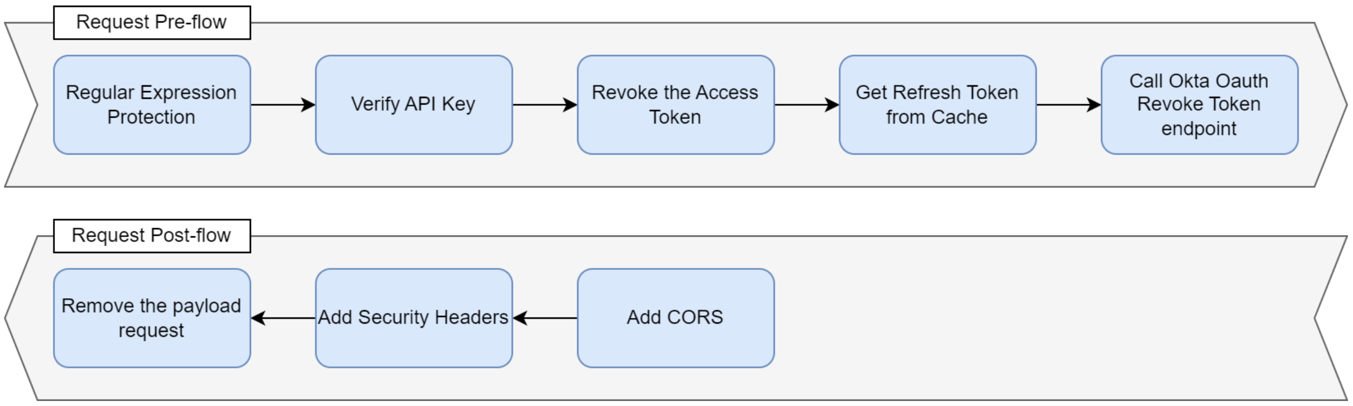
The diagram below represents endpoint POST /token (JSON Web Token) using grant_type=password



The diagram below represents the endpoint GET /checkValidity



The diagram below represents the endpoint POST /revokeToken (Revoke JWT)


Request Pre-flow
Regular Expression Protection
Verify API Key
Revoke the Access Token
Get Refresh Token from Cache
Call Okta Oauth Revoke Token endpoint
Request Post-flow
Add CORS
Add Security Headers
Remove the payload request

Authentication Process

The authentication process to generate the access token depends of which endpoint you will call, the most common endpoint called by the customers is the '**JSON WEB Token Request**' endpoint. This endpoint requires the **client_id** and **client_secret** from AWS Cognito App Client, and the user and password from Okta. To understand more about how to create a request to consume this endpoint, check below the '**Endpoints, HTTP Verb, Headers, Body and Query Params table**' to learn how to do that.

Specifications

Version

- Version: /v2

Path

- Path: /oauth2
- Full path with URL and Version: https://{ENVIRONMENT-ORGANIZATION}-api.experian.com/oauth2/v2

Endpoints, HTTP Verb, Headers, Body and Query Params table

	Description	Endpoint	Verb	Headers	Query Params	Body
1	Check Access Token	/checkvalidity	GET	Name; Value; Required?; 1; authorization; Bearer {{access_token}}; Yes		

2	JSON WEB Token Request	/token	POST	| | Name | Value | Required? |
				| 1 | client_id | {{client_id}} | Yes |
				| 2 | client_secret | {{client_secret}} | Yes |
				| 3 | exp-system-info | {{exp-system-info}} | No |
						JSON:
						
						{
						"username":"{{user_name}}",
						"password":"{{user_password}}"
						}

3	Revoke Access Token	/revoketoken	POST	| | Name | | Required? | | 1 | client_id | {{client_id}} | Yes | | 2 | client_secret | {{client_secret}} | Yes | | 3 | token | {{token}} | Yes |		Empty JSON {}

Response when Successful

	Description	Endpoint	Verb	Status Code	Response Example	Details
1	Check Token	/checkvalidity	GET	200	None	

2	JSON WEB Token	/token	POST	200	{
					"issued_at": "1694809674274",
					"expires_in": "1800",
					"token_type": "Bearer",
					"access_token": "",
					
					}	

3	Revoke Token	/revoketoken	POST	200	None	

Response when Fail

	Description	Endpoint	Verb	Status Code	Response Example	Example

1	Check Token	/checkvalidity	GET	401	{
					"errors": [
					{
					"errorType": "Unauthorized",
					"message": "Access is denied due to invalid access token"
					}
					],
					"success": false
					}
						Error when you are missing the header Authorization

2	JSON Web Token Request	/token	POST	400	{
					"errors": [
					{
					"errorType": "Bad Request",
					"message": "The 'client_id' and 'client_secret' attributes are required"
					}
					],
					"success": false
					}
						Error when you are missing the header client_id OR client_secret

3				401	{
					"errors": [
					{
					"errorType": "Unauthorized",
					"message": "Access is denied due to invalid 'username' or 'password'. For further assistance, please contact Experian Helpdesk at 800-854-7201 or TSCAPISupport@experian.com"
					}
					],
					"success": false
					}
						Error when the body user OR password are wrong.

4				415	{
					"errors": [
					{
					"errorType": "Unsupported Media Type",
					"message": "Content-Type header is unsupported"
					}
					],
					"success": false
					}
						Error when you are missing the body JSON request

5				500	{
					"errors": [
					{
					"errorType": "Internal Server Error",
					"message": "Internal Server Error. If problems persist, please contact apisupport@experian.com"
					}
					],
					"success": false
					}
						Error when you are missing the body user OR password.

6	Revoke Access Token	/revoketoken	POST	401	{
					"errors": [
					{
					"errorType": "Unauthorized",
					"message": "Access is denied due to invalid 'username' or 'password'. For further assistance, please contact Experian Helpdesk at 800-854-7201 or TSCAPISupport@experian.com"
					}
					],
					"success": false
					}
						Error when you are missing the header client_secret

7				401	{
					"errors": [
					{
					"errorType": "Unauthorized",
					"message": "Failed to resolve API Key variable request.header.client_id"
					}
					],
					"success": false
					}
						Error when you are missing the header client_id

8				500	{
					"errors": [
					{
					"errorType": "Internal Server Error",
					"message": "Internal Server Error. If problems persist, please contact apisupport@experian.com"
					}
					],
					"success": false
					}
						Error when you are missing the header token

Proxy Dependencies

Target Server

No target server

KVMs

	Map Identifier	Name	Encrypted	Details
1	FINICITY_TOKEN_CUSTOMIZATION	ACCESS_TOKEN_EXPIRY_MILLIS	Yes	If the request header exp-System-info is equals Yes, the values from this KVM will be used
2		ACCESS_TOKEN_EXPIRY_SEC	Yes	
3	Apigee_JWT_Keys	private	Yes	The values from this KVM are to generate the JWT or JWE
4		public	Yes	
5	OKTA_API_KEY_ENCR	OktaAPIKey	Yes	The value from this KVM is necessary to Apigee-Okta integration. This is the authorization bearer token.
6	OKTA_OPEN_ID_CONNECT_ATTR	clientId	Yes	This KVM looks like it is not finished by the developer.
7		clientSecret	Yes	
8		AUTH_SERVER_ID_DEFAULT	Yes	
9	SPIKE_ARREST_RATE	RESOURCE_OWNER_PASSWORD_GRANT	Yes	The values from this KVM are the reference rate to the policy Spike Arrest, for each endpoint is a different value
10		CLIENT_CREDENTIALS_GRANT	Yes	
11		REVOKE_GRANT	Yes	

12		REFRESH_GRANT	Yes	
13		PASSWORD_CHANGE	Yes	
14	SPLUNK_TOKEN	AUTHORIZATION_TOKEN	Yes	The values from this KVM are to the integration between Apigee and Splunk
15		SPLUNK_URL	Yes	
16	IP_WHITELIST_CONFIG	HeaderToCheck	Yes	The values from this KVM are necessary to validate the IP range and to the validate the IP restrictions.
17		ExtraIPsToCheck	Yes	
18	TOKEN_EXPIRY_TIME	ACCESS_TOKEN_EXPIRY_MILLIS	Yes	The values from this KVM are necessary to the expiration time for the access token and refresh token.
19		ACCESS_TOKEN_EXPIRY_SEC	Yes	
20		REFRESH_TOKEN_EXPIRY_MILLIS	Yes	

TLS Key Store

Not applicable/No target Server

Virtual Host

- secure

Cache

	Resource	Prefix	Key Fragment	Details
1	JWT_Cache	Oauth	JWT	This cache keep the Fat JWT token, when the endpoint '**Show Cached JWT**' is called we retrieve the FAT Token using the JTI from the access token and the API Product from the query param.
2			Apigee API Product	
3	OKTA_CLAIMS_CACHE	Oauth	OKTACLAIMS	This cache keep the Okta claims
4	JWT_Cache	UserType	JTI	This cache keep the user type from Okta profile
5	JWE_Cache	Oauth	JWE	This cache keep the JWE, IF the custom attribute '**Cache_Encrypted_Payload**' from API Product, is equals YES.
6			Okta User	
7	JWT_Cache	OktaOauth	Okta Access Token	This cache keep the Okta **access** token
8	JWT_Cache	OktaOauth	Okta Refresh Token	This cache keep the Okta **refresh** token

Review Comments

Following are the answers to the questions raised during the B2B Okta Integration discussion on 08-Jan-2024

1) What's the solution on Time bound cache storage - Dheeraj/Cleison

Brillio:: There is no need for 'time bound cache storage' as a separate feature. AWS Cognito takes care of it, ie. token is cached till its valid. AWS API gateway validates token validity from AWS Cognito.

2) Details on how does internal users access APIs

Brillio:: Internal user are also part of OKTA and can follow the same process of as that of the external user. ie. Either from developer portal or via B2B flow. The internal user needs to be onboarded or be part of the Okta node for the respective environment that they want to access.

3) what components are invoked in what sequence and how is data stored (ARB point of view)

Brillio:: *Pending*

4) B2B and B2C: what is the segregation point between the 2 on the design and implementation (how does the traffic flow between for B2C and B2B)

Brillio:: B2B and B2C

- will have separate application in OKTA and
- have separate AWS cognito user pool.
- use different grant flow type
- will have a unique Client_id/client_secret
- will have separate app clients that are associated with different AWS cognito user pool.
- will have different forms of Token generation url . In B2B, a custom token generator is being invoked whereas, in B2C Cognito authorization_code grant_type is being used

5) B2B: How is the lambda function integrated with OKTA?

Brillio:: Lambda function uses OKTA endpoint URL and sends back the validation request to OKTA , with password grant flow along with OKTA username and password provided by user in request header and OKTA Client_id/client_secret already stored in Lambda function. This will create an access token, that validates if the username is part of that OKTA application.

6) JTI is important. Explain how it is being used

Brillio:: JTI information is available when we are generating token. eg.

```
{
"sub": "7ub4lfkif73rqgd8pgv9p566i1",
"token_use": "access",
"scope": "b2b/read",
"auth_time": 1704811268,
"iss": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_kDHNiT8cR",
"exp": 1704814868,
"iat": 1704811268,
"version": 2,
"jti": "36b8df7a-8b33-4938-89e5-3f1ffbfad46b",
"client_id": "7ub4lfkif73rqgd8pgv9p566i1"
}
```

7) To prevent routing to Okta, where is the payload information stored? Is it on DynamoDB? What are the claims in JWT (refer thin JWT that is generated in APIGEE)

Brillio:: In AWS solution, thin and FAT JWT token are not generated. JWT token are generated from AWS Cognito and Cognito caches the JWT token until its validity. Every time a request reaches to Enterprise API gateway, it is get validated with Cognito authorizer for scope and validity. Hence in the AWS hence in the AWS solution , there is no need of Dynamo DB for storing payload information.

Summarizing

B2B :

1.User profile is created in OKTA and app client is created in AWS cognito user pool.

2.Curl the token URL with password grant flow. Along with OKTA username password and AWS cognito App client ID and secret.

3. Token URL is invoking AWS API gateway , that API gateway is integrated with custom token lambda function.

4. Lambda function will validate user from OKTA with the help of OKTA token endpoint.

5. After that lambda validates the user in AWS cognito user pool , if user exist then generate access token.

6. If user does not exist in AWS cognito user pool , then create user in cognito and then generate JWT access token.

7. Once enterprise got the access token , same can be pass along with AWS API URL and access.

8. Once enterprise sends the request, AWS APi gateway validates the token from AWS cognito and they forward the request to microgateway.

B2C:

1) User create a App in developer portal, in the backend an Application client is created in AWS Cognito user pool.
2) Client ID and Secret is shared with user .
3) When Clicked on API docs there will be 2 buttons available . First one is to generate authorization_code and another one is for authorization.
4) When clicked on authorization_code button , it will redirect to AWS cognito hosted UI and then generate the authorization code .
5) Now click on authorization button, where grant type , token URL and redirect URI will be static , and user need to fill authorization code and client ID and secret from drop down. Once done and send. It will generate access token and then using that access token create a session for that product.