CX Cloud Authentication and Authorization Model

v0.3.0

David Wang, Joshua Dotson, Erik Burgess, Tony Porterfield

Build Date 2020-05-04

Table of Contents

1 Changelog

- **v0.3.0** (josdotso@cisco.com)
  - Update claims JSON example, add authz approach section.
- **v0.2.0** (josdotso@cisco.com)
  - Add standard product and account role sketches for authz.
- **v0.1.3** (josdotso@cisco.com)
  - Integrate previous docs:
    * https://hackmd.platform.ksng.io/hd3I8RCAQqiCBmqfUfwgzw#
    * https://confluence-eng-sjc1.cisco.com/conf/display/CXCLOUD/CX+Cloud+IDP+Authentication+Flow
- **v0.1.2** (josdotso@cisco.com)
  - Add content from "CX Cloud - Authentication and Authorization Requirements"
- **v0.1.1** (josdotso@cisco.com)

  - Add component details as taken from Rally user stories.
- **v0.1.0** (josdotso@cisco.com)
  - Initial version.

2 Overview

CX Cloud is a suite of cloud services that support Cisco customers in new and exciting ways (e.g. CX Customer Portal). Each service in CX Cloud (e.g. CX Cloud Data) in the portfolio exposes useful resources to its users.

CX Cloud APIs employ carefully selected patterns for authentication and authorization. These patterns closely align with industry best practices while ensuring high levels of security, auditability, monitoring and availability.

This document describes how Authentication and Authorization will be evaluated by user and machine accounts within CX Cloud services.

2.1 Authentication (aka. authN)

CX Cloud's authentication model (aka. authN mode) uses OpenID Connect and JSON Web Tokens (JWT).

2.2 Authorization (aka. authZ)

CX Cloud's authorization model (aka. authZ model) is an Role Based Access Control (RBAC) authorization model that defines which permissions are afforded to which user roles.

3 Background

3.1 OAuth2 and OpenID Connect (OIDC)

- MUST watch:
  - OAuth 2.0 and OpenID Connect (in plain English)

4 Requirements

This section covers implied requirements for both authentication AND authorization in CX Cloud.

4.1 Definitions

- Authentication (AuthN) - Proving identity
- Authorization (AuthZ) - Access control

4.2 Requirements

4.2.1 Standards-Compliance

- 100% commonly adopted, open, modern standards must be used for end-to-end authentication and authorization (OAUTH2/OIDC/JWT)
- Ensure an ecosystem of standards-compliant software may easily integrate with and utilize CX Cloud without custom development.
- Ensure CX Cloud Identity Provider (IdP) may easily federate with 3rd party IdP's without custom development (and vice-versa).

4.2.2 Security

- The CX Cloud IdP must not directly store human credentials or information.
- Cisco CCO user credentials must not pass through the CX Cloud IdP.
- Cisco CCO user credentials must be presented to the Cisco IdP directly.
- Credential and authorization data must not be pushed from Cisco into the Cloud. Only the minimal information should be pulled when needed, as needed.
- Security principals whether human users, machines, or third-party application must share the same authorization model (scopes, roles, permissions, formats, etc.). Applying a common authorization model to all security principals regardless of type reduces the attack surface, testing, and bugs.
- Services in a particular region must be able to reject a principal's access token based on geolocation restrictions (GDPR compliance).
- Security principal access_tokens must be rotated at minimum on an hourly basis. Rotation must not require the user to present their original credentials.

- Security principal access_tokens must support minimal resource scoping and minimal permissions. If a user is a super-admin, that user must be able to request tokens customized for a limited subset of resources.

4.2.3 Scalability

- Callbacks to Cisco AuthN/AuthZ databases (SAVA, EB, Cisco IdP) must be minimized to avoid bottlenecks, avoid single points of failures, and reduce data-in-transit attack surface.
- Callbacks to Cisco IdP, Smart Accounts Virtual Accounts (SAVA), and Entitlements Base (EB) must be minimized regardless of caching workarounds.
- AuthN/AuthZ databases must not be contacted with every API call, regardless of caching workarounds. These services may be called once upon security principal authentication, in order to obtain an access_token (not upon every call to API services).
- Validation of access_tokens and enforcement of access control must be constant scale. The AuthN/AuthZ system should not require additional resources to handle one user's access_token versus a million users' access_tokens. The distributed authentication and distributed authorization system should everything it needs within the access_token without reaching outbound to AuthN/AuthZ databases.

4.2.4 High Availability and Global Seamless Failover

- User redirection to failover regions must be seamless and not require any users to re-authenticate. User access_tokens are valid across the global CX Cloud deployment.
- If a CX Cloud region loses connectivity back to Cisco, the currently valid access_tokens presented to the API servers in the region must continue to be honored until token expiration. AuthZ and AuthN must continue to be enforced and API requests must continue to be serviced, even when Cisco is unreachable.
- Any receiver of an access_token must be able to independently check the authenticity and validity of the token without calling back to Cisco AuthN databases
- Any receiver of an access_token must be able to identify the authorizations represented by the token without calling back to Cisco AuthZ databases

4.2.5 Customer Experience

- Must minimize system latencies by removing unnecessary callbacks to central Cisco AuthN/AuthZ DBs (e.g. for every API call).

- If CX Cloud IdP goes offline, all currently valid, non-expired access_tokens must continue to be honored.
- If a customer is redirected to a new region during an HA event, the customer must not be required to "re-login" upon the seamless redirection. The customer access_token must be valid at any failover region.

5 Components

5.1 Access Token (JWT)

If a user is NOT supposed to have access to CX Cloud, then `claims["https://cx.cisco.com"]` return as empty with an error message.

Access tokens will contain RBAC claims that are stubbed (not from SAVA or EB) for authorization to CXC API servers deployed as different personalities.

5.1.1 About JSON Web Tokens (JWT)

https://jwt.io/introduction/

JWT tokens are a base64 encoded string made of 3 parts:

- Header
  - Describes the crypto signature algorithm
- Claims
  - A list of key-value pairs (values may be arrays)
  - For example
    - user_id: mcboats
    - account_id: supermart
    - groups: ["producer-api-rw", "consumer-api-ro"]
    - roles: ["admin", "owner"]
- Signature
  - A crypto-verifiable signature on the validity of the JWT Token
  - Can be used to verify that an IdP created this token and the token was not modified in any way.

5.1.2 JWT Tokens and Distributed Authentication

Traditional authentication and authorization mechanisms require each service to call back to a central authentication and authorization database to check permissions before servicing each request.

JWT Tokens allow for distributed authentication. Anyone with a JWT token may download the originating Identity Provider's public key from a well-known internet location in order to verify the JWT signature. If the JWT signature validates, then authorization claims within the JWT may be trusted.

Both User accounts and Machine accounts will receive access_tokens in the form of JWT. The access_tokens will contain the same set of JWT Claims which designate authorizations.

As requests cascade through the CX Data System, the JWT may be passed and reverified, and assessed for authorization, at each software component. Initially, this will be done only at the edges of the system (e.g. API servers), but eventually will be implmented at every level.

5.1.3 JWT Tokens and Distributed Authorization

After a JWT token is cryptographically verified, the embedded Claims may be trusted. If the claims include authorization groups and roles, then these claims may be used for Authorization.

Software applications should use Open Policy Agent to implement the mapping of claims to authorization policies.

- https://www.openpolicyagent.org/docs/latest/

5.2 CX Cloud AuthZ API

aka. Auth Claims Resolver

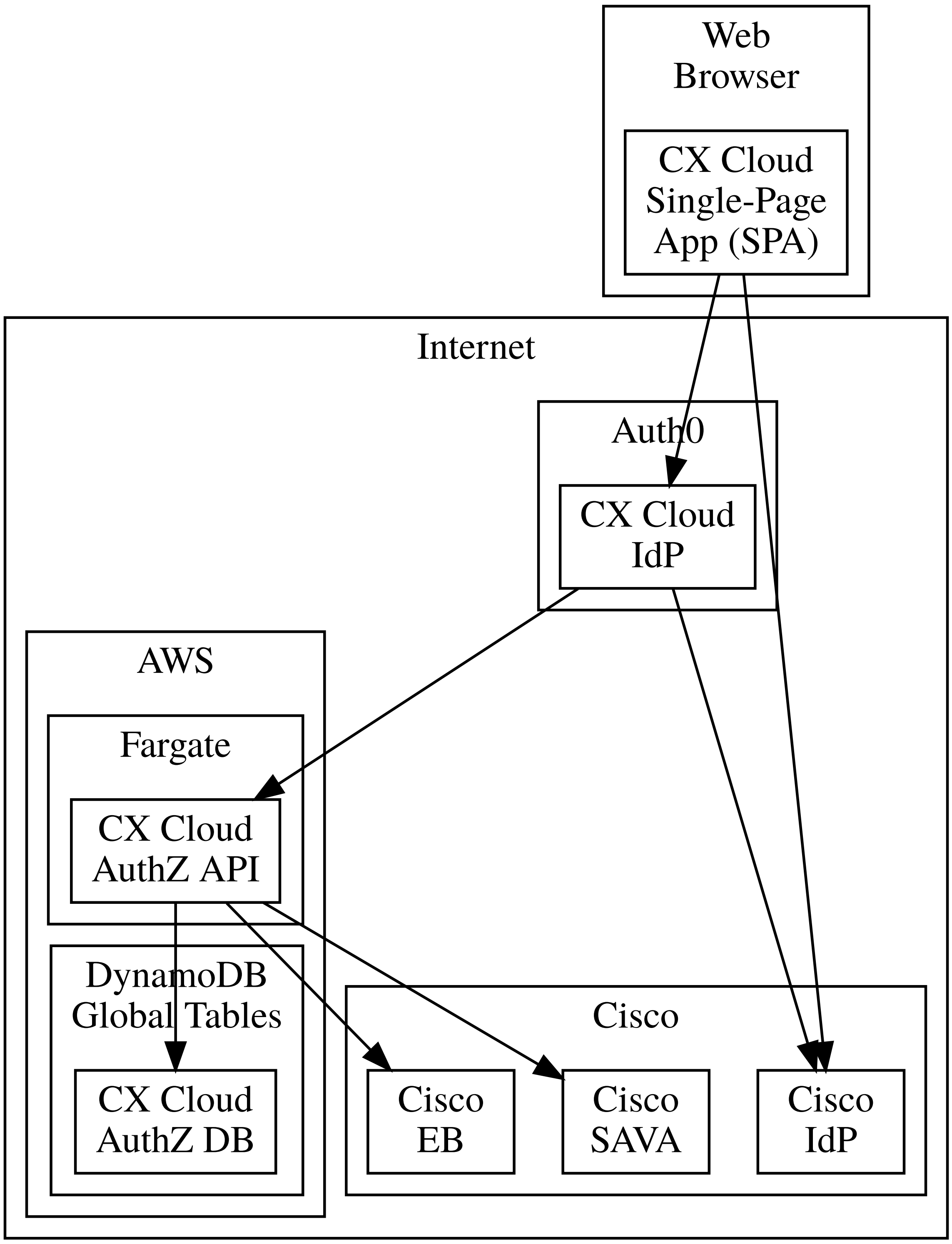


Figure 1: CX Cloud AuthZ API

The preliminary Authorization Model that we used in our JWT tokens was temporary. It is purely permissions based.

The IdP needs to call the Auth-Claims-Resolver in order to resolve a CCO username to a set of CX Cloud Authorization Claims. The IdP merges the resulting claims into the JWT access_token that is generated for a particular user.

ACR is a single API endpoint. Given a CCO username, return a structure to be embedded in JWT token claims.

```
"https://cx.cisco.com/v1": {  // NOTE the "v1"
  "accounts": {
    "global": {
      "cisco": {
        "roles": [
          "CXCloudDataAdmin",       // Use case: Product Administrator
          "CXCloudDataReader",      // Use case: Product Monitor
          "CXCloudDataUploader",    // Use case: IBES Administrator
          "CXCloudDataWriter",      // Use case: Product Support
        ]
      },
      "machine": {
        "roles": [
          "CXCloudDataUploader",    // Use case: IBES Uploader
        ]
      }
    },
    "account_scoped": {
      "cisco": {
        "<SmartAccountID1>": {
          "roles": [
            "CXCloudDataAdmin",     // Use case: Product Support
            "CXCloudDataReader",    // Use case: Product Support
            "CXCloudDataUploader", // Use case: Product Support
            "CXCloudDataWriter",    // Use case: Product Support
          ]
        }
      },
      "machine": {
        "<SmartAccountID1>": {
          "roles": [
            "CXCloudDataUploader", // Use case: Collector Uploader
          ]
        }
      }
      "smart": {
        "<SmartAccountID2>": {
          "type": "sa|va", // MAY need this
          "maxCxLevel": 1,  // users service access level
          "roles": [
            "AccountAdmin",            // Use Case: Account Administrator
            "AccountFullUser",         // Use Case: Account Writer
            "VirtualAccountAdmin",     // Use Case: Account Administrator
            "VirtualAccountFullUser"   // Use Case: Account Writer
          ]
        }
      }
    }
  },
  "sub": "cx-data"
}
```

Account IDs and types are taken from SAVA. Permissions resolve from the SAVA user role. If for any reason the user should not have access to CX Cloud, an error message must be returned. The API call should always return 200 OK, but the contents must change.

TODO: Modify the error format for the long term. This was temporary.

```
"https://cx.cisco.com": {
  error: { // presence of this key means error.
    code: <error_code>,
    description: "<human_readable_reason>",
  },
}
```

5.3 CX Cloud IdP

CX Cloud Authorization claims are embedded in a user JWT access_token.

The CX Cloud IdP issues OAuth2-compatible tokens for human and machine users.

5.3.1 CX Cloud IdP Hooks

Secret stored in appropriate place in IdP (most have write-only secrets stores)

5.4 CXC APIs

For each API request, the CXC deployments enforce the access_token authorization RBAC claims that relate to their respective resources.

5.5 EB

TODO

5.6 OAuth2 / OIDC

- https://en.wikipedia.org/wiki/OAuth#OAuth_2.0
- https://en.wikipedia.org/wiki/OpenID_Connect

5.6.1 OAuth2 Secrets Rotation

As a user or machine account, obtaining an OAuth2 access_token is necessary for presenting to resource servers (e.g. APIs) for authentication and access control. The access_token will be checked by the service to determine what requests are authorized to be serviced.

Obtaining an access_token initially require client secrets. For users, this may be a login/password/2fa. For machine accounts, this may be a client_id/client_secret.

Access_tokens are typically obtained along with a "refresh_token". Access_tokens expire on a preconfigured period (e.g. 30mins, 1hour) and may be rotated by presenting a refresh_token to the IdP, which will then provide a new access_token and refresh_token.

This allows access_tokens to be rotated without transmitting the original, sensitive client credentials over any network.

5.6.2 OAuth2 Secrets Revocation

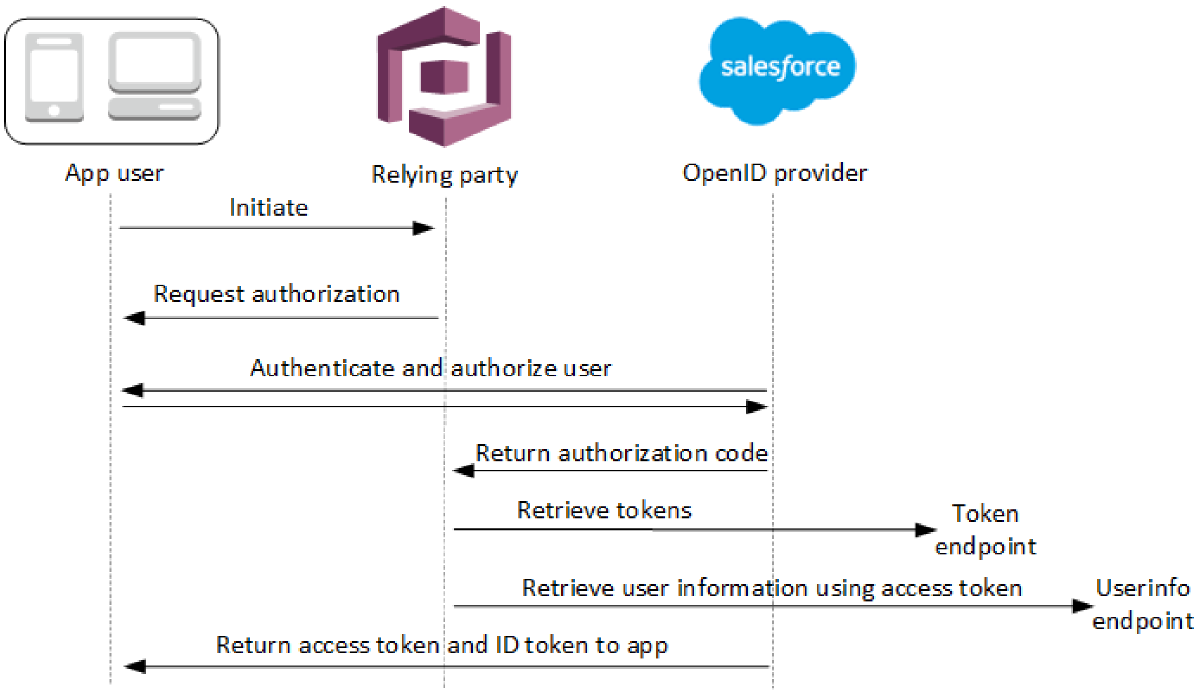
Access Tokens typically have a short expirey (minutes to hours), because they are valid for the duration of the encoded lifetime.

Revoking access_tokens may involve deleting the refresh_tokens for a particular user, and then waiting for existing access_tokens to expire.

Faster Revocation is possible, by having all entities check the IdP for a revocation list. If we add this complexity, it should be best effort, since this affects the distributed nature of the authentication system and creates a single point of failure and bottleneck.

5.6.3 OIDC OAuth2 Authentication for Users

In the below picture from the aws documentation, Cognito is the Relying party with Cisco IDP is the OpenID Provider.


salesforce
App user
Relying party
OpenID provider
Initiate
Request authorization
Authenticate and authorize user
Return authorization code
Retrieve tokens
Token endpoint
Retrieve user information using access token
Userinfo endpoint
Return access token and ID token to app

TODO: Update ^ section to reflect change to Auth0 from Cognito.

Users (i.e. human users) must follow an OpenID Connect flow in order to obtain access_tokens, which then are presented with every API call.

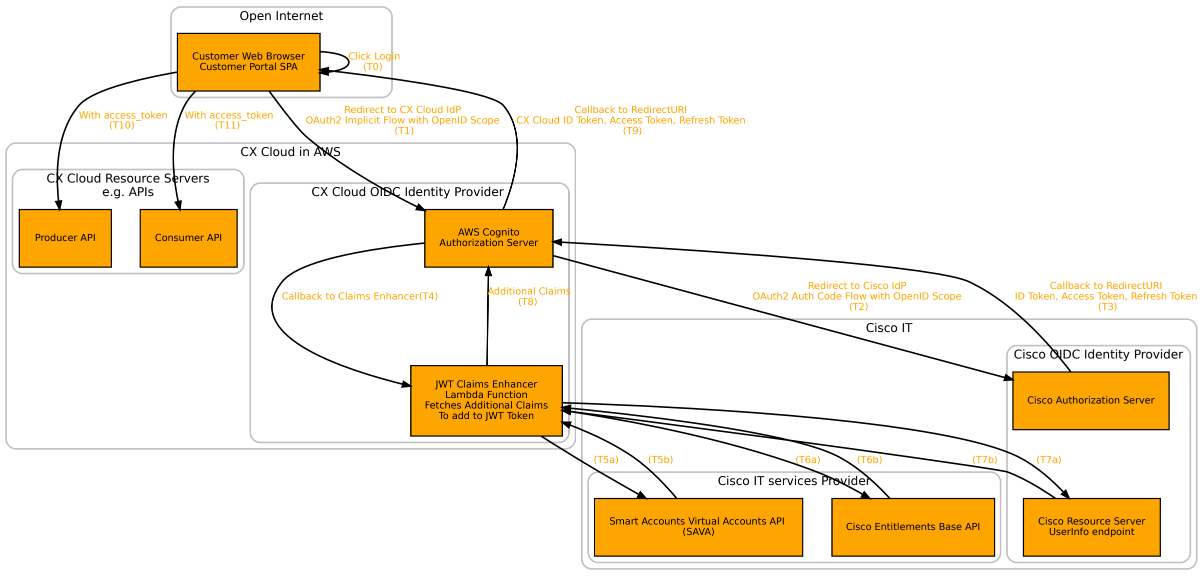


Figure 2: OIDC OAuth2 Auth Flow

T0

- The user has already downloaded the Customer Portal Single Page Application (SPA).
- The user is presented with a "Login" button, because the user's JWT access_token is not present.

T1

- The user clicks "Login" and is HTTP directed to the CX Cloud IdP, which is an OIDC-compliant Identity Provider on the public internet.
- This begins the OIDC "implicit flow" grant type, which eventually results in the CX Cloud IdP returning an id_token, access_token, and refresh_token that is applicable for the CX Cloud system, but not just yet.
  - Request Params: `&scope=openid email profile &response_type=token id_token`
  - Implicit flow is used instead of authorization code flow because the browser itself is the OAuth2 client, which does not have a more secure backend channel. Thus, authorization code exchange on the frontend channel does not increase security.
- Before the CX Cloud IdP can return the id_token, access_token, and refresh_token for the user in the context of CX Cloud, the CX Cloud IdP will require that the user presents proof of who they are, in the form of an id_token from Cisco IdP.

- The CX Cloud IdP is "connected" to the Cisco IdP. This means that the CX Cloud IdP itself is a registered OAuth2 client of the Cisco IdP, which is also an OIDC-compliant Identity Provider on the public internet.
- If the user confirms at the Cisco IdP that the CX Cloud IdP is trusted to access specific resources as defined by OAuth2 scopes, then the CX Cloud IdP will receive "Delegated Authorization" for specific resources and be able to call Cisco APIs on behalf of the user, given a valid access_token.

T2

- In order to obtain proof of authentication, the CX Cloud IdP forwards the user to the Cisco IdP so that the user may authenticate with Cisco IdP.
- This begins the OIDC "authorization code flow" grant type.
  - Request Params: `&scope=openid email profile &response_type=code id_token`
- When the user reaches the Cisco IdP, the user presents their Cisco login credentials and authenticates with Cisco Duo and 2FA.
- After successful authentication, the Cisco IdP authoriation server presents the user with a prompt to approve the CX Cloud IdP as an authorized client for the requested scopes which allow the client to access information from Cisco Resource Servers (APIs, and well-known OIDC userinfo endpoint) on the user's behalf.

T3

- The user approves, and the Cisco IdP redirects the user back to the CX Cloud IdP with an authorization_code and id_token.
- The CX Cloud IdP authorization server, which is AWS Cognito, receives the authorization_code and id_token.

T4

- The CX Cloud IdP authorization server calls the JWT Claims Enhancer and passes the authorization_code and id_token.
- The JWT Claims Enhancer exchanges the authorization_code at the Cisco IdP for an access_token and refresh_token. Now, it has an access_token, refresh_token, and an id_token for the user.
- All of the JWT tokens are validated by the public key provided by the Cisco IDP, fetched from the well-known OIDC JWKS location.

T5a + T5b

- The JWT Claims Enhancer makes queries to the **Smart Accounts Virtual Accounts APIs**. It gathers authorization information such as groups, settings, etc, and returns them as key value pairs that will become claims in the CX Cloud JWT access token which is returned to the client (i.e. Customer Web Browser).

T6a + T6b

- The JWT Claims Enhancer makes queries to the **Entitlements Base APIs**. It gathers authorization information such as groups, settings, etc, and returns them as key value pairs that will become claims in the CX Cloud JWT access token which is returned to the client (i.e. Customer Web Browser).

T7a + T7b

- The JWT Claims Enhancer makes queries to the **Cisco IdP Resource Server UserInfo Endpoint**. It gathers authorization information such as groups, settings, etc, and returns them as key value pairs that will become claims in the CX Cloud JWT access token which is returned to the client (i.e. Customer Web Browser).

T8

- The JWT Claims Enhancer has now gathered all additional claims. It now returns the list of claims to the AWS Cognito Authorization server that will embed the claims in a newly formed access_token, to be returned to the user.
- The access_token claims will include authorization groups and privilege scopes. For example, this access_token is allowed read-only access to the producer API for a specific customer identifier.

T9

- The CX Cloud Authorization Server finally returns to the Customer Web Browser Customer Portal SPA client an id_token, access_token, and refresh_token.

T10 + T11

- The CX Portal SPA running in the browser may now directly call Any CX Cloud Resource Servers. All calls to API serverw would include the JWT access_token as an HTTP Bearer Token. The API servers will validate the access_token signature, and inspect the claims in order to determine authorization privileges.

5.6.4 Non-OIDC OAuth2 Authentication for Machine Accounts

Machine Accounts will obtain an OAuth2 access_token and refresh_token directly from the CX Cloud IdP. Each machine account will be pre-registered as an OAuth2 Client in the CX Cloud IdP. Registering an OAuth2 Client means that an entity receives a unique client_id and client_secret.

A machine account will initially need to step through the OAuth2 "Client Credentials" Grant Type authentication flow. In order to obtain an access_token and refresh_token, the machine account will present its pre-registered client_id and client_secret to the IdP, and the IdP will respond with an access_token and refresh_token.

Every call that a machine account makes to an API will also include the access_token as a Bearer token, which will be used for authentication and authorization.

A wrapper service may be created to faciliate the provisioning of machine accounts.

https://auth0.com/blog/using-m2m-authorization/

5.7 Open Policy Agent (OPA)

Open Policy Agent is used to embed in the CXC API a system of evaluating access token claims for the permissions afforded to that token.

5.8 SAVA

TODO

6 AuthZ Approach

6.1 Use Cases

- User logs in using UI
    - UI uses a client id of type "Single Page Application"
    - Customer logs in and role data is fetched from SAVA/EB
    - Cisco employee logs in has a role for a specific customer
        - TODO - currently not supported Jaswant to create requirements for EB
    - Cisco employee logs in and has a role based on an ADAM group
    - 3rd-party app developer logs in and has access to data which customers have shared via oauth2 scopes and grants

- Machine users
  - Customer registers a machine user
    * machine account gets assigned a role with privileges no greater than the users current role
  - Cisco “software account” (universal – a token with access to all customer data for a specific API)
    * IBES upload-er needs a role to upload anything and not be tied to a customer
  - 3rd-party applications
    * partner develops an app which is granted access to a customers data

6.2 API Types and Scope

API scopes and types is something not required at this time for simplicity. All APIs producer and cosumer will be publicaly avaiable and protected by RBAC.

6.3 Types

Edge: - APIs which are deployed and available externally Intermittent: - APIs used internally for data processing i.e. the producer API for inserting uploaded device data

6.4 Scope

- Public: APIs, which a Cisco customer has access to based on role level wither thought he UI or machine account
- Private: APIs built for Cisco needed to specific data ingestion and processing use cases

API	Type	Scope	Method (HTTP request type)
Presigned-url	Edge	Private	POST
Producer	Intermittent	Private	POST
Consumer (network elements)	Edge	Public	GET
Auth claims	Edge	Private	GET

6.5 Cx Level

The scope contains a max cxClevel field whcih the UI uses to determine what is displayed to the user.

Machine accounnts do not require max CX level as authorization is done based on the role at the api level similar to UI apu calls.

The authorization model does NOT look at authorization at the device level.

6.6 Account Level Authorization

The security principals of a typical customer include human users, machine users, and 3rd party applications. All customer access is Smart Accounts Virtual Accounts (SAVA) account-scoped. Any given role applies at most to one customer account ID.

We will not attempt to map the data in Smart Accounts Virtual Accounts (SAVA) into a "common" authorization model. Instead, SAVA authorization data will be directly embedded into the JWT token as a specific type of supported authorization model, namespaced as a key under the JWT claims (e.g. "smart"). SAVA is only one type of authorization type supported. SAVA will be used for account-scoped authorization.

6.6.1 Product Level Authorization (i.e Global)

Another authorization model that will be supported for product We will support multiple types of accounts. SAVA is one type.

Everything in SAVA/EB is scoped to a virtual account. There is no "global"

6.7 Limited Backend Details in Access Token

- Access Token MUST never mention individual gRPC Services (e.g Producer API's
- `NetworkElement` gRPC Service). Access Token MAY mention API names (e.g.
- Producer API).

6.8 Desired AuthZ UX

- Settings page in SPA, allowing change to different account and role, follow
- go2/aws pattern of selecting account/role.
- Option 1: Auth twice and receive two small tokens (SPA, API)

    1) Auth1: Get small token to list possible account/roles combos. 2) Call to claims resolver API to get the list. 3) Auth2: Get small token for specific account/role combo. 4) Use the API

- Option 2: Auth once and receive larger token (PREFERRED)

    1) Auth1: Get large token with all accounts/roles embedded. 2) Use the API.

6.9 Size and Transmission of Access Tokens

Usage of GRPC does not retransmit the token repeatedly (token is set once in the metadata of the channel). OpenAPI protocol will require retransmiting token upon on every request.

6.9.1 Large Access Tokens

(PREFERRED OPTION)

- Much less complexity compared with small token Less load on Auth systems.
- Less code to be written. Reduce the number of calls needed to use API. If
- we adopt Large Tokens, Account ID must be mentioned in every RPC. This is
- because we cannot infer from a Large Token which Account ID and Role the user
- wants to use at RPC time.

TODO: Fill in max expected number of accounts per user from Jaswant or similar.

6.9.2 Small Access Tokens

- Improve performance of OpenAPI calls. Avoid needing to raise the HTTP
- request size at web server.

7 Machine accounts

Users create a machine account either with an API or with a UI.

Machine account consists of a client id and a secret. Once a machine account is created and returned to the user, the user can never retrieve the secret again. New accounts may be generated and existing accounts may be revoked / deleted.

The generated machine account receives the same role as that of the user who creates it.

7.1 Registration

- A user will first obtain an access token by the logging into the UI (using a human user flow) or using any other mechanism that gets an access token with a ccoid and role information.
- The token obtained above will be presented to the machine accounts API
- The API will validate this token for presence of a role that must have permissions necessary to allow a machine account create action
- If validated, the API will create a non-interactive(machine) client application using Auth0 management APIs, store some information about this newly created machine account in a database and return the client id and client secret to the user.

7.2 Machine account create request

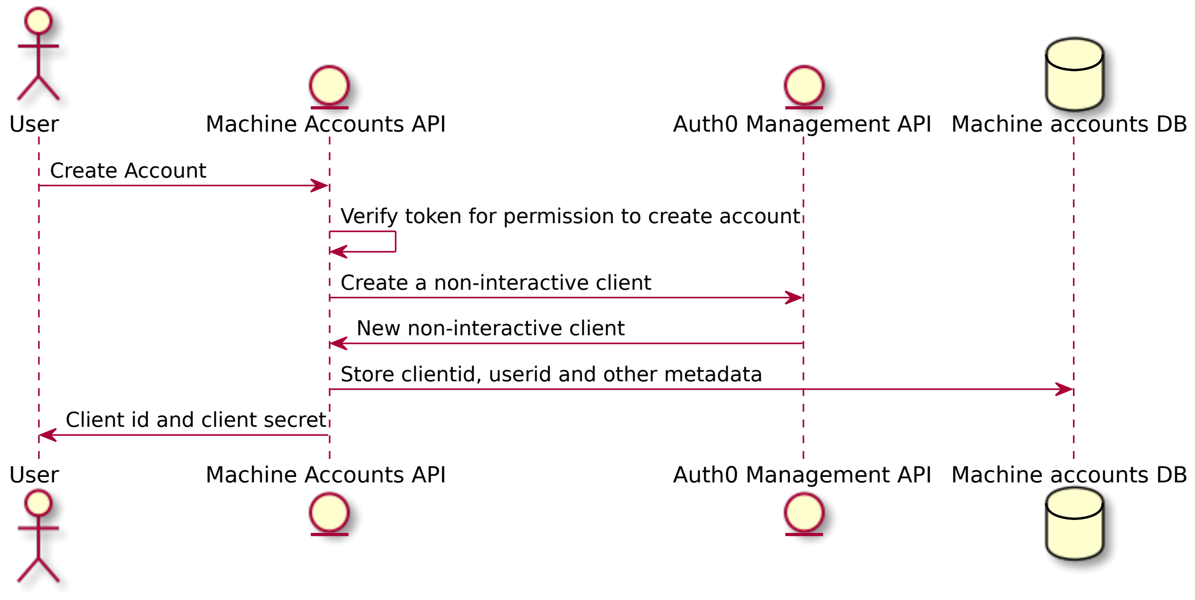

User
Machine Accounts API
Auth0 Management API
Machine accounts DB
Create Account
Verify token for permission to create account
Create a non-interactive client
New non-interactive client
Store clientid, userid and other metadata
Client id and client secret
User
Machine Accounts API
Auth0 Management API
Machine accounts DB

Figure 3: Machine account create request

NOTE: The client secret will not be stored in the database. It is returned from the API directly to the user at the time of creation. Client secret can never be queried again after creation.

7.3 Machine account token request

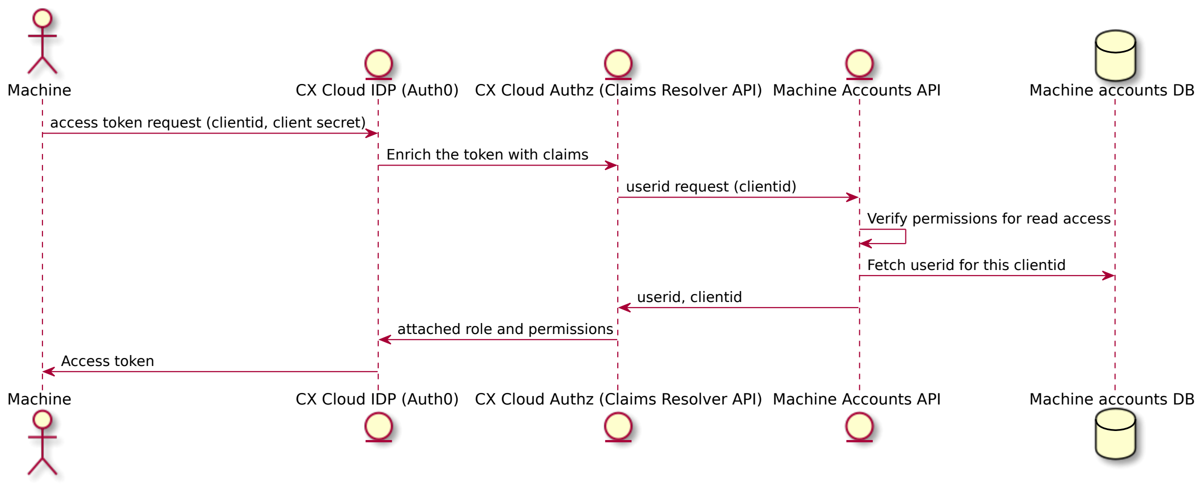

Machine
CX Cloud IDP (Auth0)
CX Cloud Authz (Claims Resolver API)
Machine Accounts API
Machine accounts DB
access token request (clientid, client secret)
Enrich the token with claims
userid request (clientid)
Verify permissions for read access
Fetch userid for this clientid
userid, clientid
attached role and permissions
Access token

Figure 4: Machine account token request

7.4 API

Action	JSON/HTTP	Returns
Create a machine account	POST /machine-accounts/v1/	201 CREATED Just created machine account with client_secret
Get the machine account	GET /machine-accounts/v1/:id	200 OK return the machine account with the creator info (will not return the client secret)
Get all machine accounts created by this user	GET /machine-accounts/v1/	200 OK List of machine accounts (will not return the client secret)
Delete a machine account	DELETE /machine-accounts/v1/:id	200 OK (should just change the status to DELETED and update the deleted_by and deleted_at columns)

7.5 Relational data model

Basic information about the machine account needs to be stored in a db.

7.5.1 machine

Column	Description
id	uuid (auto-generated)
auth0_name	short but descriptive name
auth0_client_id	auth0 client id
auth0_token_url	auth0 tenant based url to request access tokens (may not be stored in the db)
status	ACTIVE, DELETED
created_by	cco userid (taken from the access token used to make the request)
created_at	timestamp at creation (auto-generated)
deleted_by	cco userid
deleted_at	timestamp at deletion

7.6 Auth0

Auth0 exposes a management API [0] as part of registering a machine app the API will have to call Auth0 [1] create client to generate the client id and secret in auth0.

Users call the auth0 authentication api when using a client id and secret for accessing APIs. See sample script [2] to get an access token.

0. https://auth0.com/docs/api/management/v2
1. https://auth0.com/docs/api/management/v2#!/Clients/post_clients
2. https://www-github3.cisco.com/cxe/cxe-throwaways/blob/master/auth0/bin/get_machine_token.sh

7.7 Discussion

- **Question: [kamjoshi]** Can we avoid storing the machine role in the machine accounts db if we use the following approach.

  - Use the Auth0 hook to query the cx-cloud-authz(auth claims resolver) by passing in the client id
  - The cx-cloud-authz can query the machine accounts api to fetch the user info and attach the appropriate machine role
  - This will help keep the machine account role in sync with the user role at all times.

  Resolution: In a discussion on April 8, 2020 between Erik, Kamal and Mike, it was decided that the machine accounts API will not store roles. Cx-Cloud-Authz (Claims resolver) will simply query the machine accounts api for the creator info passing in a client id and use that information to attach roles to the access token requested by a machine account.

- **Question: [kamjoshi]** Should the machine accounts API allow a user to specify the role on the machine account that is more restricted than his/her own role?
  TODO: Currently this is being avoided as the role hierarchies are not clear. We could simplify by stating Admin > All > Writer > Reader (product-wise or account-wise) but this needs more thought and consideration.

- **Question: [erburges]** Should machine accounts be scoped at the company level i.e. any company admin can see the generated machine accounts?
  [kamjoshi] I think a machine account access token can have exactly the same structure and semantics as that for a human user account. So, it can be scoped at any or all of company, product or account levels.

- **Question: [kamjoshi]** What is the strategy to clean up the deleted accounts after the audit period has passed?
  TBD:

8 Diagrams

8.1 Auth Sequence #1

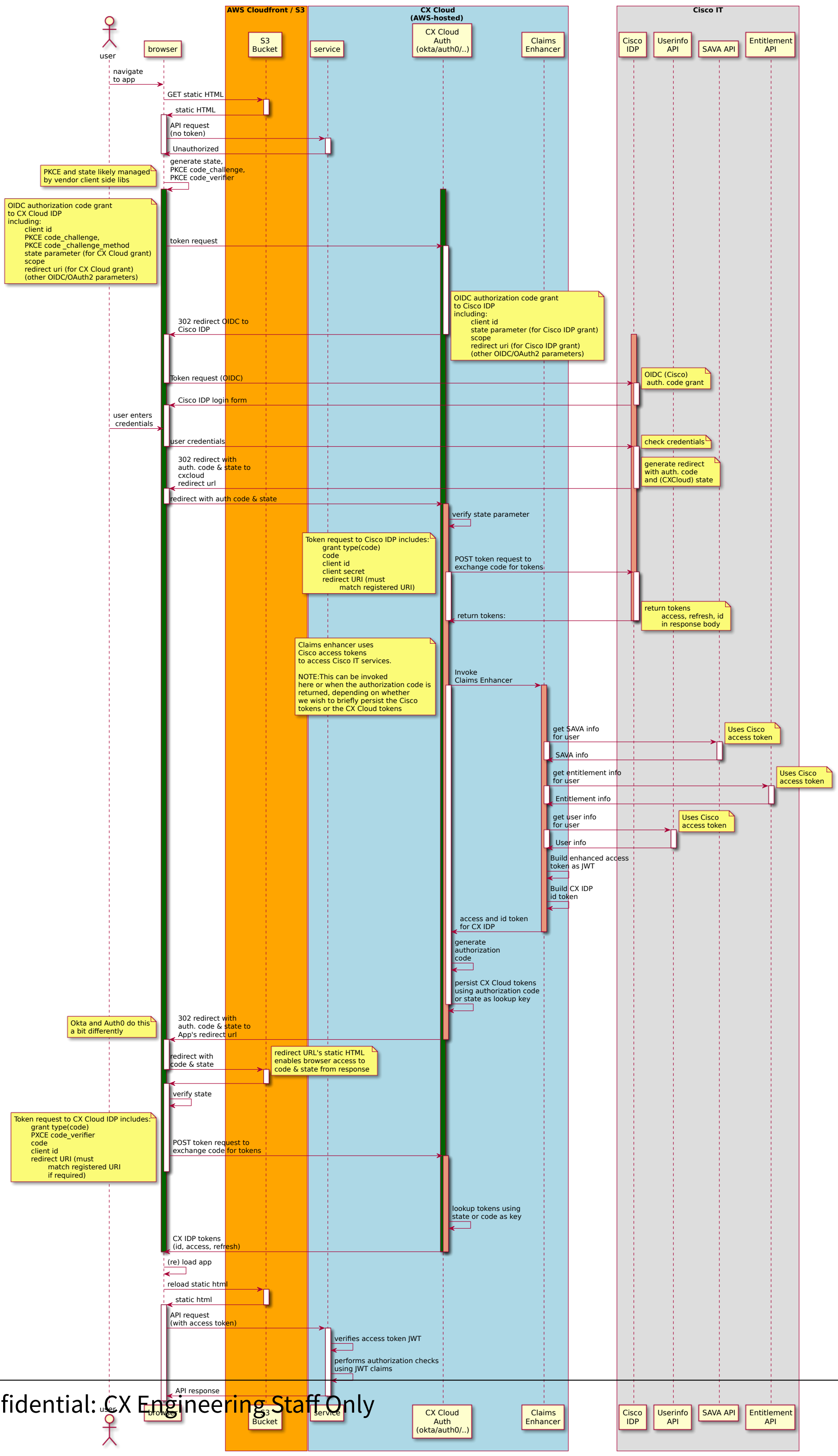

AWS Cloudfront / S3
CX Cloud (AWS-hosted)
Cisco IT
user
browser
S3 Bucket
service
CX Cloud Auth (okta/auth0/..)
Claims Enhancer
Cisco IDP
Userinfo API
SAVA API
Entitlement API
navigate to app
GET static HTML
static HTML
API request (no token)
Unauthorized
generate state, PKCE code_challenge, PKCE code_verifier
PKCE and state likely managed by vendor client side libs
OIDC authorization code grant to CX Cloud IDP including: client id, PKCE code_challenge, PKCE code_challenge_method, state parameter (for CX Cloud grant), scope, redirect uri (for CX Cloud grant), (other OIDC/OAuth2 parameters)
token request
OIDC authorization code grant to Cisco IDP including: client id, state parameter (for Cisco IDP grant), scope, redirect uri (for Cisco IDP grant), (other OIDC/OAuth2 parameters)
302 redirect OIDC to Cisco IDP
Token request (OIDC)
OIDC (Cisco) auth. code grant
Cisco IDP login form
user enters credentials
user credentials
check credentials
generate redirect with auth. code and (CXCloud) state
302 redirect with auth. code & state to cxcloud redirect url
redirect with auth code & state
verify state parameter
Token request to Cisco IDP includes: grant type(code), code, client id, client secret, redirect URI (must match registered URI)
POST token request to exchange code for tokens
return tokens:
return tokens access, refresh, id in response body
Claims enhancer uses Cisco access tokens to access Cisco IT services.
NOTE:This can be invoked here or when the authorization code is returned, depending on whether we wish to briefly persist the Cisco tokens or the CX Cloud tokens
Invoke Claims Enhancer
get SAVA info for user
Uses Cisco access token
SAVA info
get entitlement info for user
Uses Cisco access token
Entitlement info
get user info for user
Uses Cisco access token
User info
Build enhanced access token as JWT
Build CX IDP id token
access and id token for CX IDP
generate authorization code
persist CX Cloud tokens using authorization code or state as lookup key
Okta and Auth0 do this a bit differently
302 redirect with auth. code & state to App's redirect url
redirect with code & state
redirect URL's static HTML enables browser access to code & state from response
verify state
Token request to CX Cloud IDP includes: grant type(code), PXCE code_verifier, code, client id, redirect URI (must match registered URI if required)
POST token request to exchange code for tokens
lookup tokens using state or code as key
CX IDP tokens (id, access, refresh)
(re) load app
reload static html
static html
API request (with access token)
verifies access token JWT
performs authorization checks using JWT claims
API response

Figure 5: Auth Sequence #1

9 FAQ

9.1 Will customers be allowed to create custom RBAC roles?

Not at this time.

10 Discussion

10.1 Discussion 1

[David Stought]> David Wang I have reviewed the document and I am not following the last bullet in T1 and the last bullet in T2. I don't understand the scenario in which the user will be provided a prompt for consent after they successfully logged in. I am not sure what the user would be consenting to.

[David Wang] OAuth2 allows for delegated authorization. The first few minutes of the OAuth2/OIDC video has a good example. If Yelp needs access to Google to read your contacts, you don't have to give your Google password to Yelp. Instead, Yelp is pre-registered as a client to the Google IDP, and yelp forwards you to google. Once at google you type your google password directly to google, and then you are prompted on which authorization scopes that you would like to delegate to Yelp which is the client. If you approve the requested permissions, then the user is forwarded back to Yelp along with the id_token and access_token, which now Yelp has access to in order to make calls directly to the Google Contacts API. This access_token is scoped only with the permissions requested and approved by the user.

We have the same pattern here. Our systems in CX Cloud AWS need access to some information inside of the cisco IDP resources servers, especially the userinfo endpoint where we can read additional info about the user such as LDAP groups, full name, address, email, etc. We're not going to ask the user to give CX Cloud our Cisco passwords. That would be highly insecure, and not the type of liability that we would like to take on especially if the Cisco IT IdP can handle it. In fact, we never need to know our user's Cisco passwords because we can forward the user to Cisco, ask the user to delegate an authoriation that enables us to make very specific calls to the Cisco userinfo endpoint, and have the user present an id_token signed by the Cisco IdP to prove they are who they say they are. Subsequently, our system will use the access_token to query Cisco for additional user information.

One thing that might make you feel better. The user is only prompted to approve authorizations scopes for a particular client once. If approval has already occurred in the past, then the prompt will not show up for subsequent login requests.

Even if we decide we don't need any information from the Cisco IdP userinfo endpoint, we would still need the user to consent to the "openid" scope so that we may receive an id_token from the user to prove authentication.

10.2 Discussion 2

[David Stought] and based on the video the CX Cloud IdP only gets the authorization code. Not until the JWT Claims Enhancer sends the authorization code to the Cisco IdP will it get the access token and the id token correct?

[David Wang] Yes, you are correct.

10.3 Discussion 3

[Tony Porterfield] Does Cisco IDP allow users to revoke consent given to apps? For example if a user no longer wants to allow a CX app to access user info from Cisco IDP & associated APIs?

10.4 Discussion 4

[Tony Porterfield] is “staleness” of the context info stored in the JWT a concern? The data in the JWT will be a snapshot of state at the start of the session, and if group memberships, permissions, entitlements etc change this will not be reflected in the user session until the JWT is regenerated. (These don’t seem likely to change often FWIW)

The JWT could be regenerated at each refresh if necessary to keep the “lag” short if things change, or regenerated if an authz fails, then retried for a more immediate update. Else it would require logout/login to see changes

10.5 Discussion 5

[Tony Porterfield] Will the CX Cloud IDP handle “session management” of the access and refresh tokens it issues? This would include invalidating on explicit logout, as well as a mechanism for idle timeout. Idle timeout should, if possible, delete any tokens that contain PII/confidential info from the browser side.

10.6 Discussion 6

[Tony Porterfield] If a user logs out of Cisco IDP do we wish/expect the session of the CX cloud app to end, to give a single-sign-out experience? Likewise, do we want a logout from the CX cloud app to terminate the Cisco IDP session?

10.7 Discussion 7

[Tony Porterfield] One effect of using the stateless/JWT pattern is that if a user logs out, an access token is still valid until it expires. And, if we have a key compromise or token compromise we can't revoke unexpired JWTs. We'd delete the tokens from the user browser on explicit logout, which mitigates the risk of the token being used after a logout. But there are other paths for token exposure so we are accepting some risk with this design. and it's a consideration when we balance token validity duration against load on the server for refreshes.

10.8 Discussion 8

[Tony Porterfield] The tokens should be stored in browser session storage, not local storage, so that they will be deleted if the browser is closed. Browser local/session storage does not have the same protections against XSS and unencrypted transmission that cookies do, so we'll need to be sure we have strong controls for those elsewhere in the design

10.9 Discussion 9

[Tony Porterfield] If the access token JWT's enhanced claims will contain information that should not be exposed at the user client, we should encrypt the JWT.

10.10 Discussion 10

[Tony Porterfield] If possible, use CORS filters on the token endpoint in CX Cloud IDP, to mitigate impersonator apps.

10.11 Discussion 11

[Tony Porterfield] In step T1, I think the implicit grant type does not return a refresh token. Best practice for untrusted apps and OIDC/oAuth has moved toward code grant flow with PKCE. For these 2 reasons, suggest we consider code grant + PKCE for this grant request. Cognito supports PKCE (https://aws.amazon.com/blogs/mobile/understanding-amazon-cognito-user-pool-oauth-2-0-grants/)

This is a good general writeup on the topic https://www.pingidentity.com/en/company/blog/posts/2018/securely-using-oidc-authorization-code-flow-public-client-single-page-apps.html

10.12 Discussion 12

[Tony Porterfield] the state parameter should be used with Cisco IDP if supported, and should be used and checked in the CX Cloud IDP if we change to code flow (see Question 11)

11 Appendix A: RBAC Sketches

11.1 Product Roles

11.1.1 Role: Product Administrator

The Product Administrator or `product-admin` role is the most privileged role available in CX Cloud. This role has permission to perform all possible RPCs on all possible resources for all possible customer account IDs.

WARNING: As with all roles with the `product-` prefix, it MUST only ever be assigned to Cisco employees or HIGHLY TRUSTED contractors that have a global need to know and/or ability to do.

Resources (ALL ACCOUNTS)	Create	Get	Update	Delete	List	ALL
Atticastatus	X	X	X	X	X	X
BlockedDate	X	X	X	X	X	X
BusinessProcess	X	X	X	X	X	X
Cdet	X	X	X	X	X	X
CdetInfo	X	X	X	X	X	X
Cdetstatus	X	X	X	X	X	X
CollectionSchedule	X	X	X	X	X	X
CommandBlackList	X	X	X	X	X	X
ContractCoverage	X	X	X	X	X	X
ContractCoverageReport	X	X	X	X	X	X
Cve	X	X	X	X	X	X
DataCollector	X	X	X	X	X	X
Equipment	X	X	X	X	X	X
EquipmentReport	X	X	X	X	X	X

Resources (ALL ACCOUNTS)	Create	Get	Update	Delete	List	ALL
FieldNotice	X	X	X	X	X	X
FieldNoticeBulletin	X	X	X	X	X	X
FieldNoticeReport	X	X	X	X	X	X
FnCdet	X	X	X	X	X	X
HardwareView	X	X	X	X	X	X
HwEoL	X	X	X	X	X	X
HwEoLBulletin	X	X	X	X	X	X
HwEoLMilestone	X	X	X	X	X	X
HwEoLReport	X	X	X	X	X	X
ICType	X	X	X	X	X	X
Methods	X	X	X	X	X	X
NeScan	X	X	X	X	X	X
NetworkElement	X	X	X	X	X	X
NetworkElementReport	X	X	X	X	X	X
OrganizationUnit	X	X	X	X	X	X
PidMigration	X	X	X	X	X	X
Policy	X	X	X	X	X	X
PolicyGroup	X	X	X	X	X	X
ProductCoverage	X	X	X	X	X	X
ProductEligibility	X	X	X	X	X	X
ProductEligibilityInfo	X	X	X	X	X	X
ResourceTag	X	X	X	X	X	X
ScanResult	X	X	X	X	X	X
SecAdvisoryBulletin	X	X	X	X	X	X
SecurityAdvisory	X	X	X	X	X	X
SecurityAdvisoryCdet	X	X	X	X	X	X
SecurityAdvisoryReport	X	X	X	X	X	X

Resources (ALL ACCOUNTS)	Create	Get	Update	Delete	List	ALL
SucessTrackInfo	X	X	X	X	X	X
SwEoL	X	X	X	X	X	X
SwEoLAlertBulletin	X	X	X	X	X	X
SwEoLMilestone	X	X	X	X	X	X
SwEoLReport	X	X	X	X	X	X
SystemView	X	X	X	X	X	X

ref: https://www-github3.cisco.com/cxe/cx-data-model-proto/blob/master/src/cx/v1/cx.js.proto

11.1.2 Role: Product Writer

The Product Writer or `product-writer` role is the second most privileged role available in CX Cloud. This role has permission to perform all CRUD+L RPCs on all possible resources for all possible customer account IDs.

WARNING: As with all roles with the `product-` prefix, it MUST only ever be assigned to Cisco employees or HIGHLY TRUSTED contractors that have a global need to know and/or ability to do.

Resources (ALL ACCOUNTS)	Create	Get	Update	Delete	List	ALL
Atticastatus	X	X	X	X	X	
BlockedDate	X	X	X	X	X	
BusinessProcess	X	X	X	X	X	
Cdet	X	X	X	X	X	
CdetInfo	X	X	X	X	X	
Cdetstatus	X	X	X	X	X	
CollectionSchedule	X	X	X	X	X	
CommandBlackList	X	X	X	X	X	
ContractCoverage	X	X	X	X	X	
ContractCoverageReport	X	X	X	X	X	
Cve	X	X	X	X	X	

Resources (ALL ACCOUNTS)	Create	Get	Update	Delete	List	ALL
DataCollector	X	X	X	X	X	
Equipment	X	X	X	X	X	
EquipmentReport	X	X	X	X	X	
FieldNotice	X	X	X	X	X	
FieldNoticeBulletin	X	X	X	X	X	
FieldNoticeReport	X	X	X	X	X	
FnCdet	X	X	X	X	X	
HardwareView	X	X	X	X	X	
HwEoL	X	X	X	X	X	
HwEoLBulletin	X	X	X	X	X	
HwEoLMilestone	X	X	X	X	X	
HwEoLReport	X	X	X	X	X	
ICType	X	X	X	X	X	
Methods	X	X	X	X	X	
NeScan	X	X	X	X	X	
NetworkElement	X	X	X	X	X	
NetworkElementReport	X	X	X	X	X	
OrganizationUnit	X	X	X	X	X	
PidMigration	X	X	X	X	X	
Policy	X	X	X	X	X	
PolicyGroup	X	X	X	X	X	
ProductCoverage	X	X	X	X	X	
ProductEligibility	X	X	X	X	X	
ProductEligibilityInfo	X	X	X	X	X	
ResourceTag	X	X	X	X	X	
ScanResult	X	X	X	X	X	
SecAdvisoryBulletin	X	X	X	X	X	

Resources (ALL ACCOUNTS)	Create	Get	Update	Delete	List	ALL
SecurityAdvisory	X	X	X	X	X	
SecurityAdvisoryCdet	X	X	X	X	X	
SecurityAdvisoryReport	X	X	X	X	X	
SucessTrackInfo	X	X	X	X	X	
SwEoL	X	X	X	X	X	
SwEoLAlertBulletin	X	X	X	X	X	
SwEoLMilestone	X	X	X	X	X	
SwEoLReport	X	X	X	X	X	
SystemView	X	X	X	X	X	

11.1.3 Role: Product Reader

The Product Reader or `product-reader` role is the third most privileged role available in CX Cloud. This role has permission to perform read and list RPCs on all possible resources for all possible customer account IDs.

WARNING: As with all roles with the `product`– prefix, it MUST only ever be assigned to Cisco employees or HIGHLY TRUSTED contractors that have a global need to know and/or ability to do.

Resources (ALL ACCOUNTS)	Create	Get	Update	Delete	List	ALL
Atticastatus		X			X	
BlockedDate		X			X	
BusinessProcess		X			X	
Cdet		X			X	
CdetInfo		X			X	
Cdetstatus		X			X	
CollectionSchedule		X			X	
CommandBlackList		X			X	
ContractCoverage		X			X	
ContractCoverageReport		X			X	

Resources (ALL ACCOUNTS)	Create	Get	Update	Delete	List	ALL
Cve		X			X	
DataCollector		X			X	
Equipment		X			X	
EquipmentReport		X			X	
FieldNotice		X			X	
FieldNoticeBulletin		X			X	
FieldNoticeReport		X			X	
FnCdet		X			X	
HardwareView		X			X	
HwEoL		X			X	
HwEoLBulletin		X			X	
HwEoLMilestone		X			X	
HwEoLReport		X			X	
ICType		X			X	
Methods		X			X	
NeScan		X			X	
NetworkElement		X			X	
NetworkElementReport		X			X	
OrganizationUnit		X			X	
PidMigration		X			X	
Policy		X			X	
PolicyGroup		X			X	
ProductCoverage		X			X	
ProductEligibility		X			X	
ProductEligibilityInfo		X			X	
ResourceTag		X			X	
ScanResult		X			X	

Resources (ALL ACCOUNTS)	Create	Get	Update	Delete	List	ALL
SecAdvisoryBulletin		X			X	
SecurityAdvisory		X			X	
SecurityAdvisoryCdet		X			X	
SecurityAdvisoryReport		X			X	
SucessTrackInfo		X			X	
SwEoL		X			X	
SwEoLAlertBulletin		X			X	
SwEoLMilestone		X			X	
SwEoLReport		X			X	
SystemView		X			X	

11.2 Account Roles

11.2.1 Role: Account Administrator

The Account Administrator or `account-admin` role is the most privileged role available to a CUSTOMER of CX Cloud. This role has permission to perform all possible RPCs on all possible resources for a specific customer account ID.

WARNING: As with all roles with the `account-` prefix, it SHOULD only ever be assigned to account holders that have an account-global need to know and/or ability to do.

Resources (1 ACCOUNT)	Create	Get	Update	Delete	List	ALL
Atticastatus	X	X	X	X	X	X
BlockedDate	X	X	X	X	X	X
BusinessProcess	X	X	X	X	X	X
Cdet	X	X	X	X	X	X
CdetInfo	X	X	X	X	X	X
Cdetstatus	X	X	X	X	X	X
CollectionSchedule	X	X	X	X	X	X

Resources (1 ACCOUNT)	Create	Get	Update	Delete	List	ALL
CommandBlackList	X	X	X	X	X	X
ContractCoverage	X	X	X	X	X	X
ContractCoverageReport	X	X	X	X	X	X
Cve	X	X	X	X	X	X
DataCollector	X	X	X	X	X	X
Equipment	X	X	X	X	X	X
EquipmentReport	X	X	X	X	X	X
FieldNotice	X	X	X	X	X	X
FieldNoticeBulletin	X	X	X	X	X	X
FieldNoticeReport	X	X	X	X	X	X
FnCdet	X	X	X	X	X	X
HardwareView	X	X	X	X	X	X
HwEoL	X	X	X	X	X	X
HwEoLBulletin	X	X	X	X	X	X
HwEoLMilestone	X	X	X	X	X	X
HwEoLReport	X	X	X	X	X	X
ICType	X	X	X	X	X	X
Methods	X	X	X	X	X	X
NeScan	X	X	X	X	X	X
NetworkElement	X	X	X	X	X	X
NetworkElementReport	X	X	X	X	X	X
OrganizationUnit	X	X	X	X	X	X
PidMigration	X	X	X	X	X	X
Policy	X	X	X	X	X	X
PolicyGroup	X	X	X	X	X	X
ProductCoverage	X	X	X	X	X	X
ProductEligibility	X	X	X	X	X	X

Resources (1 ACCOUNT)	Create	Get	Update	Delete	List	ALL
ProductEligibilityInfo	X	X	X	X	X	X
ResourceTag	X	X	X	X	X	X
ScanResult	X	X	X	X	X	X
SecAdvisoryBulletin	X	X	X	X	X	X
SecurityAdvisory	X	X	X	X	X	X
SecurityAdvisoryCdet	X	X	X	X	X	X
SecurityAdvisoryReport	X	X	X	X	X	X
SucessTrackInfo	X	X	X	X	X	X
SwEoL	X	X	X	X	X	X
SwEoLAlertBulletin	X	X	X	X	X	X
SwEoLMilestone	X	X	X	X	X	X
SwEoLReport	X	X	X	X	X	X
SystemView	X	X	X	X	X	X

ref: https://www-github3.cisco.com/cxe/cx-data-model-proto/blob/master/src/cx/v1/cx.js.proto

11.2.2 Role: Account Writer

The Account Writer or `account-writer` role is the second most privileged role available in CX Cloud. This role has permission to perform all CRUD+L RPCs on all possible resources for a specific customer account ID.

WARNING: As with all roles with the `account-` prefix, it SHOULD only ever be assigned to Cisco employees or HIGHLY TRUSTED contractors that have a global need to know and/or ability to do.

Resources (1 ACCOUNT)	Create	Get	Update	Delete	List	ALL
Atticastatus	X	X	X	X	X	
BlockedDate	X	X	X	X	X	
BusinessProcess	X	X	X	X	X	
Cdet	X	X	X	X	X	

Resources (1 ACCOUNT)	Create	Get	Update	Delete	List	ALL
CdetInfo	X	X	X	X	X	
Cdetstatus	X	X	X	X	X	
CollectionSchedule	X	X	X	X	X	
CommandBlackList	X	X	X	X	X	
ContractCoverage	X	X	X	X	X	
ContractCoverageReport	X	X	X	X	X	
Cve	X	X	X	X	X	
DataCollector	X	X	X	X	X	
Equipment	X	X	X	X	X	
EquipmentReport	X	X	X	X	X	
FieldNotice	X	X	X	X	X	
FieldNoticeBulletin	X	X	X	X	X	
FieldNoticeReport	X	X	X	X	X	
FnCdet	X	X	X	X	X	
HardwareView	X	X	X	X	X	
HwEoL	X	X	X	X	X	
HwEoLBulletin	X	X	X	X	X	
HwEoLMilestone	X	X	X	X	X	
HwEoLReport	X	X	X	X	X	
ICType	X	X	X	X	X	
Methods	X	X	X	X	X	
NeScan	X	X	X	X	X	
NetworkElement	X	X	X	X	X	
NetworkElementReport	X	X	X	X	X	
OrganizationUnit	X	X	X	X	X	
PidMigration	X	X	X	X	X	
Policy	X	X	X	X	X	

Resources (1 ACCOUNT)	Create	Get	Update	Delete	List	ALL
PolicyGroup	X	X	X	X	X	
ProductCoverage	X	X	X	X	X	
ProductEligibility	X	X	X	X	X	
ProductEligibilityInfo	X	X	X	X	X	
ResourceTag	X	X	X	X	X	
ScanResult	X	X	X	X	X	
SecAdvisoryBulletin	X	X	X	X	X	
SecurityAdvisory	X	X	X	X	X	
SecurityAdvisoryCdet	X	X	X	X	X	
SecurityAdvisoryReport	X	X	X	X	X	
SucessTrackInfo	X	X	X	X	X	
SwEoL	X	X	X	X	X	
SwEoLAlertBulletin	X	X	X	X	X	
SwEoLMilestone	X	X	X	X	X	
SwEoLReport	X	X	X	X	X	
SystemView	X	X	X	X	X	

11.2.3 Role: Account Reader

The Account Reader or `account-reader` role is the third most privileged role available in CX Cloud. This role has permission to perform read and list RPCs on all possible resources for a specific customer account ID.

WARNING: As with all roles with the `account`– prefix, it SHOULD only ever be assigned to Cisco employees or HIGHLY TRUSTED contractors that have a global need to know and/or ability to do.

Resources (1 ACCOUNT)	Create	Get	Update	Delete	List	ALL
Atticastatus		X			X	
BlockedDate		X			X	
BusinessProcess		X			X	

Resources (1 ACCOUNT)	Create	Get	Update	Delete	List	ALL
Cdet		X			X	
CdetInfo		X			X	
Cdetstatus		X			X	
CollectionSchedule		X			X	
CommandBlackList		X			X	
ContractCoverage		X			X	
ContractCoverageReport		X			X	
Cve		X			X	
DataCollector		X			X	
Equipment		X			X	
EquipmentReport		X			X	
FieldNotice		X			X	
FieldNoticeBulletin		X			X	
FieldNoticeReport		X			X	
FnCdet		X			X	
HardwareView		X			X	
HwEoL		X			X	
HwEoLBulletin		X			X	
HwEoLMilestone		X			X	
HwEoLReport		X			X	
ICType		X			X	
Methods		X			X	
NeScan		X			X	
NetworkElement		X			X	
NetworkElementReport		X			X	
OrganizationUnit		X			X	
PidMigration		X			X	

Resources (1 ACCOUNT)	Create	Get	Update	Delete	List	ALL
Policy		X			X	
PolicyGroup		X			X	
ProductCoverage		X			X	
ProductEligibility		X			X	
ProductEligibilityInfo		X			X	
ResourceTag		X			X	
ScanResult		X			X	
SecAdvisoryBulletin		X			X	
SecurityAdvisory		X			X	
SecurityAdvisoryCdet		X			X	
SecurityAdvisoryReport		X			X	
SucessTrackInfo		X			X	
SwEoL		X			X	
SwEoLAlertBulletin		X			X	
SwEoLMilestone		X			X	
SwEoLReport		X			X	
SystemView		X			X	

11.3 What are some example PRODUCT users and their roles?

User	Company	Description	Roles
caleb@cisco.com	Cisco	Product Administrator	`CXCloudDataAdmin`
charlie@cisco.com	Cisco	IBES Administrator	`CXCloudDataUploader`
cbot1@cisco.com	Cisco	Machine Account at Collector	`CXCloudDataUploader`
cbot2@cisco.com	Cisco	Machine Account at IBES	`CXCloudDataUploader`
cora@cisco.com	Cisco	Customer Support Representative	`CXCloudDataWriter`

11.4 What are some example ACCOUNT users and their roles?

User	Company	Description	Roles
alice@acme.com	Acme	Account Administrator	`AccountAdmin` or `VirtualAccountAdmin`
abot2@acme.com	Acme	Machine Account	`AccountAdmin` or `VirtualAccountAdmin`
abot3@acme.com	Acme	Machine Account	`AccountFullUser` or `VirtualAccountFullUser`

12 Appendix B: Brain Dump

- Sequence:
  - (normal authentication flow)
  - CX Cloud IdP (Auth0) is prompted to generate a claims data-enriched auth token.
  - In order to obtain the claims data used to enrich the token, an Auth0 Hook is called.
  - The Auth0 Hook makes a gRPC call to CX Cloud's auth claims resolver API.
  - The CX Cloud auth claims resolver API contacts SAVA to get a list of account IDs, calls EB (TODO: be specific about for what) and returns the claim as it should be used in the final auth token.

13 Sources

1. INCITS 359-2012: Information technology - Role Based Access Control
2. Role Based Access Control (RBAC) NIST Slides
3. Role-Based Access Control, 2nd edition (2007) by Ferraiolo, Chandramouli, & Kuhn
4. NIST Role Based Access Control Model
5. Role Engineering and RBAC Standards

6. RBAC Case Studies
7. RBAC Library
8. INCITS 359-2012 (R2017)
9. Open Policy Agent: Comparison to Other Systems