SMPTE PCD ST 2138-50:202x

SMPTE Public Committee Draft

Catena — Authenticity - Integrity - Access Control - Confidentiality and Availability

Table of Contents Page

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices, and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in its Standards Operations Manual. This SMPTE Engineering Document was prepared by Technology Committee 34CS.

Introduction

This clause is entirely informative and does not form an integral part of this Engineering Document.

Security is, by necessity, at the heart of the Catena initiative. When controlling media microservices in the cloud, it is paramount that all data flows remain absolutely secure from interference and interruption by bad actors.

This is ideally accomplished via industry-accepted best practices in the area of security.

1 Scope

This document specifies how to securely utilize the Catena control protocol (as defined in ST 2138) with respect to authenticity, integrity, access control, confidentiality and availability.

This document is applicable to how the following are to be supported by compliant devices and services:

- Authenticity
  - Clients can verify that communications are from the intended device and no other source.
  - Devices can verify that the access tokens accompanying each request were issued by the authorization server in the token's iss claim.
- Integrity — Clients can trust that communications from a compliant device or service have not been tampered with in transit.
- Interoperability — Which OAuth2 Access Control flows are used by clients and supported by a compliant device or service.
- Access Control
  - The responsibility of the device to provide Policy Enforcement Point functionality, and how this responsibility may optionally be divided between devices and gateways.
  - Access revocation functionality required by the device.
  - Correct handling in the event of access token expiry.
- Confidentiality — Confidentiality of data in transit.
- Availability — How *timely* access to device capabilities is promoted by using JWS rather than opaque access tokens, thereby avoiding round trips by devices to the Authorization server's introspection endpoint.

The following are outside of the scope of this document:

- Policy Decision Point and Policy Engine functionality, including the following:
    - Authentication — how a user proves their identity
    - Auditing — Access, Authorization and Audit logging
    - Non-repudiation
    - Availability in terms of high availability or device redundancy

2 Conformance

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any clause explicitly labeled as "Informative" or individual paragraphs that start with "Note:”

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition; tables shall be next; then formal languages; then figures; and then any other language forms.

3 Normative References

The following Standard contains provisions that, through reference in this text, constitute provisions of this standard. Dated references require that the specific edition cited shall be used as the reference. Undated citations refer to the edition of the referenced document (including any amendments) current at the date of publication of this document. All standards are subject to revision, and users of this engineering document are encouraged to investigate the possibility of applying the most recent edition of any undated reference.

IETF RFC 6749, *OAuth2.0 Authorization Framework, Internet Engineering Task Force (IETF), 2012*

IETF RFC 7515, *JSON Web Signature (JWS), Internet Engineering Task Force (IETF), 2015*

IETF RFC 7518, *JSON Web Algorithms, Internet Engineering Task Force (IETF), 2015*

IETF RFC 7519, *JSON Web Token (JWT), Internet Engineering Task Force (IETF), 2015*

IETF RFC 7636, *Proof Key for Code Exchange by OAuth Public Clients (IETF), 2015*

IETF RFC 8446, *Transport Layer Security (TLS) Protocol Version 1.3, Internet Engineering Task Force (IETF), 2018*

IETF RFC 8725, *JSON Web Token Best Current Practices, Internet Engineering Task Force (IETF), 2020*

NIST SP 800-186, *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters, 2023*

NIST 800-207, *Zero Trust Architecture, S. Rose, O. Borchert, S. Mitchell, and S. Connelly, National Institute of Standards and Technology, 2020*

NIST FIPS 180-4, *Secure Hash Standard, S. Rose, National Institute of Standards and Technology, 2015*

SMPTE ST 2138-10:202x, *Catena – Model*. Pending publication.

4 Terms and Definitions

For the purposes of this document, the following terms and definitions apply:

4.1
Policy Enforcement Point
PEP

system entity that requests and subsequently enforces authorization decisions

4.2
OpenID Connect
OIDC

simple, standardized identity layer built on top of OAuth 2.0 that enables users to log in once and access multiple applications (Single Sign-On)

Note 1 to entry: More detail can be found at https://openid.net/.

4.3
Catena API

code for implementing ST 2138 as found in https://github.com/SMPTE/st2138-a/tree/main/interface

Note 1 to entry: https://github.com/SMPTE/st2138-a/tree/main/interface is a sub-folder within https://github.com/SMPTE/st2138-a.git (see ST 2138-10:202x Annex B Additional Elements).

4.4
Enclave

secure, isolated network created by installing software agents on each endpoint to enforce strict access controls, authenticating and encrypting all communication between systems based on identity, not location

4.5
JSON Web Key Set
JWKS

standardized JSON document containing public keys used to validate signed JSON Web Tokens

4.6 Proof Key for Code Exchange Authorization Code Flow
PKCE Code Flow

OAuth2.0 authorization flow that binds the authorization code to the client using a cryptographic proof key

Note 1 to entry: PKCE Code Flow mitigates interception attacks as defined in IETF RFC 7636 and used by Open IDConnect.

5 Precedence

In the event of a conflict between the schema found in https://github.com/SMPTE/st2138-a/blob/main/interface/schemata/device.yaml (see ST 2138-10, Annex B Additional Elements) and other information in this document, the schema shall take precedence.

6 Secure Communications

Catena devices and clients shall use TLS v1.2 or later to protect data in motion. If the IETF deprecates TLS v1.2, it shall be possible to upgrade devices and clients to a later version of TLS.

7 Policy Enforcement

Devices that support the Catena Interface shall follow NIST's Zero Trust (ZT) Architecture (specified in NIST 800-207) as Resource Servers with a Policy Enforcement Point between the user agent and the Device.

At a high level, the architecture creates a trusted zone that subjects shall access only via a Policy Decision Point working in cooperation with a Policy Enforcement Point, as shown in Figure 1.

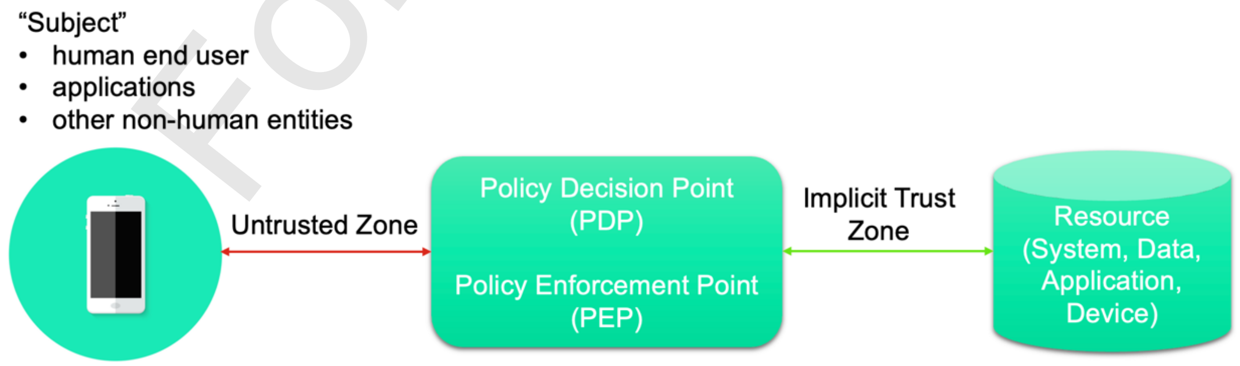
"Subject"
• human end user
• applications
• other non-human entities
Untrusted Zone
Policy Decision Point (PDP)
Policy Enforcement Point (PEP)
Implicit Trust Zone
Resource (System, Data, Application, Device)

Figure 1 — Zero Trust Access

Figure 2 shows the various inputs to the Policy Decision Point, some of which may be real-time data streams.

The inputs as shown in Figure 2 are as follows:

- Continuous Diagnostics and Mitigation (CDM)
- Compliance systems
- Threat intelligence feeds
- Authentication, Authorization and Access (AAA) logs
- Data Access Policy
- Public Key Infrastructure (PKI) — can also be Private in some Enterprises)
- Identity management systems such as LDAP and SAML
- Security Information and Event Management (SIEM) systems

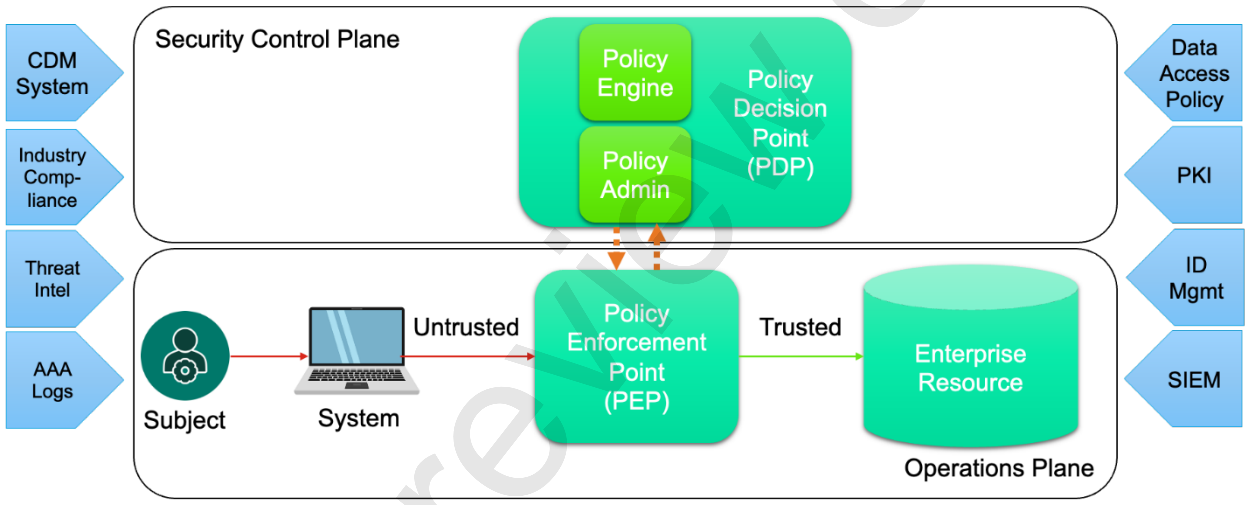

Figure 2 — Core Zero Trust Logical Components

The PDP comprises the Policy Engine and the Policy Admin.

- The Policy Engine runs an algorithm using these inputs, which determines the ultimate grant/deny decision.
- The Policy Admin is responsible for creating and tearing down communication paths between the subject and the Enterprise Resource via commands to the Policy Enforcement Point. In OAuth2 (as defined in IETF RFC 6789), this is where access tokens are generated and distributed.
- In some implementations, the Policy Engine and Policy Admin may be integrated into a single entity.

The Policy Enforcement Point shall enable, monitor, and terminate connections between subject and Enterprise Resource. It is logically a single point, but may be broken into two components: client side (user agent), and resource side (gateway controlling access to Enterprise Resource). It may also be a single portal, as shown in Figure 2.

NIST 800-207's Device Agent/Gateway Based Deployment model (illustrated in Figure 3) shall be utilized for devices that natively support Catena's interface. The PEP functionality may be implemented solely on the device, or may be distributed across the device and collaborating application gateways — an architecture commonly used in service mesh architectures.

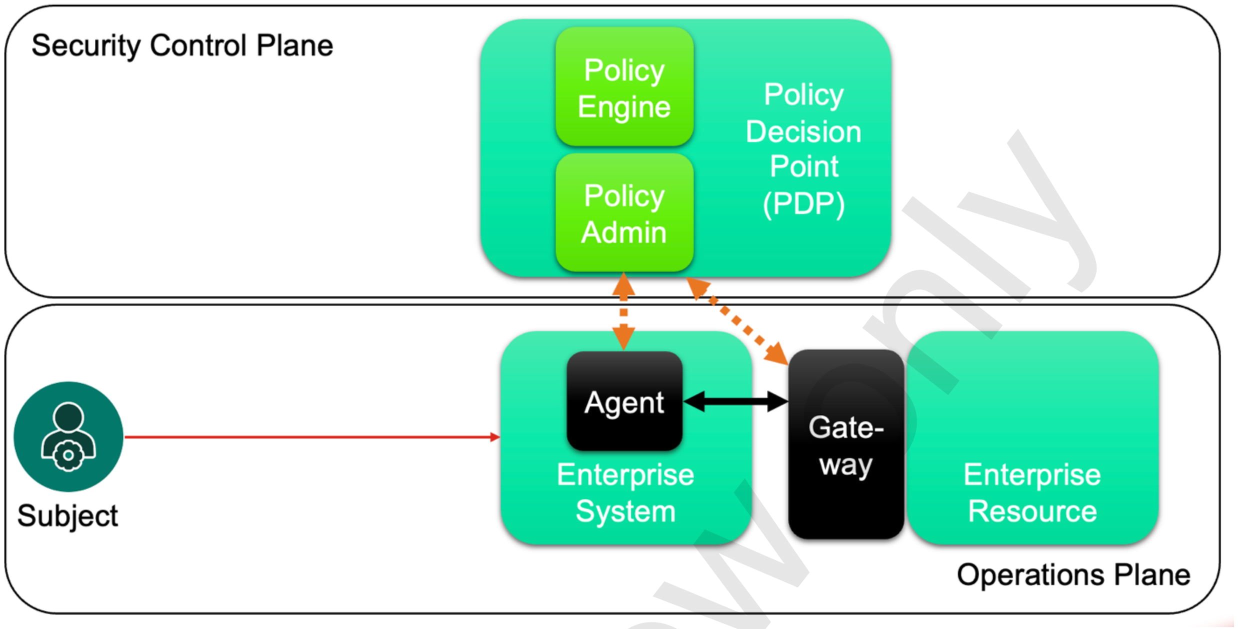

Figure 3 — Device Agent / Gateway Model

Older legacy equipment may be adapted to the Catena framework using the Enclave-Based deployment model shown in Figure 4, in which the PEP resides in a Gateway in a secure Enclave containing the devices.

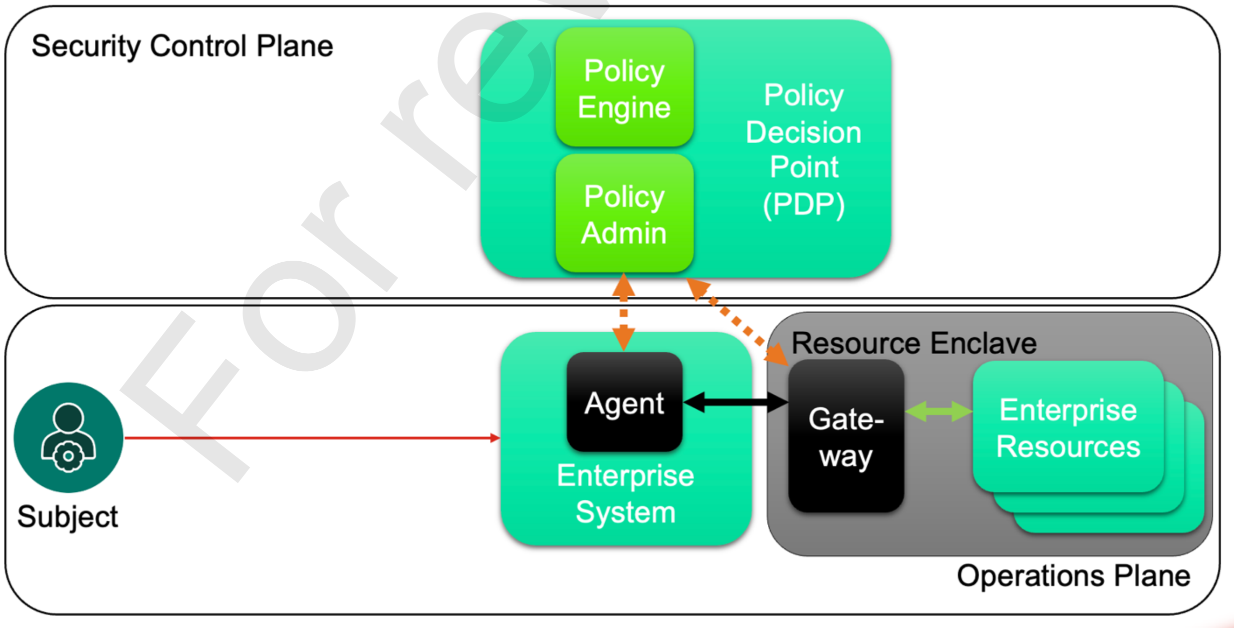

Figure 4 — Device Agent / Enclave Model

For completeness, NIST SP 800-207 also documents a Resource Portal model (shown in Figure 5) for situations where it is not possible to install an Agent on the system used by the subject to gain access to the Enterprise Resources.

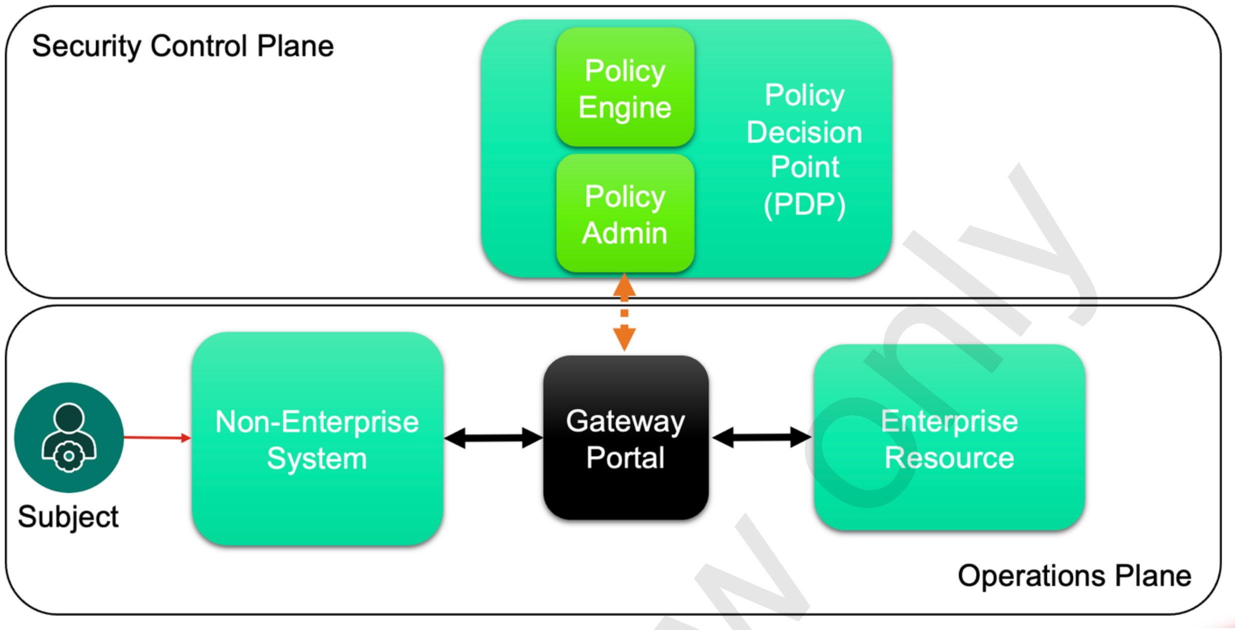

Figure 5 — Resource Portal Model

Each access request to a Catena Device/Resource Server in production shall be accompanied by an access token in compact serialization format per SMPTE ST 2138-10. The PEP shall use the access token to determine whether the requested action is authorized.

Devices in development or test environments may opt out of providing PEP functionality.

8 Overview of OAuth2 Authorization Flows

8.1 Authorization Flow Actors

The actors in the authorization flows specified in this document are as shown in Figure 6.

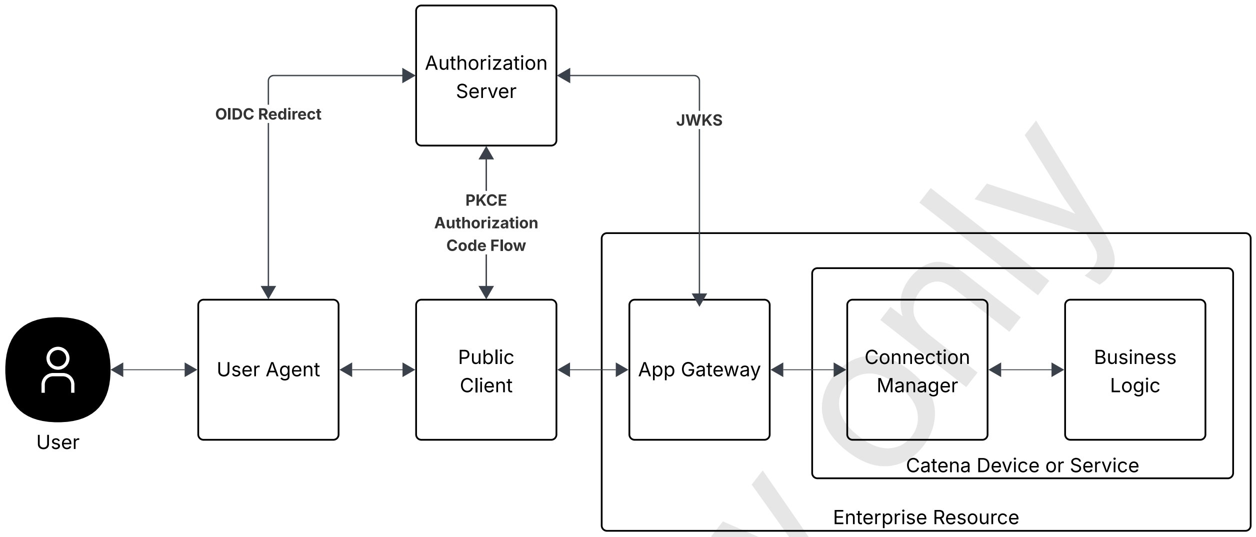

Figure 6 — Authorization Flow Actors

A brief description of each actor and flow component is as follows:

- User — human being requesting authorization.
- User Agent — i.e., a web browser, not necessarily hosted on the enterprise network.
- Public Client — typically a single page application. Hosted on the enterprise network.
- OIDC Redirect — a standards-defined HTTP redirection mechanism, whereby an OpenID Provider returns authentication responses to a pre-registered client redirect URI, as specified by OpenID Connect Core 1.0 and OAuth 2.0.
- Authorization Server — serves the OAuth2 APIs required here, hosted on the enterprise network.
- PKCE Authorization Code Flow — an OAuth2.0 authorization flow that binds the authorization code to the client using a cryptographic proof key, mitigating interception attacks as defined in IETF RFC 7636 and used by Open IDConnect.
- JSON Web Key Set (JWKS) — a standardized JSON document containing public keys used to validate signed JSON Web Tokens.
- Enterprise Resource — Comprising the optional App Gateway and the Resource Server.
    - App Gateway — optional, but commonly used as a “sidecar” process for routine access control checks.
    - Resource Server — comprising the Connection Manager and Business Logic.
        - Connection Manager — an implementation of one of the Catena APIs (gRPC or REST).
        - Business Logic — the device’s media processing function.

There shall be two flows as specified by this standard:

1. A user interacts with a service via a web application hosted on the enterprise network. This shall use the Proof Key for Code Exchange (PKCE) Authorization Code Flow.
2. A service interacts with another service. This shall use the Client Credentials Flow.

8.2 PKCE Authorization Code Flow

The sequence diagram in Figure 7 documents the PKCE Authorization Code Flow that is standardized in IETF RFC 7636.

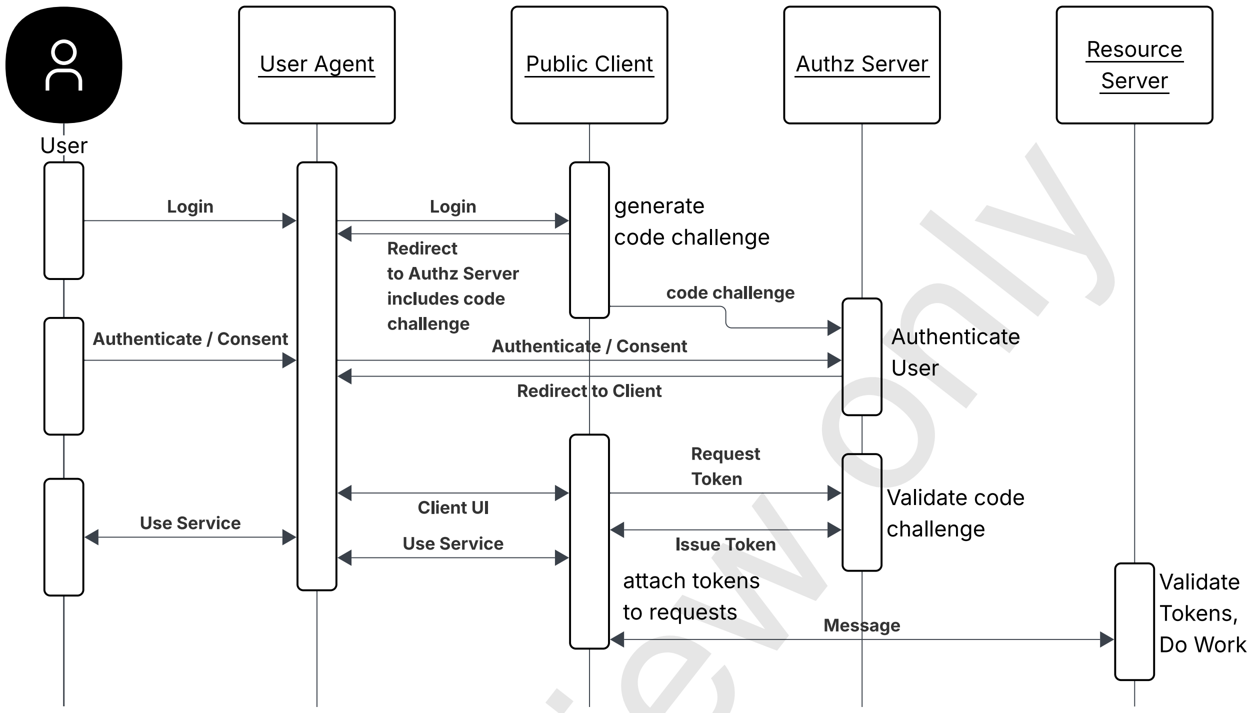

Figure 7 — PKCE Authorization Code Flow Sequence Diagram

1. The User shall make a login request of the Public Client via the User Agent (web browser).
2. The Public Client shall generate a code verifier and challenge. The code verifier shall be a high-entropy random string, and the challenge is its SHA-256 (as defined in NIST FIPS 180-4) digest.
3. The Public Client shall send the code challenge to the Authorization Server and shall redirect the User Agent to the Authorization Server's OIDC login prompt.
4. The Authorization Server shall authenticate the user, and on success, shall redirect the User Agent back to the Public Client.
5. The Public Client shall send a request to the Authorization Server for an Access Token. The request shall include the code verifier.
6. The Authorization Server shall verify that the code verifier matches the digest of the code challenge sent in step 3. It may then connect the authenticated user identity with the Public Client and issue an access token that encodes the user's access rights.

9 Catena Access Token Specification

The registered Claim Names in Table 1 shall be required:

Table 1 — Registered Claim Names

Claim	Comment
“iss” (Issuer)	Unambiguously identifies the issuer (i.e., authz server) that issued the token. The PEP shall be aware of and recognize the authz server.
“sub” (Subject)	Identifies the user — provides non-repudiation.
“aud” (Audience)	“catena.grpc.service” for gRPC, “https://catena-api” for REST.
“exp” (Expiration Time)	Access Token time-out.
“nbf” (Not Before)	Access Tokens are valid only after this time.
“iat” (Issued At)	Cannot be in the future.
“jti” (JWT ID)	Uniquely identifies the JWT. A UUID.
“scope”	A string containing space delimited scopes. Examples: “st2138:mon st2138:op:w st2138:cfg” provides read-only access to the device’s monitor and config scopes, read/write access to the operate scope, and no access to the admin scope.
“azp” (Authorized Party)	When a user has delegated an authorized party (such as a third-party application), this token indicates which client or third-party application is authorized to act on the user’s behalf.
“client_id”	Identity of the client that requested the access token.

10 Catena JSON Web Signature Specification

Catena Access tokens shall be signed when used in production. The signature algorithm shall conform to the ES256 algorithm as defined in IETF RFCs 7515 and 7518.

11 Validation of Catena Access Tokens

11.1 Payload Validation

11.1.1 Access Token Structure

Catena access tokens comprise 3 parts: header, payload and signature. They are encoded as base-64 strings that are delimited by the "." (period) character.

The subclauses that follow (11.1.2 through 11.1.5) define how the PEP validates API calls, as illustrated in the flowcharts in Figure 8, Figure 9, Figure 10, and Figure 11.

11.1.2 Jason Object Signing and Encryption (JOSE) Header Processing

- The JOSE header is decoded to a JSON object containing alg, type and, optionally, kid fields.
- The alg field shall be "ES256", which signifies that the algorithm used to produce the signature is Elliptic Curve with a 256 bit prime key field.
- The typ field shall be "JWT", which signifies that the body of the JWS is a JSON Web Token.
- The kid field, if present, identifies the public key used to sign the JWS and is used to retrieve it from the authorization server. If not present, the authorization server's default public key shall be used.

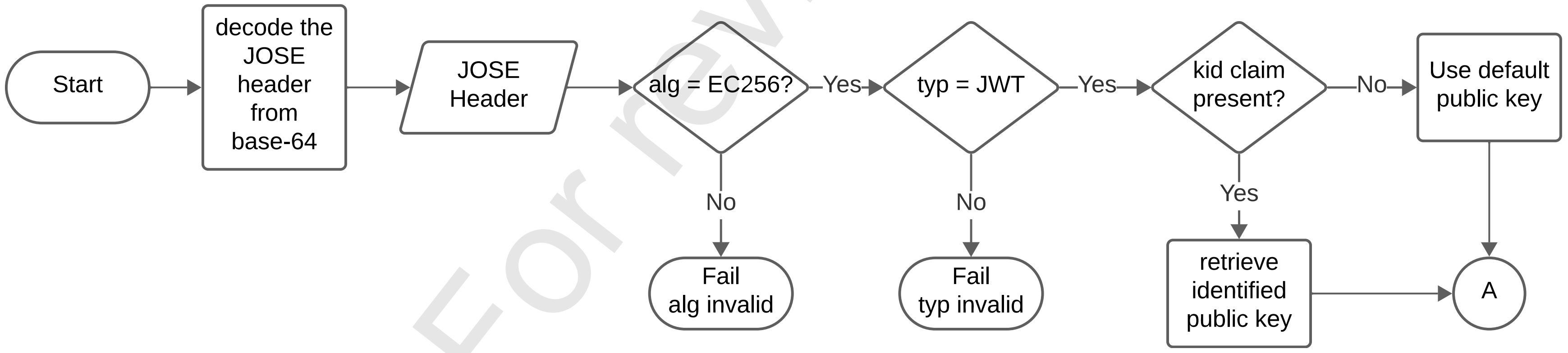

Figure 8 — JOSE Header Processing

11.1.3 Signature Validation

Once the public key has been obtained from the authorization server, it shall be used to validate the JWS's signature as specified in IETF RFC 7515 Section 5.2.

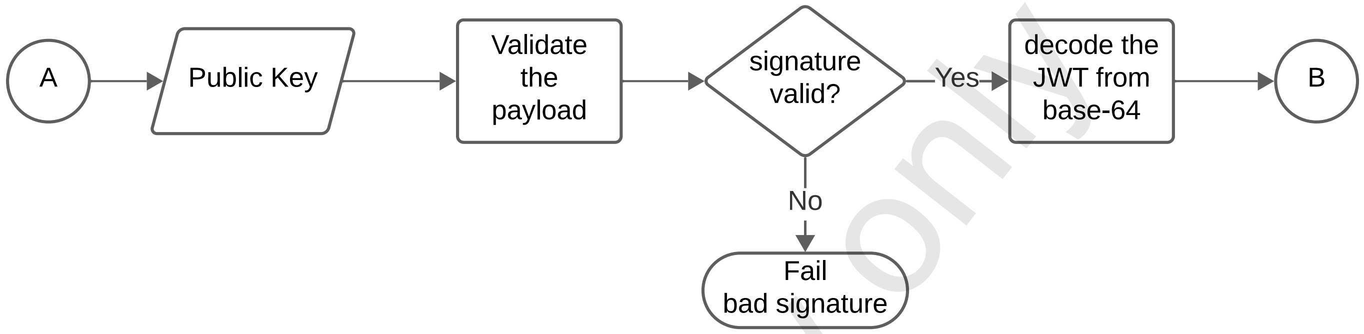

Figure 9 — Signature Validation

11.1.4 Issuer and Audience Validation

If the signature is valid, the JWS's body is decoded into a JSON object to enable its claims to be inspected.

The "iss" claim shall be the URL of the authorization server, otherwise the token shall be rejected.

The "aud" claim shall contain either "catena.grpc.service" or "catena.rest.service", otherwise the token shall be rejected.

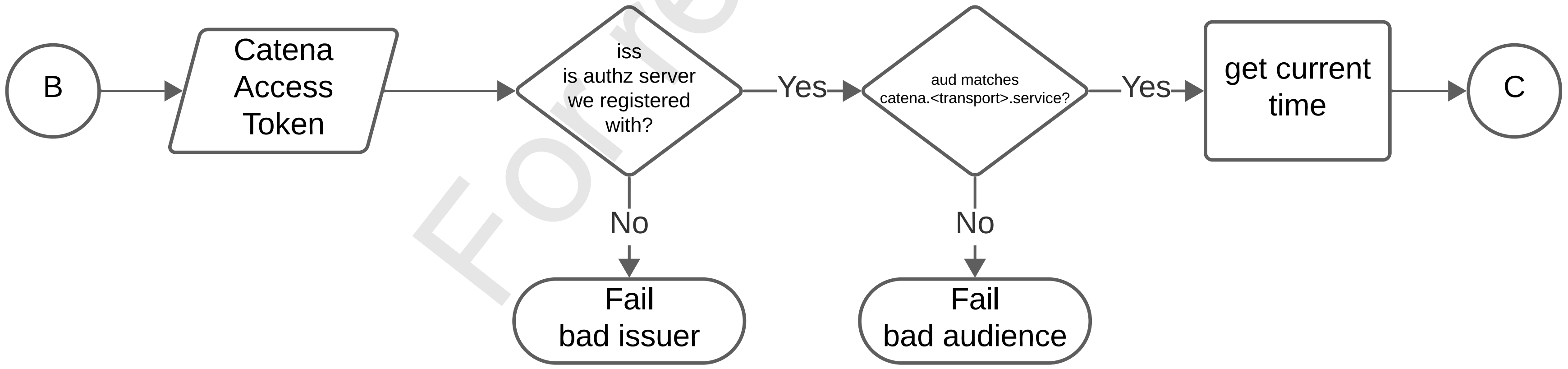

Figure 10 — Issuer and Audience Validation

11.1.5 Timebox and Access Validation

Access tokens are valid only for a specific time period. They shall be rejected if the current time is outside of that period as specified in IETF RFC 7519 Sections 4.1.4, 4.1.5, and 4.1.6.

One of the “client_id” or “azp” claims shall be present and match the requesting client application’s identity.

The “scopes” claim shall then be used to validate that the part of the device model being accessed is within scope as defined in Clause 12.

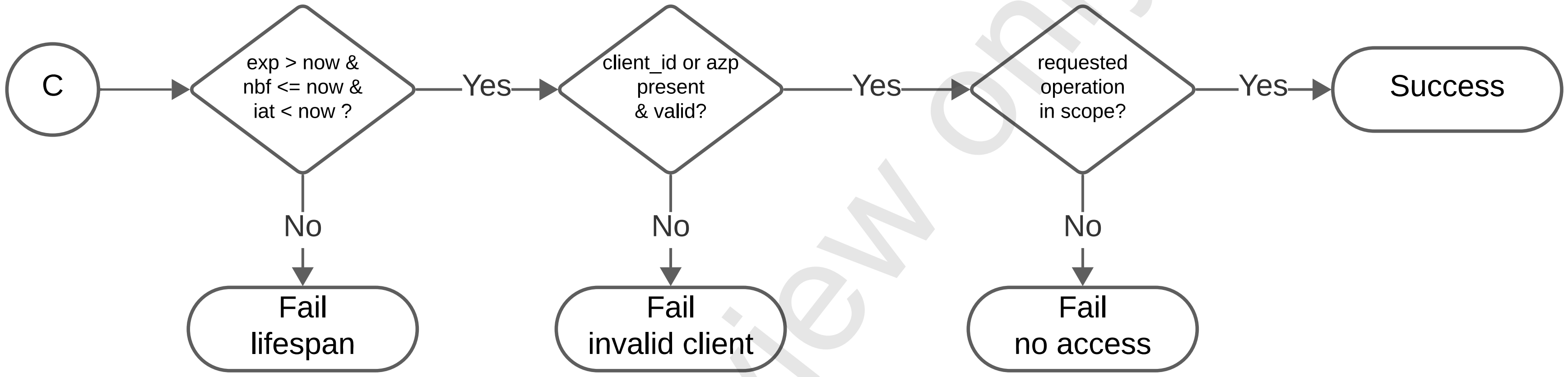
C
exp > now & nbf <= now & iat < now ?
Yes
client_id or azp present & valid?
Yes
requested operation in scope?
Yes
Success
No
Fail lifespan
No
Fail invalid client
No
Fail no access

Figure 11 — Timebox and Access Validation

11.2 Practical PEP Implementations (Informative)

In practical systems, it is often convenient to distribute the work of the PEP.

With reference to Figure 6, the Application Gateway performs the task of validating the access token if:

- It was issued by the Authorization Server — it uses the server's JSON Web Key Set endpoint to obtain its public key(s), which it requires to validate the token's signature (iss claim).
- The token has not expired (nbf, exp, iat) claims.
- It is for the resources in question (aud claim).

Envoy is one App Gateway that can perform these functions.

Note that this has no relation to the scopes claim. If the token is validated, the App Gateway relays the request to the Device where it first encounters the Connection Manager. The Connection Manager can perform the next level of token validation using the scopes claim.

The device model's parameter and command objects have an access scope associated with them and a read-only flag. The Connection Manager can:

- Verify that the accessed object's access scope is included in the token's scopes claim.
- If the access is a write access, verify that the read-only flag is false.

After passing these checks, the Connection Manager relays the token to the Business Logic for further specialized processing that is not appropriate to attempt to standardize. If the Device is an Asset Manager, for example, the rules for accessing the assets could be complex and dynamic.

11.3 Network Segmentation with Application Gateways (Informative)

The standard requires only that the API being used is included in the aud claim. But it is possible, and useful, to include other information that can be processed by App Gateways.

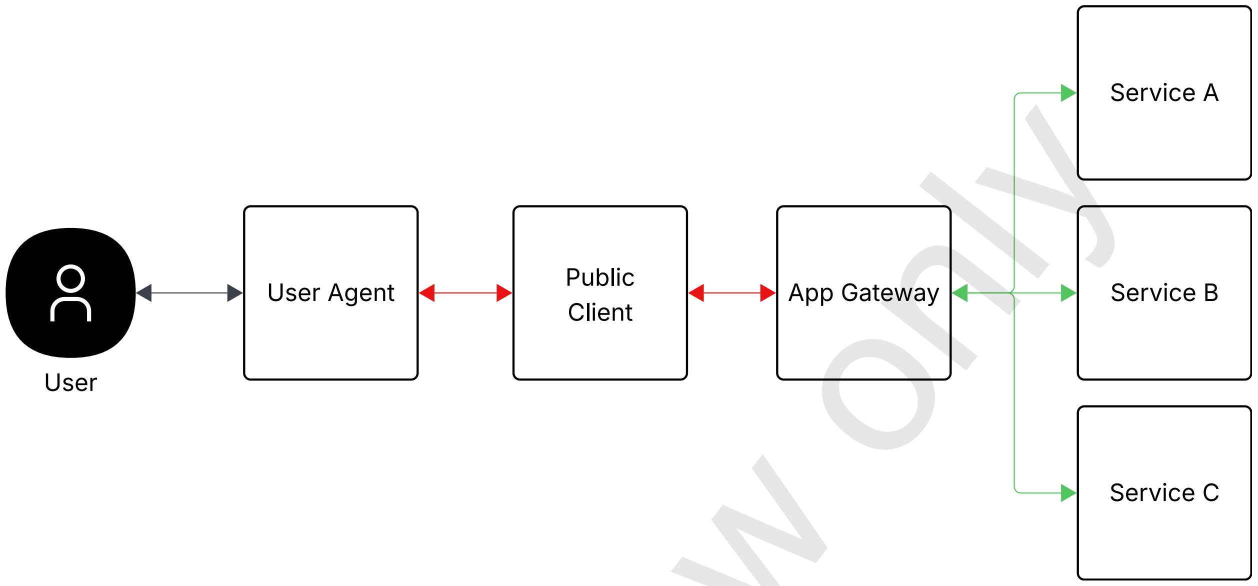

Figure 12 — App Gateway for TLS Termination and Routing

Figure 12 above illustrates a common use for App Gateways: TLS termination which enables the communications between the Services and the Gateway to be in clear text.

It also provides a routing service, which enables multiple services to share a (possibly public) domain name and IP address. The App Gateway routes messages to hostname service-a.example.com to Service A, and those with hostname service-b.example.com to Service B.

Gateway API is one package that can accomplish this, as can the load balancers provided by cloud hosting providers.

NOTE Ingress NGINX is to be retired in March 2026, i.e., there will be no new updates. SIG Network and the Security Response Committee recommend that all Ingress NGINX users immediately migrate to Gateway API or another Ingress controller. See Ingress NGINX Retirement: What You Need to Know | Kubernetes for additional information.

Another use of App Gateways is to segment the enterprise infrastructure to provide fine-grained access control.

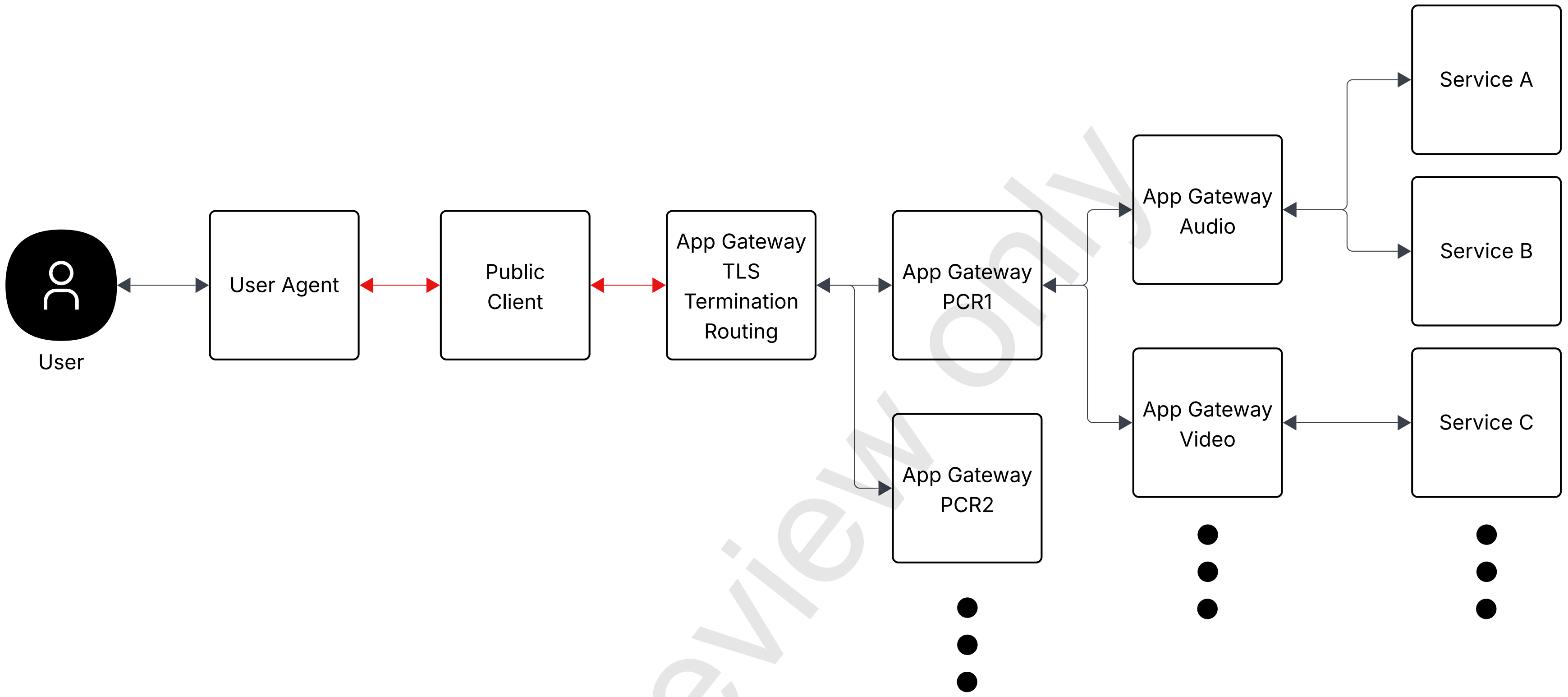

Figure 13 — Network Segmentation using the aud Claim

Figure 13 above illustrates how App Gateways such as Envoy can be used to provide fine-grained network segmentation.

In a case in which it is desirable to limit a user to accessing the audio deck in PCR1, by including the strings “PCR1” and “Audio”, the aud claim would enable the App Gateways to pass/deny access by extending the logic used to validate the token, as shown in Figure 13 above.

12 Interpretation of Catena-Scopes

12.1 Claims and Scopes

The Catena-scopes claim shall encode the access scopes enabled for the claim.

There are four standard scopes defined within the Catena Interface standard (ST2138-10): monitor, operate, configure, and administer.

PEPs shall interpret the Catena-scopes claim as follows:

- Claim is absent, no access. Neither read nor write permissions exist for this scope.
- Claim is present with no declarations. Read-only permission is enabled for this scope.
- Claim is present with ":w" declaration. Read and write permissions are enabled for this scope, unless the object being accessed is a parameter with its read_only flag set.

Examples:

"Catena-scopes": ["st2138:mon", "st2138:op:w", "st2138:cfg:w"] translates to:

- All objects within the *monitor* scope can be read.
- All objects within the *operate* scope can be read and written (unless the read_only flag is set).
- All objects within the *configure* scope can be read and written (unless the read_only flag is set).
- No objects within the *administer* scope can be accessed because that scope is not present in the claim.

12.2 Validating JWS and Supported Algorithms

In production, JWTs (as specified in IETF RFC 7519) shall be signed using the ES256 algorithm, as defined in IETF RFC 7519. The elliptic curve used shall be NIST P-256, aka prime256v1, as defined in NIST SP 800-186. In development and test environments, it is permissible to use unsigned tokens.

JWS validation tools as specified in IETF RFC 8725 JSON Web Token Best Current Practices shall be used.

12.3 Preserving Quality of Service (Informative)

The approach of "every request gets validated" is one of the tenets of Zero Trust. However, doing so under some uses of the OAuth2 framework envisaged here requires interaction with an Authorization Server.

In large-scale production operations, it is likely that many devices will interact with such a server and require appropriate dimensioning and design of the network and compute infrastructure.

Production operations can be significantly more "chatty" than other workloads. Consider that, when an operator uses a T-bar to make a transition, or adjusts an audio mix by moving faders, they could generate thousands of update requests per second.

It is thought highly likely that the network traffic between a client, PEP, resource server, and Authorization server generated by the "validate every request" approach would cause enough jitter in the timing of fader/transition updates being received and processed by the Resource Server to undermine artistic intent.

This consideration informed the decision to use JWS tokens (as specified in IETF RFC 7515), which do not require a round trip to the Authorization server.

12.4 Token Refresh and Revocation (Informative)

Some of the RPCs and REST calls are long-lived. An example is the Connect RPC/REST endpoint, which forms a connection between the server and the client for the server to provide updates.

With typical token lifespans being 3 minutes, it is necessary for a client to refresh the connection before its expiry.

Therefore, the protocols include a TokenRefresh message which can be empty or contain data to specify why the token is being refreshed. The new token is in the normal place — the message metadata where the server can extract it, validate it in the normal way, and then update its security context (e.g., expiry counter on the Connect session). Defined by the schema at #/$defs/refresh_token_payload.

The RevokeAccess message instructs the device to revoke access for either a named user or all users. Defined by the schema at #/$defs/revoke_access_payload.

Bibliography

Ingress NGINX Retirement: What You Need to Know | Kubernetes