living draft-00
last edited on September 15, 2021

Correlated Authorization

Igor Zboran
izboran@gmail.com
September 15, 2021

Abstract

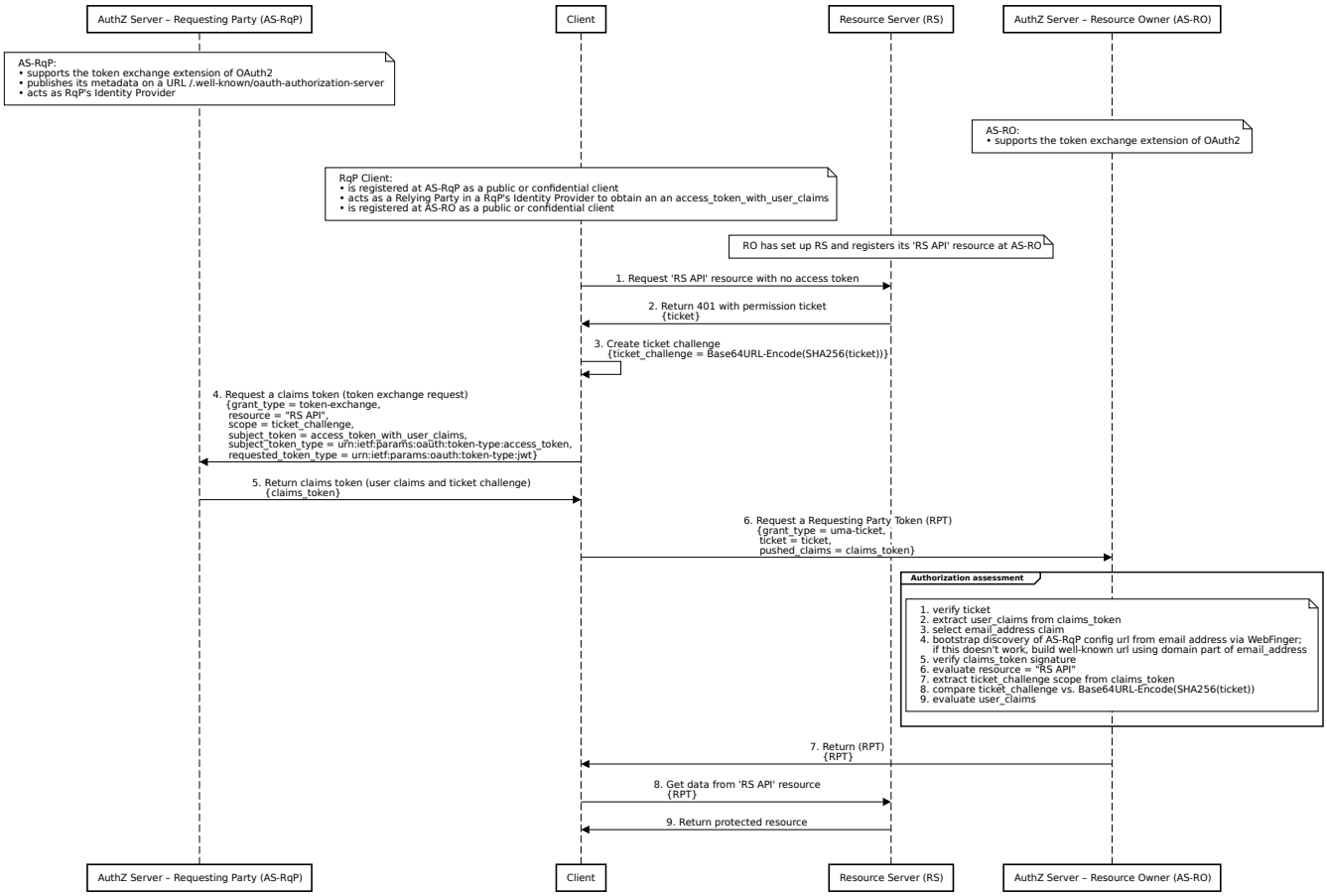
Correlated Authorization (CAZ) is a dual-authority authorization protocol built on top of User-Managed Access (UMA) and OAuth2 protocols that allows users (resource owners) to delegate access to other users (requesting parties) across security boundaries. The requesting party is responsible for creating the request, while the resource owner approves this request either when it is online or by creating a policy. The resource owner and the requesting party belong to different security domains administered by the respective authorities. This concept uses a permission ticket issued by the resource owner's authorization server as a correlation handle that binds the requesting party's claims to the authorization process. An email address is used as the unique requesting party identifier for cross-domain access control.

Sequence diagrams

There are two versions of the sequence diagram that describe the mechanism of the CAZ protocol. The first version represents the CAZ profile of the UMA protocol. The second version profiles the OAuth2 protocol. Both profiles rely on the token exchange extension of OAuth2, where an access token is used to obtain a claims token from the Security Token Service (STS) endpoint.

UMA profile

This diagram is in full compliance with the UMA specification.



Prerequisites:

- Both authorization servers support the OAuth 2.0 Token Exchange extension of OAuth2.
- The AS-RqP also acts as RqP's Identity Provider.
- The AS-RqP publishes its metadata on a URL /.well-known/oauth-authorization-server (alternatively on /.well-known/openid-configuration).
- The client is registered at the AS-RqP as a public or confidential client and acts as a Relying Party in a RqP's Identity Provider to obtain an access token with user claims.
- The client is registered at the AS-RO as a public or confidential client.
- The RO has set up the RS and registers its 'RS API' resource at the AS-RO according to the UMA Federated Authorization specification.

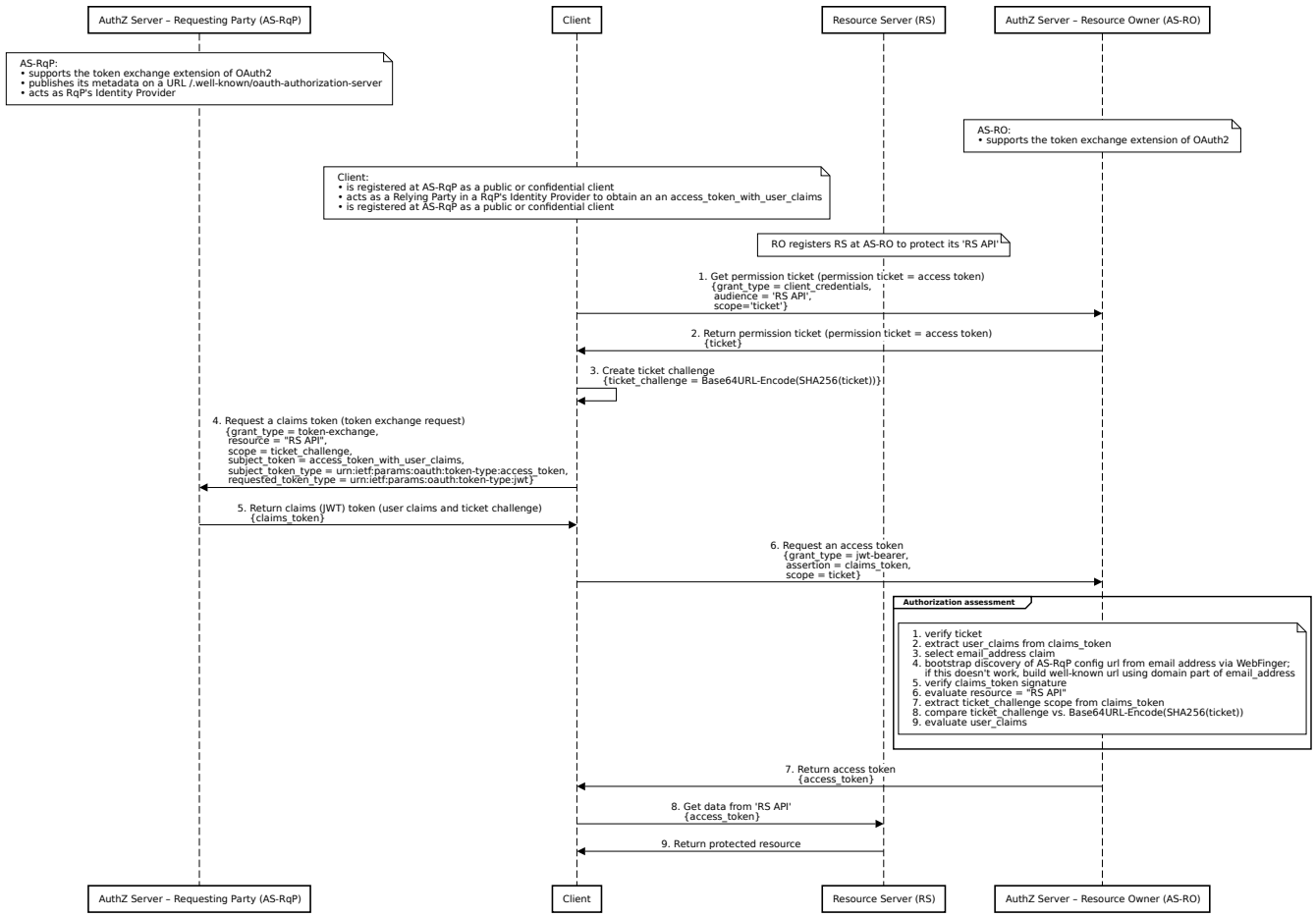
Steps:

- The RqP directs the client to access the 'RS API' resource with no access token.
- Without an access token, the RS will return HTTP code 401 (Unauthorized) with a permission ticket.

- The client creates a ticket challenge derived from the permission ticket using the following transformation ticket_challenge = Base64URL-Encode(SHA256(ticket)).
- At the AS-RqP the client requests a claims token by presenting the access token with user claims and the created ticket challenge.
- The AS-RqP returns the claims token.
- At the AS-RO the client requests an RPT by presenting the claims token and the permission ticket.
- After an authorization assessment, it is positive, the AS-RO returns RPT.
- With the valid RPT the client tries to access the 'RS API'.
- The RS validates the RPT, it is valid, the RS allow access the protected 'RS API' resource.

OAuth2 profile

This diagram represents a profile of the OAuth2 protocol and lacks some UMA features.



Prerequisites:

- Both authorization servers support the OAuth 2.0 Token Exchange extension of OAuth2.
- The AS-RqP also acts as RqP's Identity Provider.
- The AS-RqP publishes its metadata on a URL /.well-known/oauth-authorization-server (alternatively on /.well-known/openid-configuration).

- The client is registered at the AS-RqP as a public or confidential client and acts as a Relying Party in a RqP's Identity Provider to obtain an access token with user claims.
- The client is registered at the AS-RO as a public or confidential client.
- The RO registers the RS at the AS-RO to protect its RS API.

Steps:

- The RqP directs the client to get a permission ticket from the AS-RO to access the 'RS API' resource. The created permission ticket is an access token with a scope 'ticket'.
- The AS-RO returns the permission ticket.
- The client creates a ticket challenge derived from the permission ticket using the following transformation ticket_challenge = Base64URL-Encode(SHA256(ticket)).
- At the AS-RqP the client requests a claims token by presenting the access token with user claims and the created ticket challenge.
- The AS-RqP returns the claims (JWT) token.
- At the AS-RO the client requests an access token via a JWT grant type by presenting the claims (JWT) token and the permission ticket.
- After an authorization assessment, it is positive, the AS-RO returns the access token.
- With the valid access token the client tries to access the 'RS API'.
- The RS validates the access token, it is valid, the RS allow access the protected 'RS API' resource.

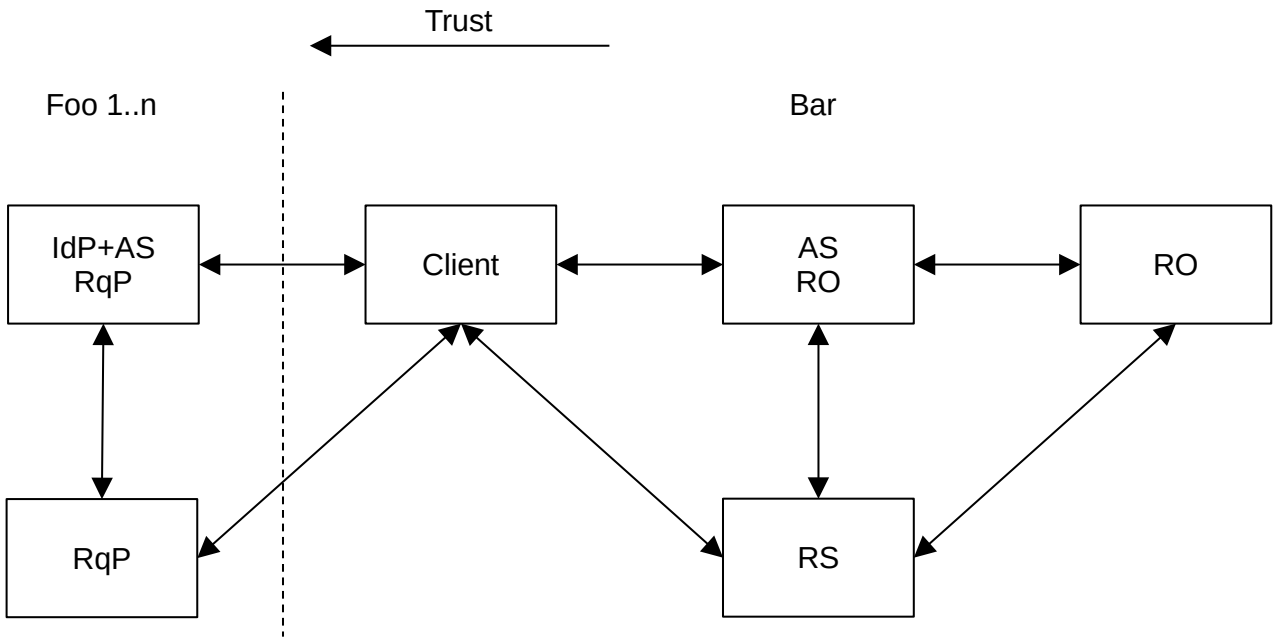
Authority boundaries, interactions and scenarios

The CAZ protocol allows us to indirectly (through the client) link identity providers with authorization services governed by different authorities that are not required to share information or collaborate.

The following scenarios demonstrate a system of trust between two authorities that allows the conveyance of identity information from identity providers to authorization services across security domains.

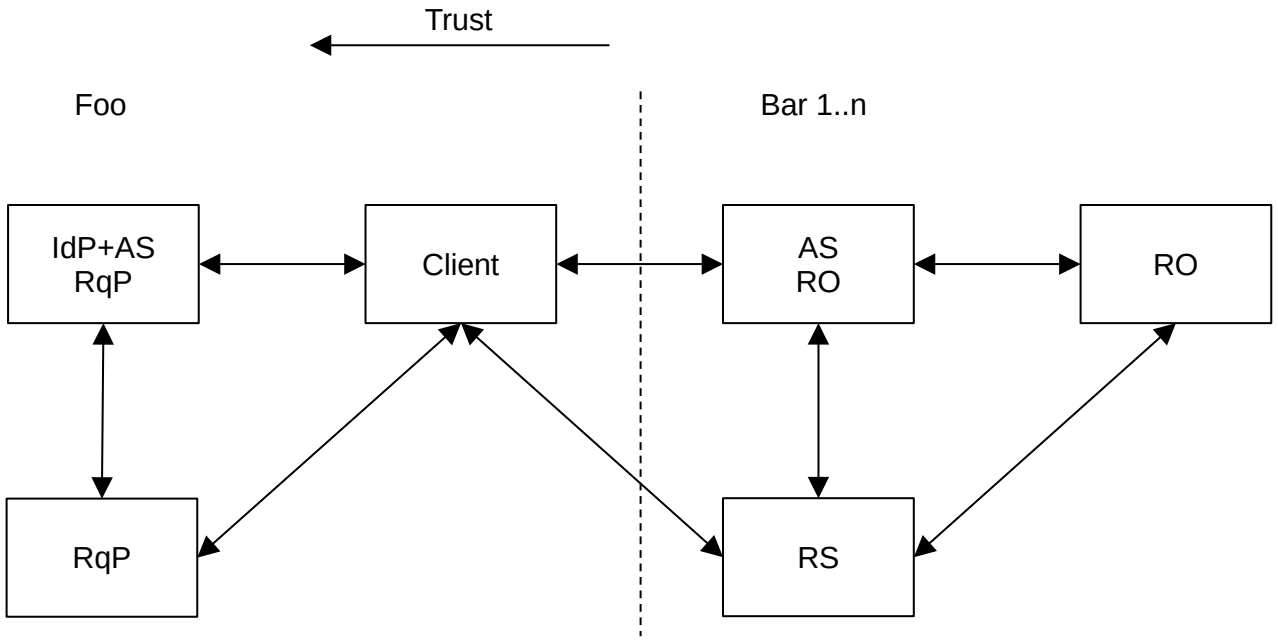
Identity federation scenario

This scenario allows to use multiple authoritative identity providers with a single authorization service. The client falls under the governance of the resource owner's respective authority.



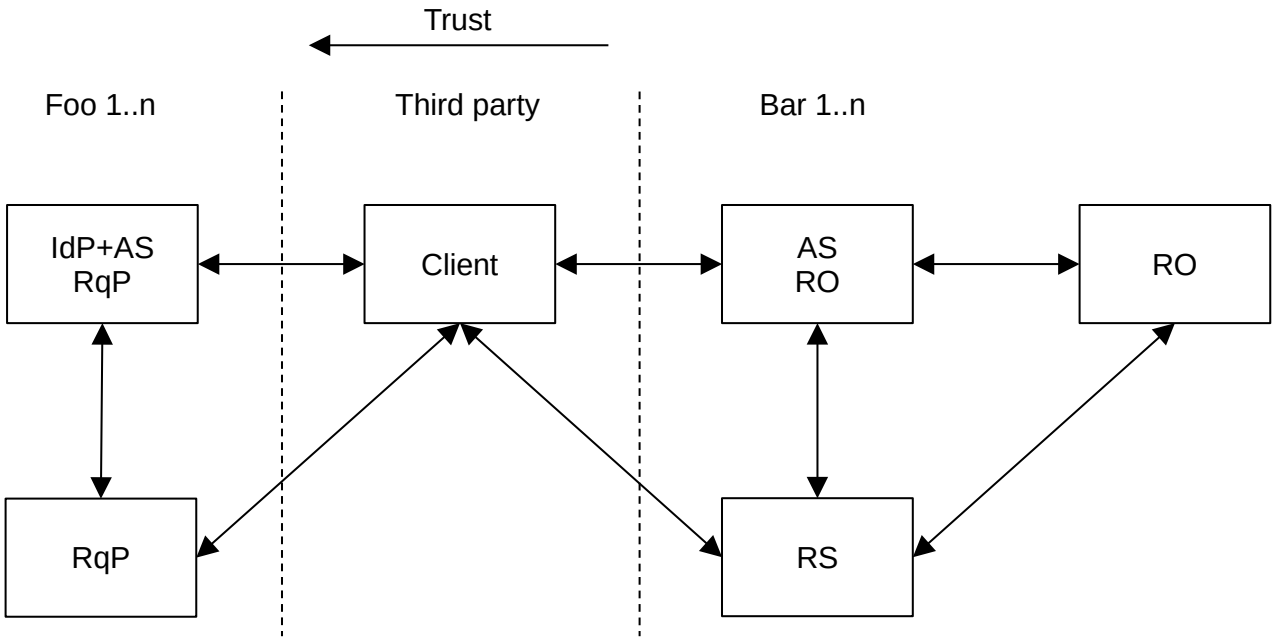
Federated authorization scenario

The federated authorization scenario shows the use of a single authoritative identity provider with multiple authorization services. The client falls under the governance of the requesting party's respective authority.



Combined federation scenario

As the name suggests, this scenario allows to use multiple authoritative identity providers with multiple authorization services. The client falls under the governance of a third-party authority.



Use cases

Healthcare and enterprise cross-domain services e.g. email, file sharing, instant messaging, tele-conferencing. Also, Fintech and Telco services.

Future Work

1. Consider a Correlated Authentication (CAN) protocol, where RS/AS acts as an external authoritative attribute/claims provider.
2. Employ the DPoP mechanism and create the permission ticket directly on the client to avoid the initial round trip to RS/AS.
3. Describe how the resource owner can use the CAZ protocol.
4. Consider using the CAZ mechanism to transfer digital/virtual assets in the form of transactions.

Acknowledgment

Credits go to WG - User-Managed Access.

About the Author

Igor Zboran is a mechanical engineer by education with professional experience as a software engineer, solutions architect and security architect. He'd like to transform his knowledge into a useful system or service that people would love to use.

Igor received Ing. degree in Mechanical Engineering from the University of Žilina, Slovakia in 1988. After graduating, he worked in several small private companies as a software developer. From 2008 to 2009, he provided expert advice to Prague City Hall IT department, Czech Republic as an external consultant. He invented a new decentralized Identity-Based Privacy (IBP) trusted model built around OAuth2 and OpenID Connect standards. Igor is a strong proponent of open source software and open standards.

References

[1] OpenID Connect (OIDC) https://openid.net/connect/
[2] User-Managed Access (UMA) https://en.wikipedia.org/wiki/User-Managed_Access
[3] WG - User Managed Access https://kantarainitiative.org/confluence/display/uma/Home
[4] OAuth 2.0 Token Exchange https://datatracker.ietf.org/doc/html/rfc8693