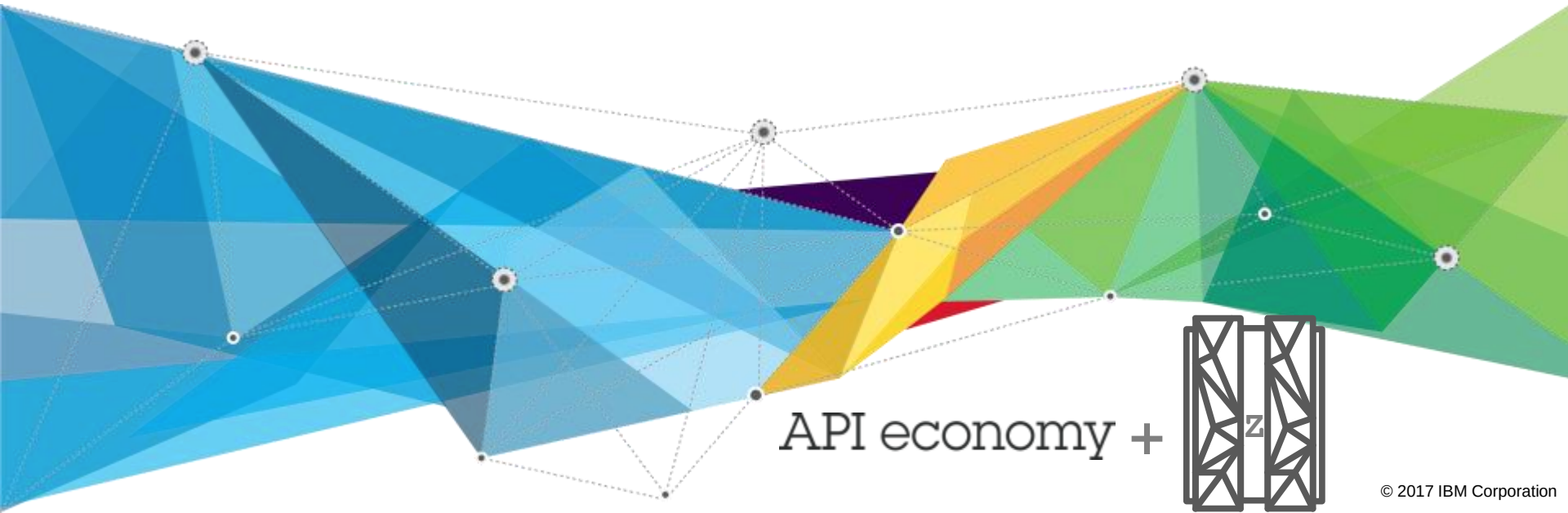
z/OS Connect EE Security Considerations

Nigel Williams nigel_williams@uk.ibm.com

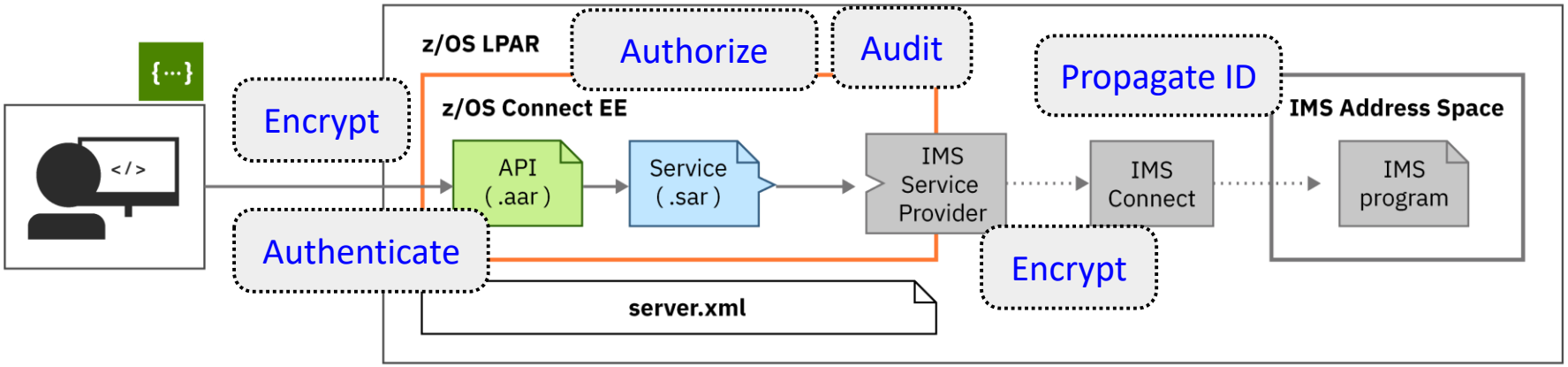


Contents

- Introduction
- API provider security
  - Authentication
  - Authorization
  - Audit
  - Encryption
  - Flowing identities to back end systems
- API requester security
  - What's different?
- More information

Recap - using z/OS Connect EE with IMS

GET https://mybank.com/myBank/accounts/{accountId}

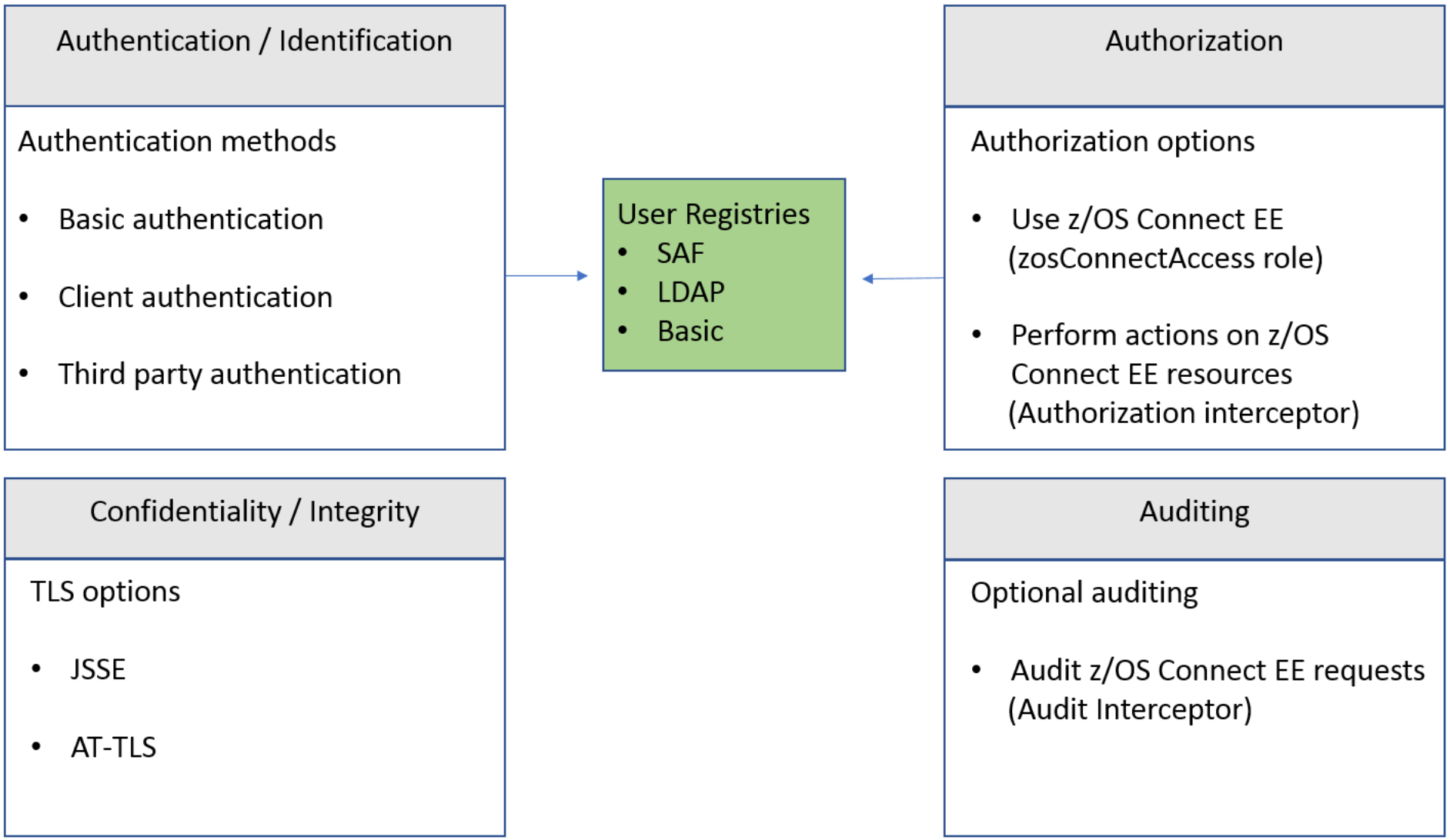


Create and deploy services and APIs using the API Toolkit

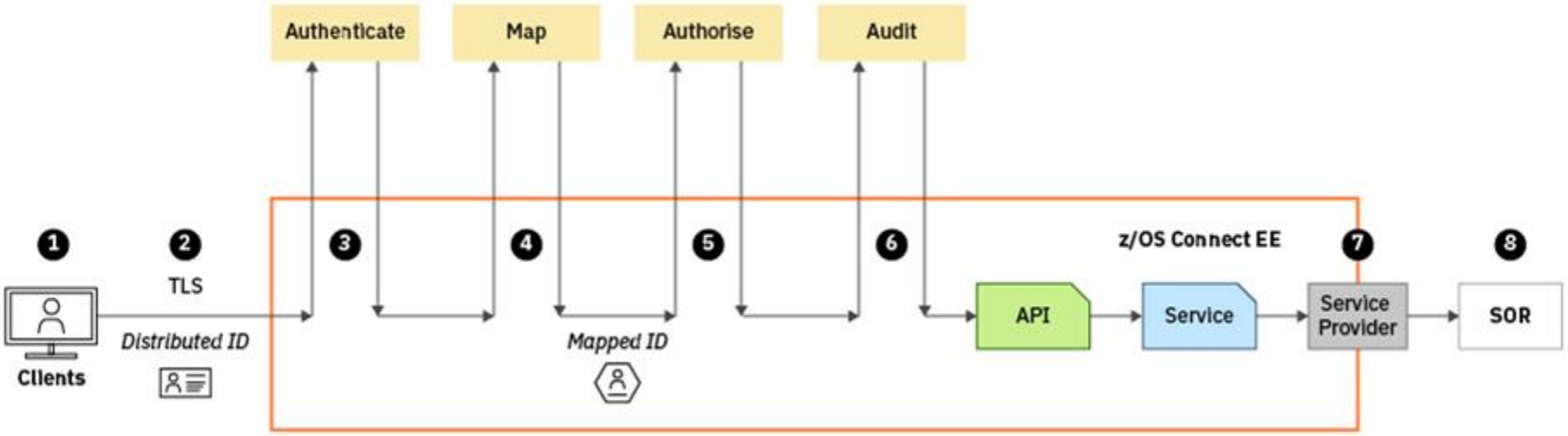
Configure the connection to IMS through `ims-connections.xml` and `ims-interactions.xml` in the IMS service registry.

ibm.biz/zosconnect-scenarios

z/OS Connect EE security options



API provider security flow



1. Client credentials
2. Secure connection
3. Authenticate the client
4. Map authenticated identity to a user ID
5. Authorize the authenticated user ID
6. Audit the request
7. Secure connection to SoR
8. Propagate identity to SoR

Security is configured in server.xml

Excerpt from server.xml

```
<featureManager>
     <feature>zosconnect:zosconnect-2.0</feature>
     <feature>imsmobile:imsmobile-2.0</feature>
     <feature>appSecurity-2.0</feature>
     <feature>zosSecurity-1.0</feature>
</featureManager>


<sslDefault sslRef="DefaultSSLSettings"/>
<ssl id="DefaultSSLSettings"
     clientAuthentication="true"
     keyStoreRef="CellDefaultKeyStore"/>


<keyStore id="CellDefaultKeyStore" fileBased="false"
     location="safkeyring:///Keyring.LIBERTY"
     password="password" readOnly="true"
     type="JCERACFKS"/>

<webAppSecurity allowFailOverToBasicAuth="true" />

<safRegistry id="saf"/>
<safAuthorization id="safAuth"/>
<safCredentials profilePrefix="BBGZDFLT"
                    unauthenticatedUser="WSGUEST"/>
```

Features

TLS/SSL

Key Store

Authentication

Authorization

Common challenges

- **End-to-end security** is hampered by the issue of how to provide secure access between middleware components that use disparate security technologies e.g registries

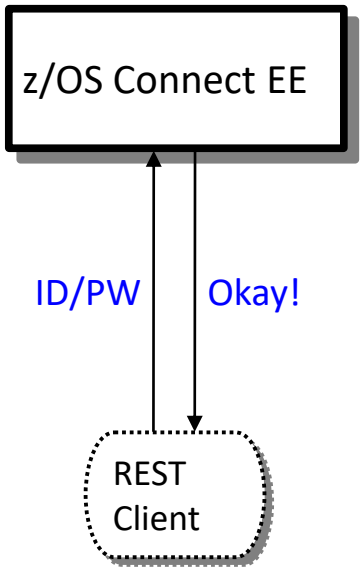
- z/OS Connect security is implemented in many products including z/OS Connect, Liberty z/OS, SAF/RACF, CICS, IMS, DB2 …

- Often security is at odds with **performance**, because the most secure techniques often involve the most processing overhead especially if not configured optimally

Authentication

Several different ways this can be accomplished:

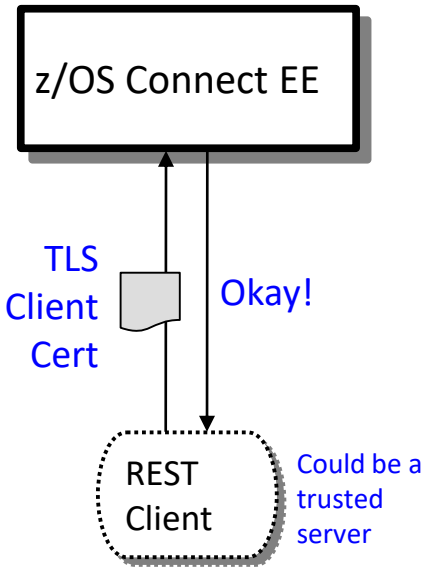
Basic Authentication



Server prompts for ID/PW
Client supplies ID/PW
Server checks registry:

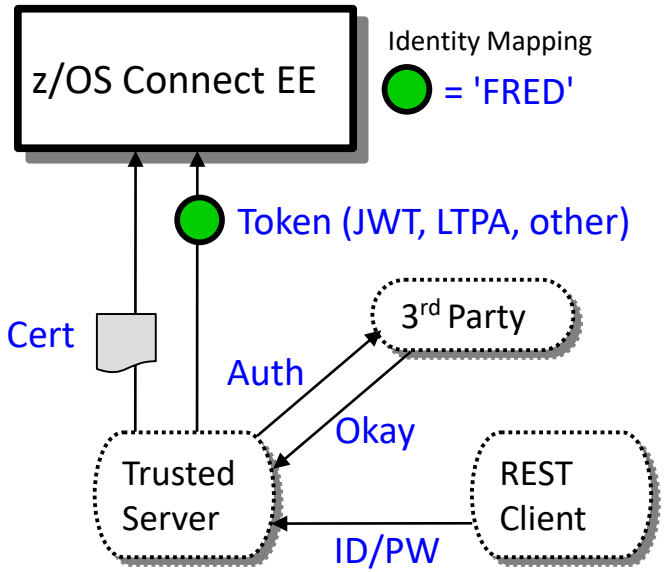
- Basic (server.xml)
- LDAP
- SAF

Client Certificate



Server prompts for cert.
Client supplies certificate
Server validates cert and maps to an identity

Third Party Authentication



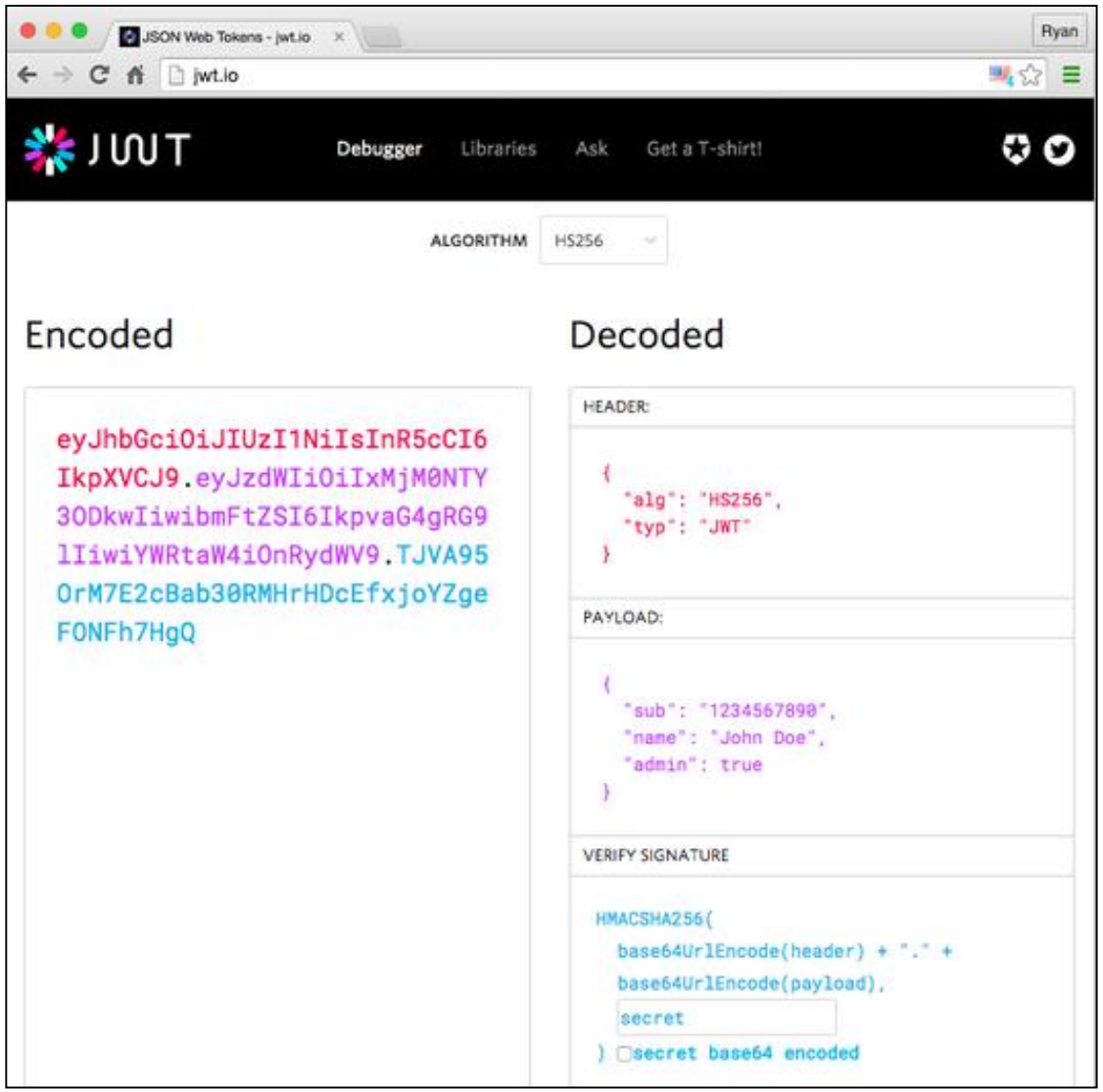
Client authenticates to 3rd party sever
Client receives a trusted 3rd party token
Token flows to Liberty z/OS across trusted connection and is mapped to an identity

Security token types supported by z/OS Connect EE

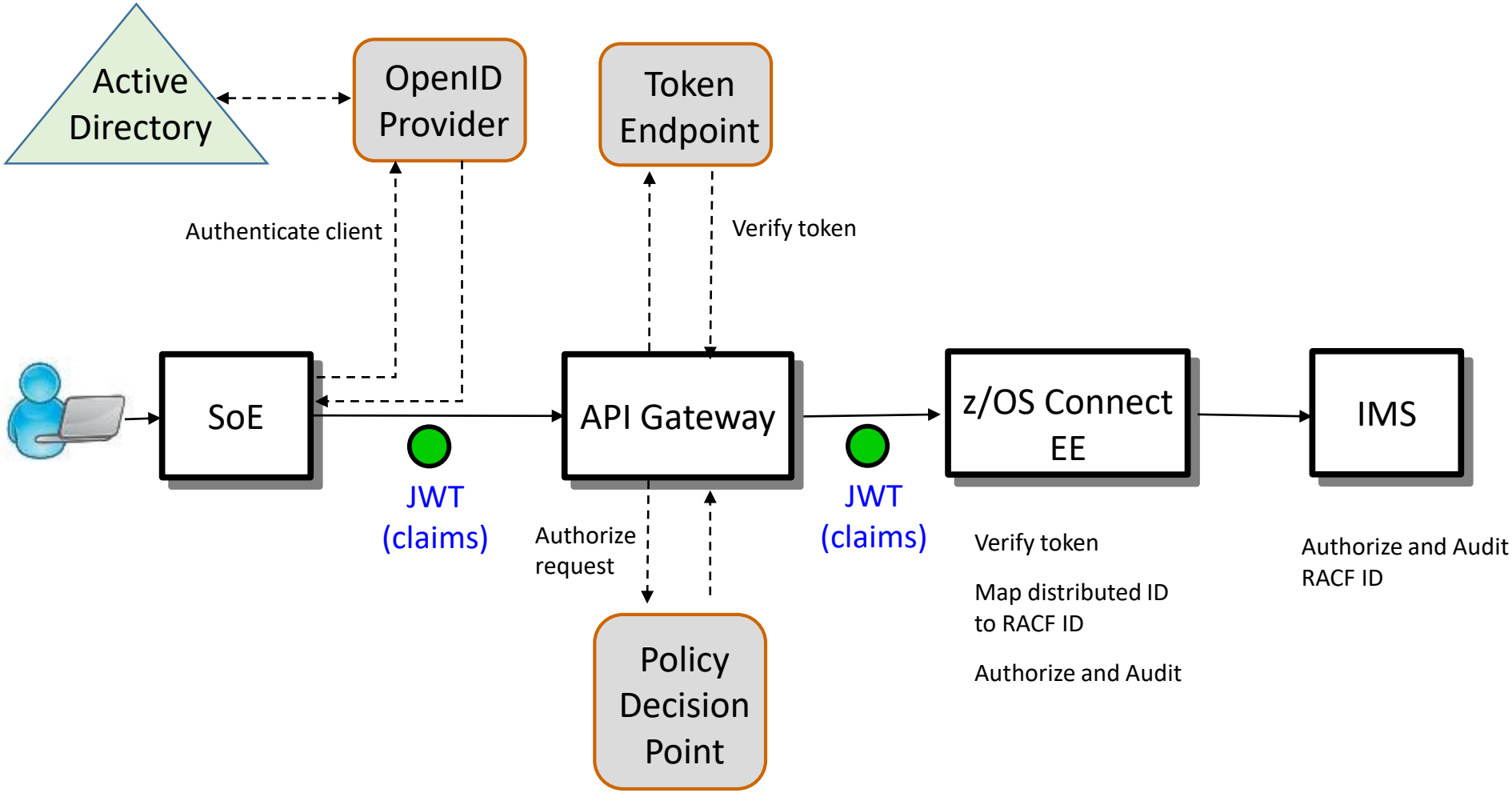
Token type	How used	Pros	Cons
LTPA	Authentication technology used in IBM WebSphere	• Easy to use with WebSphere and DataPower	• IBM Proprietary token
SAML	XML-based security token and set of profiles	• Token includes user id and claims • Used widely with Web services	• Tokens can be heavy to process • No refresh token
OAuth 2.0 opaque access token	Facilitates the authorization of one site to access and use information related to the user's account on another site	• Used widely for SoE applications e.g with Google, Facebook, Microsoft, Twitter …	• Opaque tokens need introspection endpoint to validate token
JWT	JSON security token format	• More compact than SAML • Self-contained - not necessary for the recipient to call a server to validate the token	

JWT (JSON Web Token)

- JWT is a compact way of representing claims that are to be transferred between two parties
- Normally transmitted via HTTP header
- Consists of three parts
  - Header
  - Payload
  - Signature



Example end to end security flow




```
RACMAP  [ID(mapuser) | USERDIDFILTER(NAME('distributed-identity-filter'))]
        REGISTRY(NAME('distributed-identity-registryname'))
        [WITHLABEL('Mapping RACF ID')]
```

Configuring authentication with JWT

z/OS Connect EE can perform user authentication with JWT using the support that is provided by the *openidConnectClient-1.0* feature. The ***<openidConnectClient>*** element is used to accept a JWT token as an authentication token

```
<openidConnectClient id="RS" clientId="RS-JWT-ZCEE" inboundPropagation="required"
    signatureAlgorithm="RS256" trustStoreRef="JWTTrustStore"
    trustAliasName="JWTapicSign" userIdentityToCreateSubject="sub"
    mapIdentityToRegistryUser="false" issuerIdentifier="idg"
    authnSessionDisabled="true" audiences="zCEE"/>
```

- ***inboundPropagation*** is set to required to allow z/OS Connect EE to use the received JWT as an authentication token
- ***signatureAlgorithm*** specifies the algorithm to be used to verify the JWT signature
- ***trustStoreRef*** specifies the name of the keystore element that defines the location of the validating certificate
- ***trustAliasName*** gives the alias or label of the certificate to be used for signature validation
- ***userIdentityToCreateSubject*** indicates the claim to use to create the user subject
- ***mapIdentityToRegistryUser*** indicates whether to map the retrieved identity to the registry user
- ***issuerIdentifier*** defines the expected issuer
- ***authnSessionDisabled*** indicates whether a WebSphere custom cookie should be generated for the session
- ***audiences*** defines a list of target audiences

See Dev Center article "Using a JWT with z/OS Connect EE" for full description of scenario

z/OS Connect EE – authentication filter

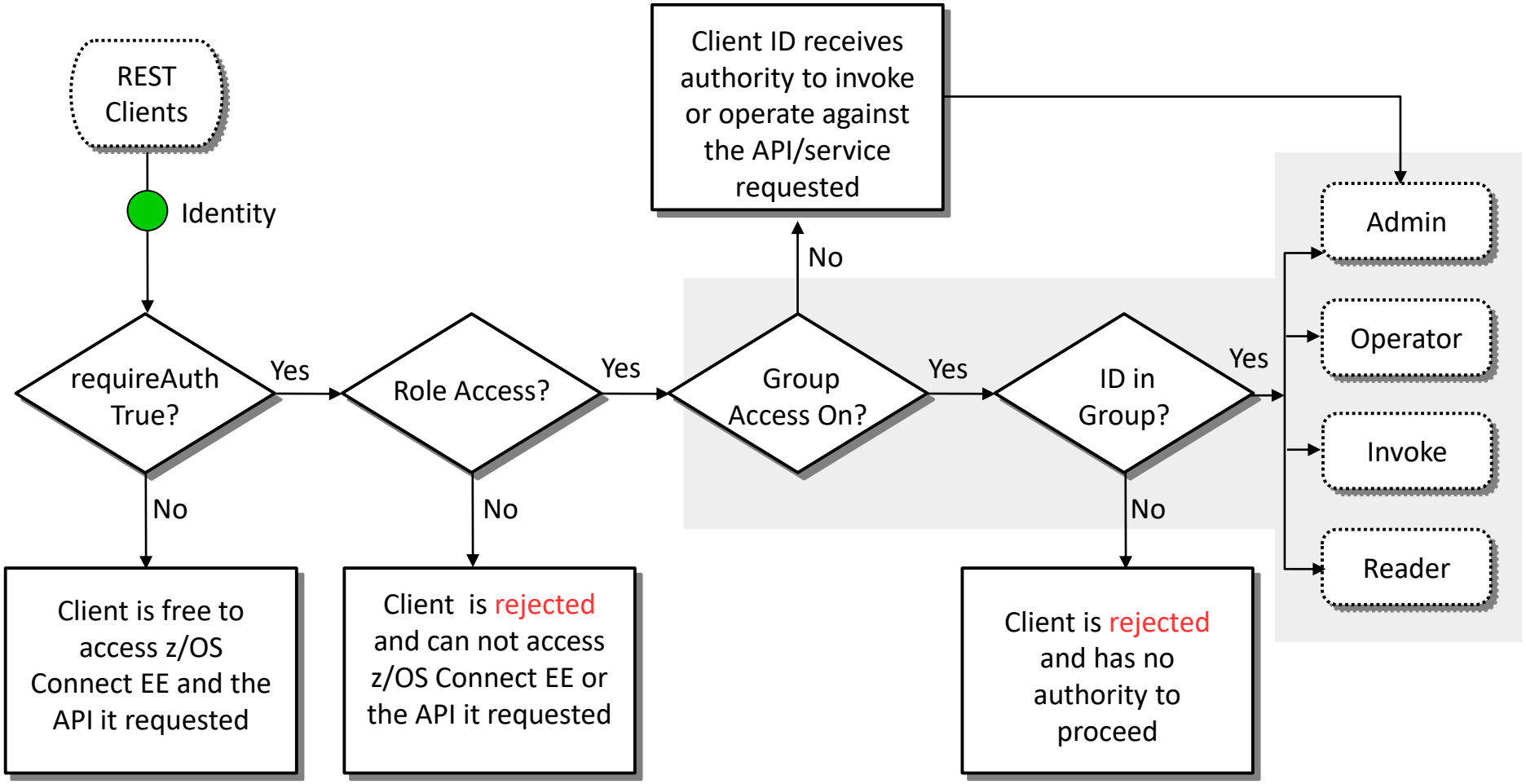
Authentication filters define filter criteria that determine whether certain requests are processed by certain providers, such as OpenID Connect, for authentication.

```
<authFilter id="AuthFilterCatalogJWT">
 <requestUrl id="CatalogJWTUrls" urlPattern="/catalogManager"
        matchType="contains"/>
 <remoteAddress id="APIConnectGateway" ip="10.3.20.92" matchType="contains"/>
</authFilter>
```

Some alternative filter types

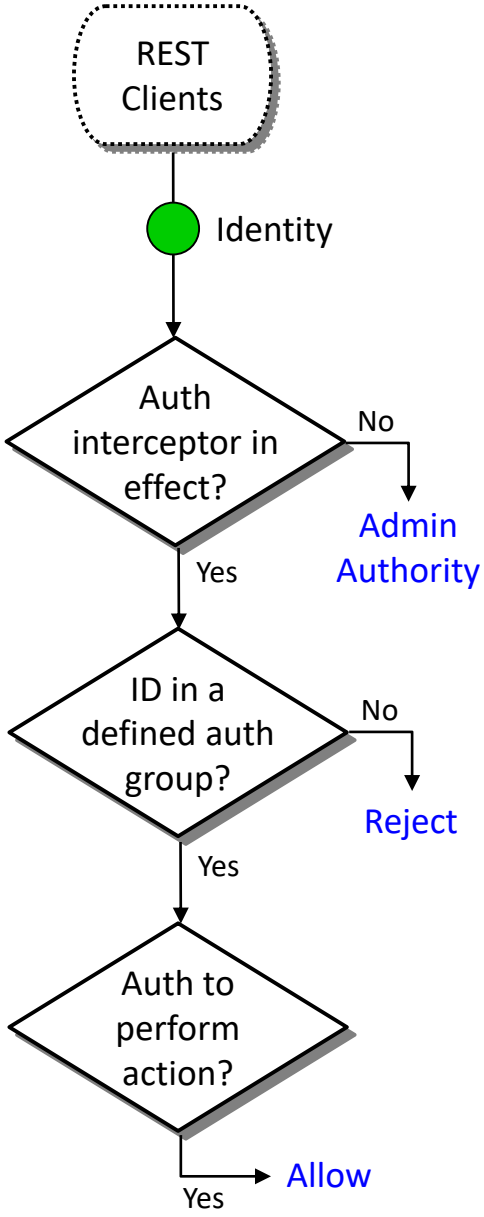
- The ***host*** element is compared against the "Host" HTTP request header, which identifies the target host name of the request
- The **userAgent** element is compared against the "User-Agent" HTTP request header, which identifies the client software that is used by the originating request

Security flow with z/OS Connect EE



PERMIT ***profilePrefix*.zos.connect.access.roles.zosConnectAccess** CLASS(EJBROLE) ID(**EMPLOY1**) ACCESS(READ)

Configuring authorization




```
<zosconnect_zosConnectManager
    globalAdminGroup="GMADMIN"
    globalInvokeGroup="GMINVOKE"
    globalInterceptorsRef="interceptorList_g" />

<zosconnect_zosConnectInterceptors
   id="interceptorList_g"
   interceptorRef="auth"/>

<zosconnect_authorizationInterceptor id="auth"/>
```

Definition of groups at the global level. Can also be defined at API and service level.

Pointer to interceptor list

An interceptor "list" that defines just the "auth" interceptor.

Element that defines authorization interceptor to z/OS Connect EE.

Define at global (shown here), API or service layer

Override, opt-out … many ways to configure this

See Techdoc WP102439 "Interceptor Scenarios" PDF for examples of variations on configuring interceptors at different levels in z/OS Connect.

Audit (SMF) Interceptor

The audit interceptor writes SMF 123.1 records. Below is an example of some of the information captured:

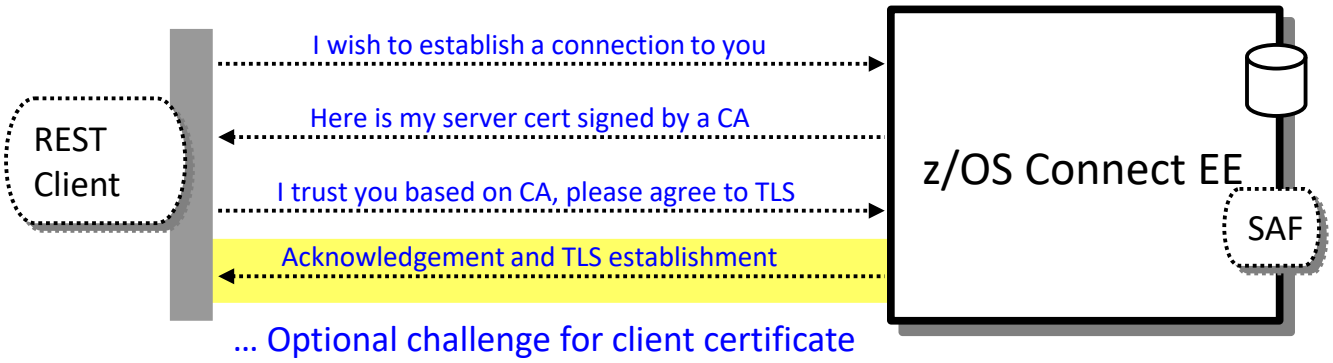
- System Name
- Sysplex Name
- Job Name
- Job Prefix
- Address Space Stoken

Server Identification Section

- Arrival Time
- Completion Time
- Target URI
- Input JSON Length
- Response JSON Length
- Method Name
- API or Service Name
- Userid
- Mapped user name

User Data Section

SSL/TLS connections



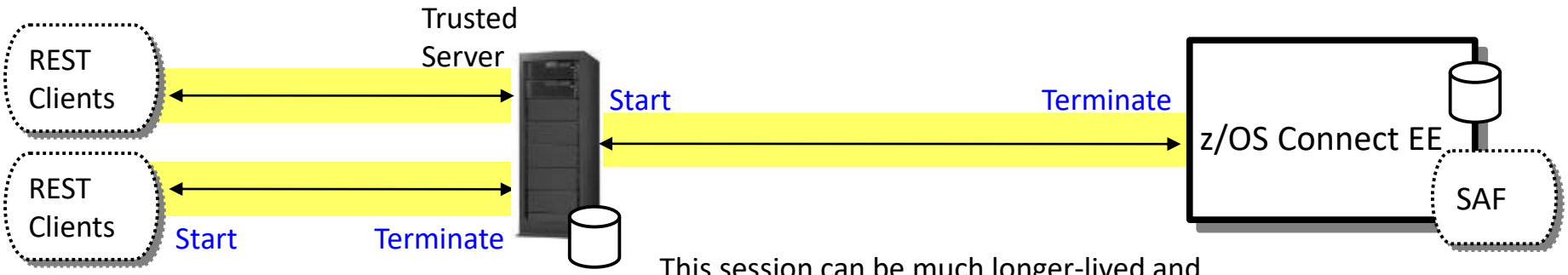
Java-based key/trust files

Easy to set up, but less control by security administrators

SAF keyrings

This is under the control of security administrators

Important to understand where the TLS sessions start and end:

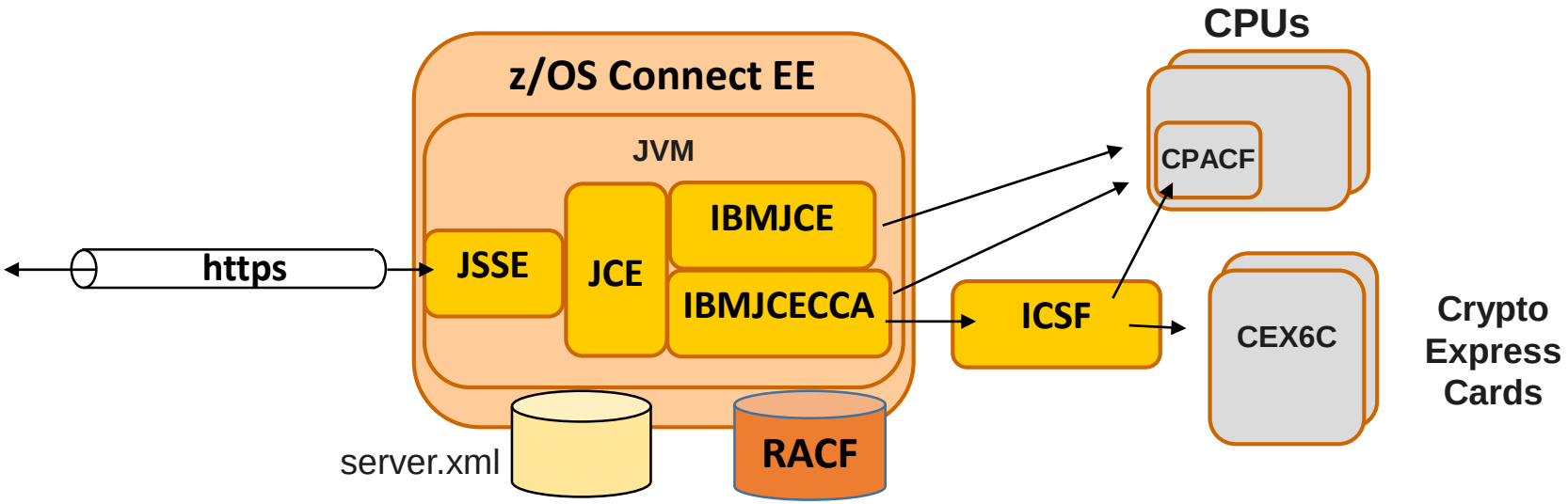


The client TLS sessions may come and go frequently. If that's the responsibility of a mid-tier trusted server, then the overhead of setup/teardown is there, not on the z/OS system

This session can be much longer-lived and thus less setup/teardown overhead

You can manage SAF-based certificates more easily here because potential clients are limited and known

z/OS Connect EE uses JSSE



- z/OS Connect EE support for SSL/TLS is based on **Liberty server** support
- **Java Secure Socket Extension** (JSSE) API provides framework and Java implementation of SSL and TLS protocols used by Liberty HTTPS support
- **Java Cryptography Extension** (JCE) is standard extension to the Java Platform that provides implementation for cryptographic services
- **IBM Java SDK** for z/OS provides two different JCE providers, **IBMJCE** and **IBMJCECCA**

Cyphers

- During the TLS handshake, the TLS protocol and data exchange cipher are negotiated
- Choice of cipher and key length has an impact on performance
- You can restrict the protocol (SSL or TLS) and ciphers to be used
- Example setting server.xml file

```
<ssl id="DefaultSSLSettings"
keyStoreRef="defaultKeyStore" sslProtocol="TLSv1.2"
enabledCiphers="TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384"/>
```

- This configures use of TLS 1.2 and two supported ciphers
- It is recommended to control what ciphers can be used in the server rather than the client

Persistent connections

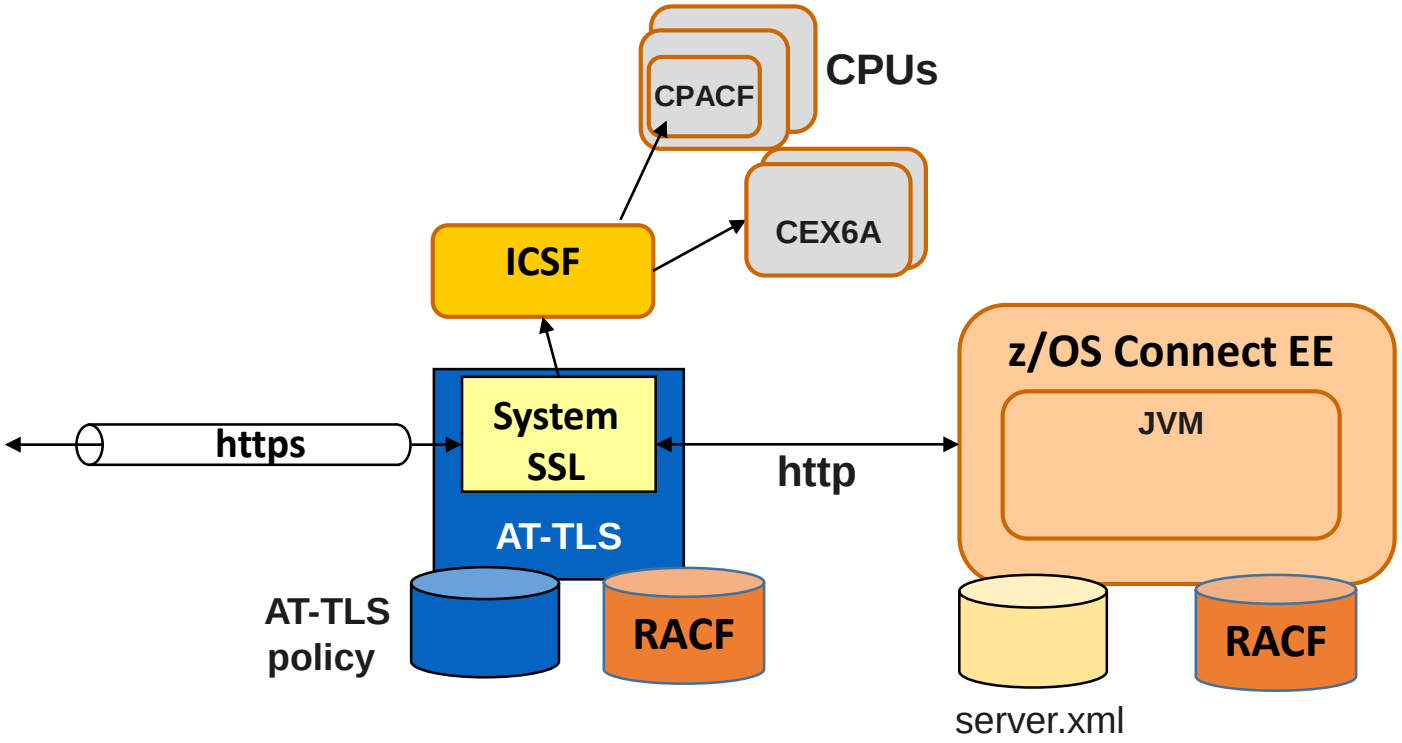
- Persistent connections can be used to avoid too many handshakes
- Configured by setting `the keepAliveEnabled` attribute on the httpOptions element to **true**
- Example setting server.xml file

```
<httpEndpoint host="*" httpPort="80" httpsPort="443"
id="defaultHttpEndpoint" httpOptionsRef="httpOpts"/>

<httpOptions id="httpOpts" keepAliveEnabled="true"
maxKeepAliveRequests="500" persistTimeout="1m"/>
```

- This sets the connection timeout to **1 minute** (default is 30 seconds) and sets the maximum number of persistent requests that are allowed on a single HTTP connection to **500**
- It is recommended to set a maximum number of persistent requests when connection workload balancing is configured
- It is also necessary to configure the client to support persistent connections

Using AT-TLS with z/OS Connect EE

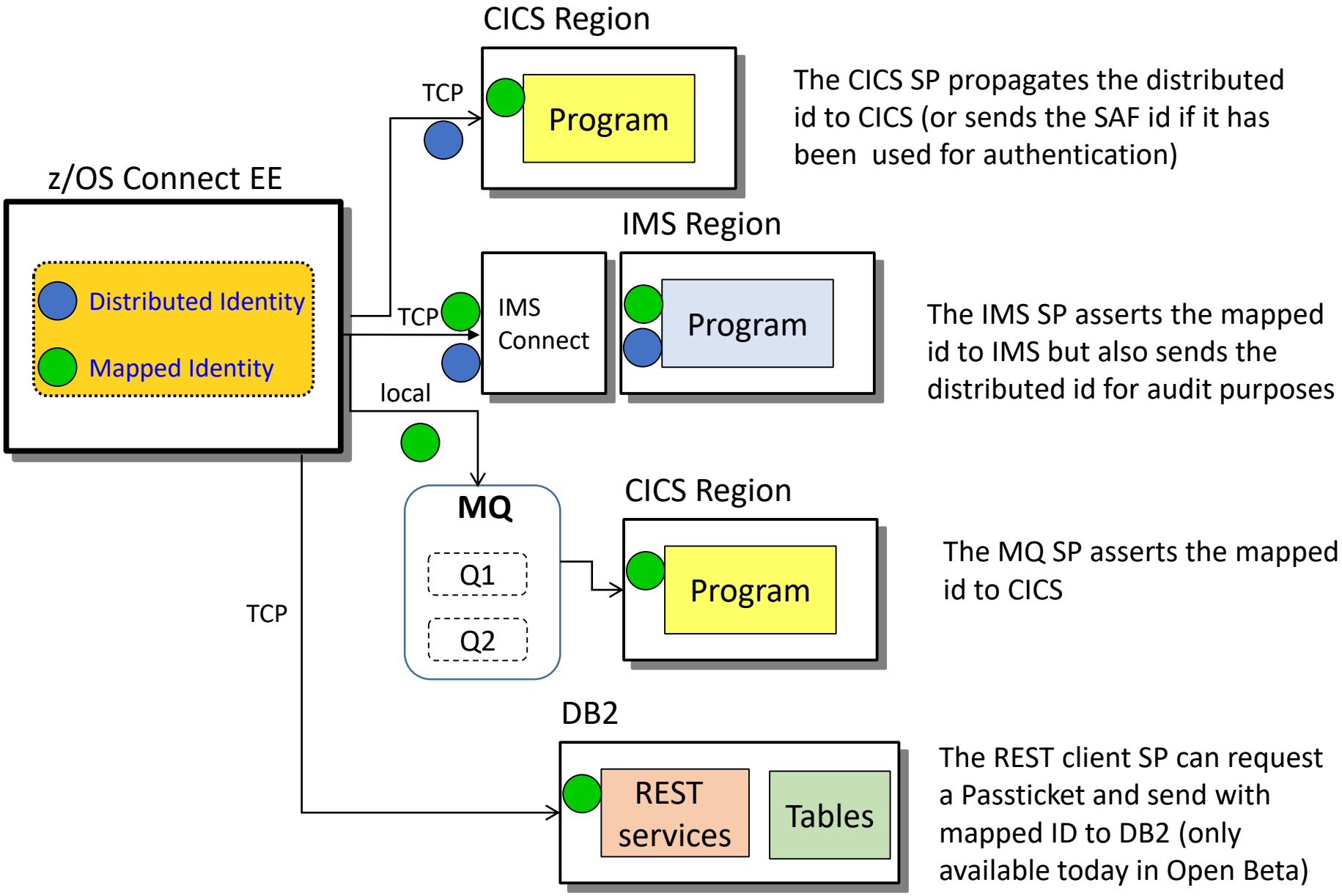


- **Application Transparent TLS** (AT-TLS) creates a secure session on behalf of z/OS Connect
- Only define http ports in server.xml (z/OS Connect does not know that TLS session exists)
- Define TLS protection for all applications (including z/OS Connect) in **AT-TLS policy**
- AT-TLS uses **System SSL** which exploits the CPACF and Crypto Express cards via ICSF

JSSE and AT-TLS comparison

Capability	Description	JSSE	AT-TLS
1-way SSL	Verification of z/OS Connect certificate by client	Yes	Yes
2-way SSL	Verification of client certificate by z/OS Connect	Yes	Yes
SSL client authentication	Use of client certificate for authentication	Yes	**No**
Support for requireSecure option on APIs	Requires that API requests are sent over HTTPS	Yes	**No**
Persistent connections	To reduce number of handshakes	Yes	Yes
Re-use of SSL session	To reduce number of full handshakes	Yes	Yes
Shared SSL sessions	To share SSL sessions across cluster of z/OS Connect instances	**No**	Yes
zIIP processing	Offload TLS processing to zIIP	Yes	**No**
CPACF	Offload symmetric encryption to CPACF	Yes	Yes
CEX6	Offload asymmetric operations to Crypto Express cards	Yes	Yes

Flowing an identity to the back end



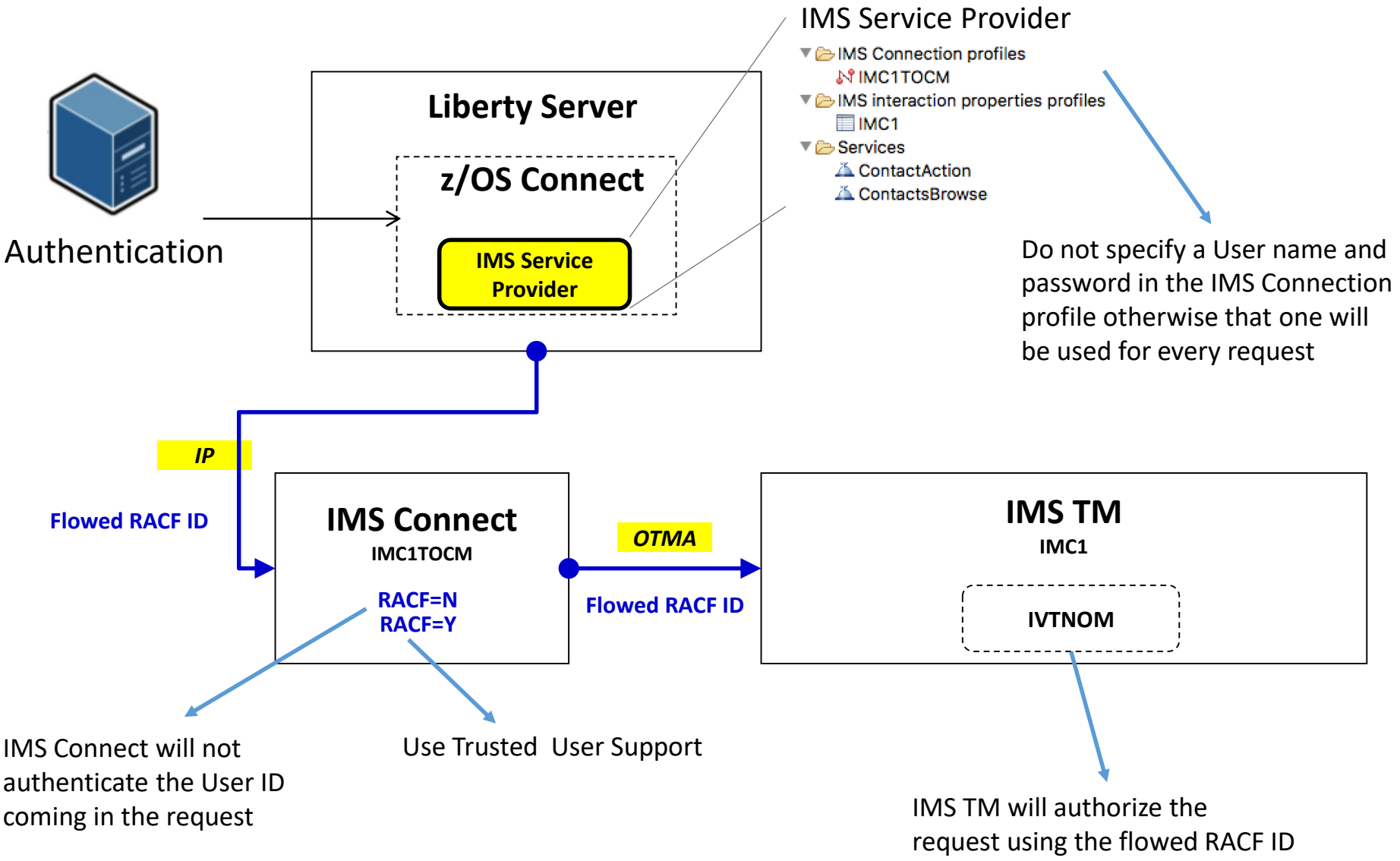
The CICS SP propagates the distributed id to CICS (or sends the SAF id if it has been used for authentication)

The IMS SP asserts the mapped id to IMS but also sends the distributed id for audit purposes

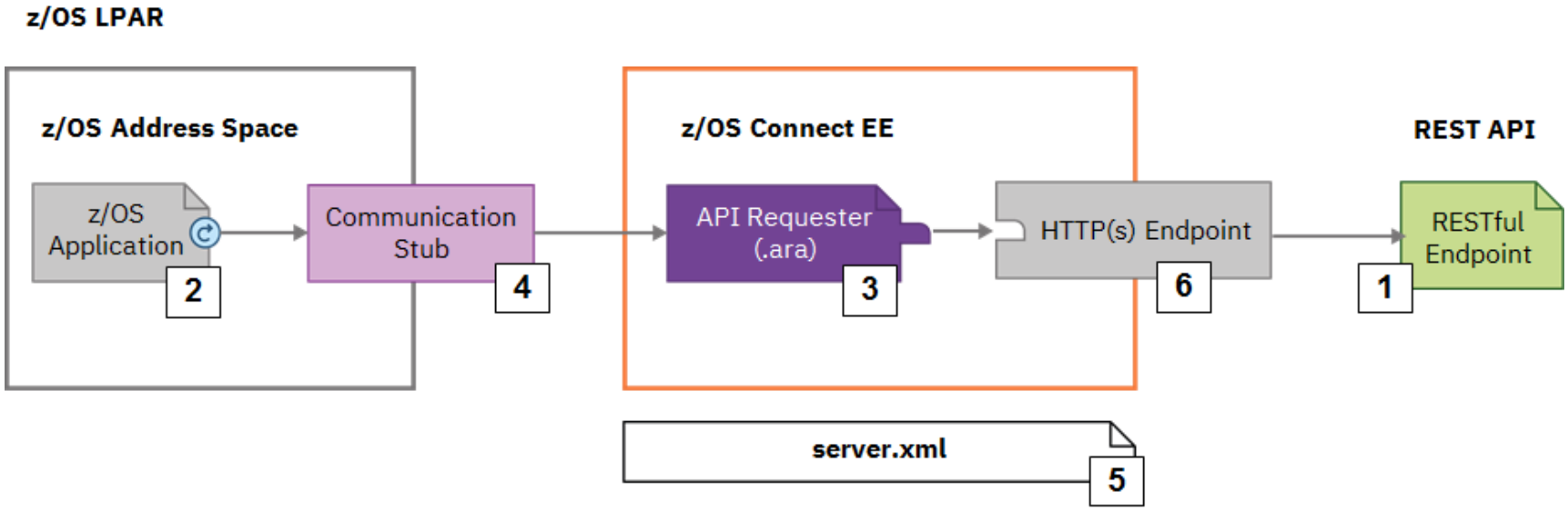
The MQ SP asserts the mapped id to CICS

The REST client SP can request a Passticket and send with mapped ID to DB2 (only available today in Open Beta)

Flowing a RACF ID to IMS

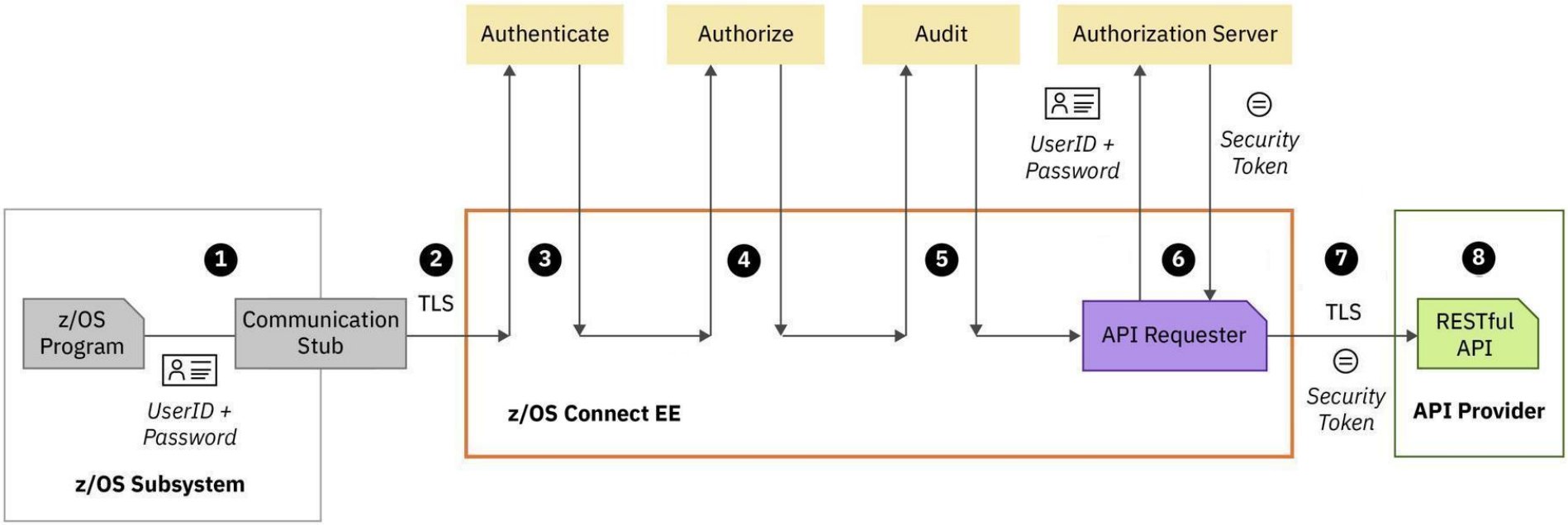


Recap – API requester



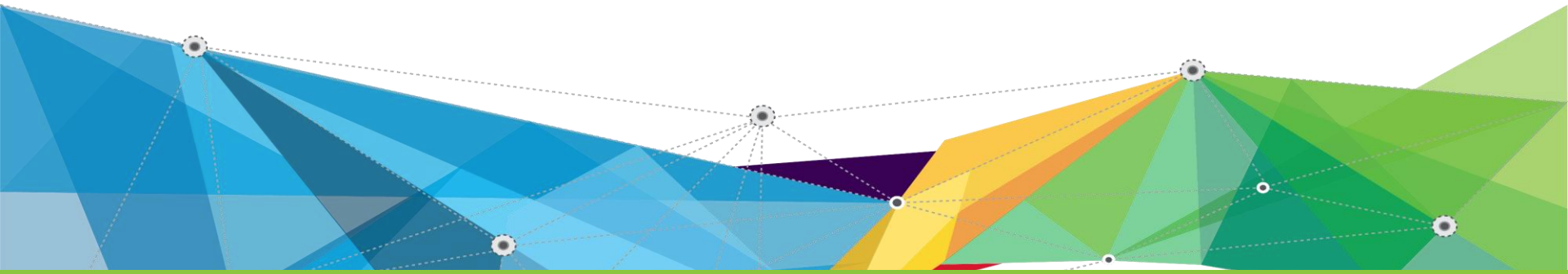
1. RESTful endpoint described in Swagger document
2. CICS application that needs to call the REST API (uses copybooks generated by the z/OS Connect EE build toolkit)
3. API requester archive (.ara) file generated by z/OS Connect EE build toolkit
4. z/OS Connect EE module that establishes an HTTP connection to z/OS Connect EE
5. z/OS Connect EE is configured in server.xml file
6. Security for RESTful endpoint is configured in server.xml
   - Basic authentication
   - Client authentication
   - API keys
   - OAuth 2.0
   - JWT

API requester security flow



1. z/OS program can provide user ID & password
2. Send request on secure connection
3. Authenticate the credentials
4. Authorize the authenticated user ID
5. Audit the request
6. Obtain token from authorization server
7. Secure connection to API provider with security token
8. RESTful API runs in API provider

Summary



Summary

- Define clear security requirements before deciding on a security design

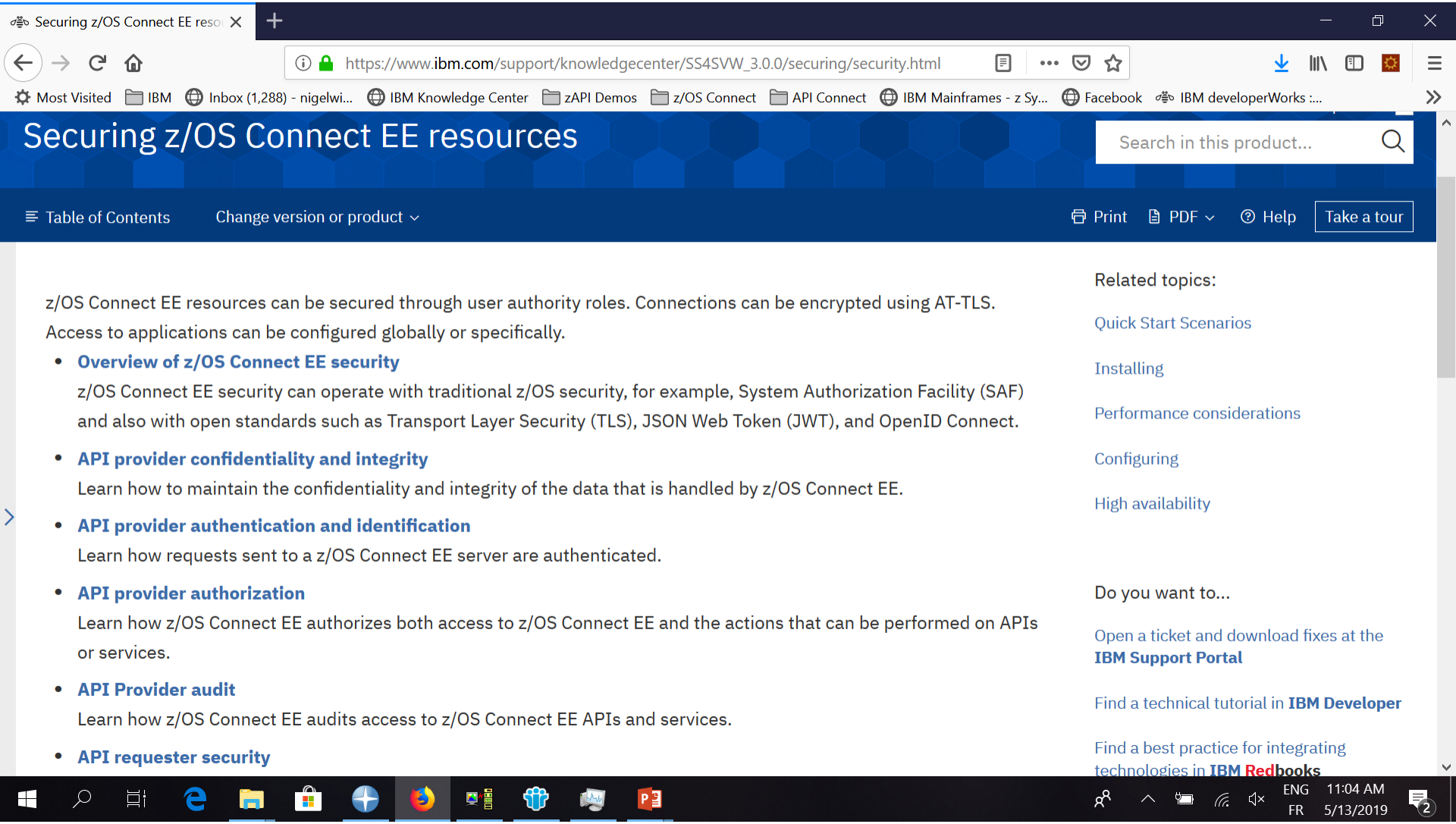
- Security design needs to consider
  - Authentication
  - Encryption
  - Authorization
  - Audit
  - Protection against attack
  - Rate limiting

- Because z/OS Connect EE is based on Liberty it benefits from a wide range of Liberty security capabilities

- z/OS Connect EE has it’s own security capabilities in the form of the authorization and audit interceptors

- Look at the security solution end to end, including the security capabilities of the API Gateway

Knowledge Center



More information