ORACLE

BKA/P20 OpenIDConnect Proxy

Konzept und Anwendungsbeispiele

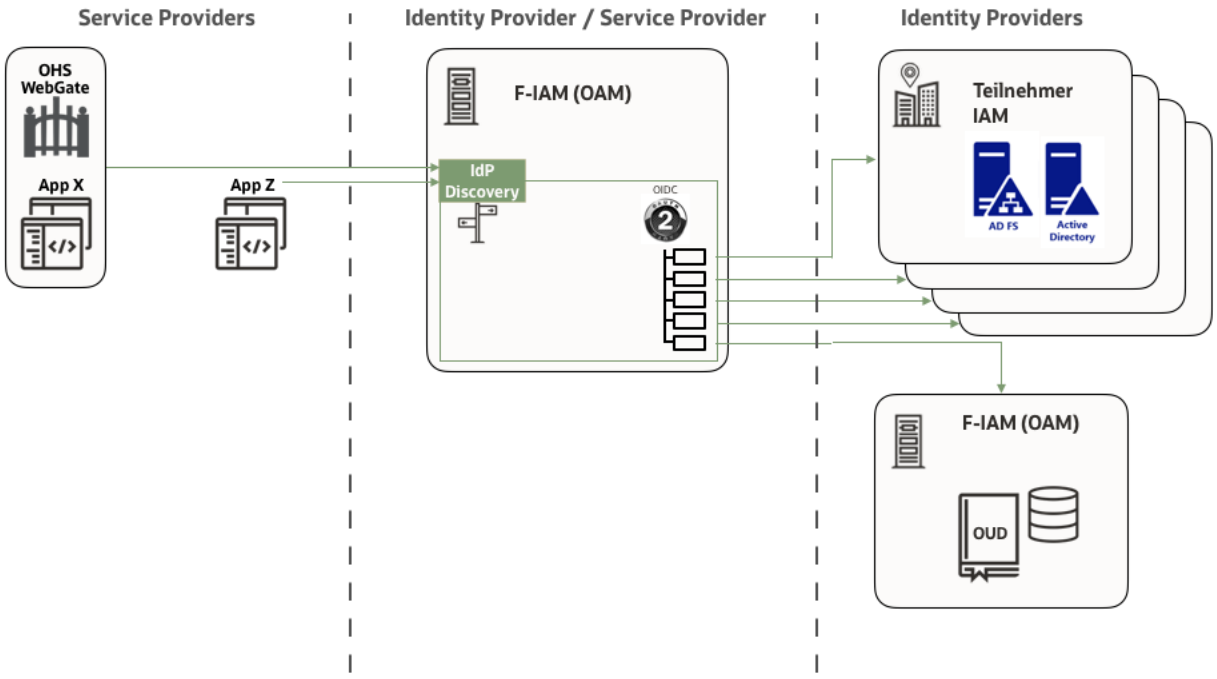
Melinda.Nath-Richter@oracle.com
Version 2.0

Inhaltsverzeichnis

Abbildungen

Tabellarische Darstellungen

Bild 1. High-Level Architektur für OpenIDConnect Proxy

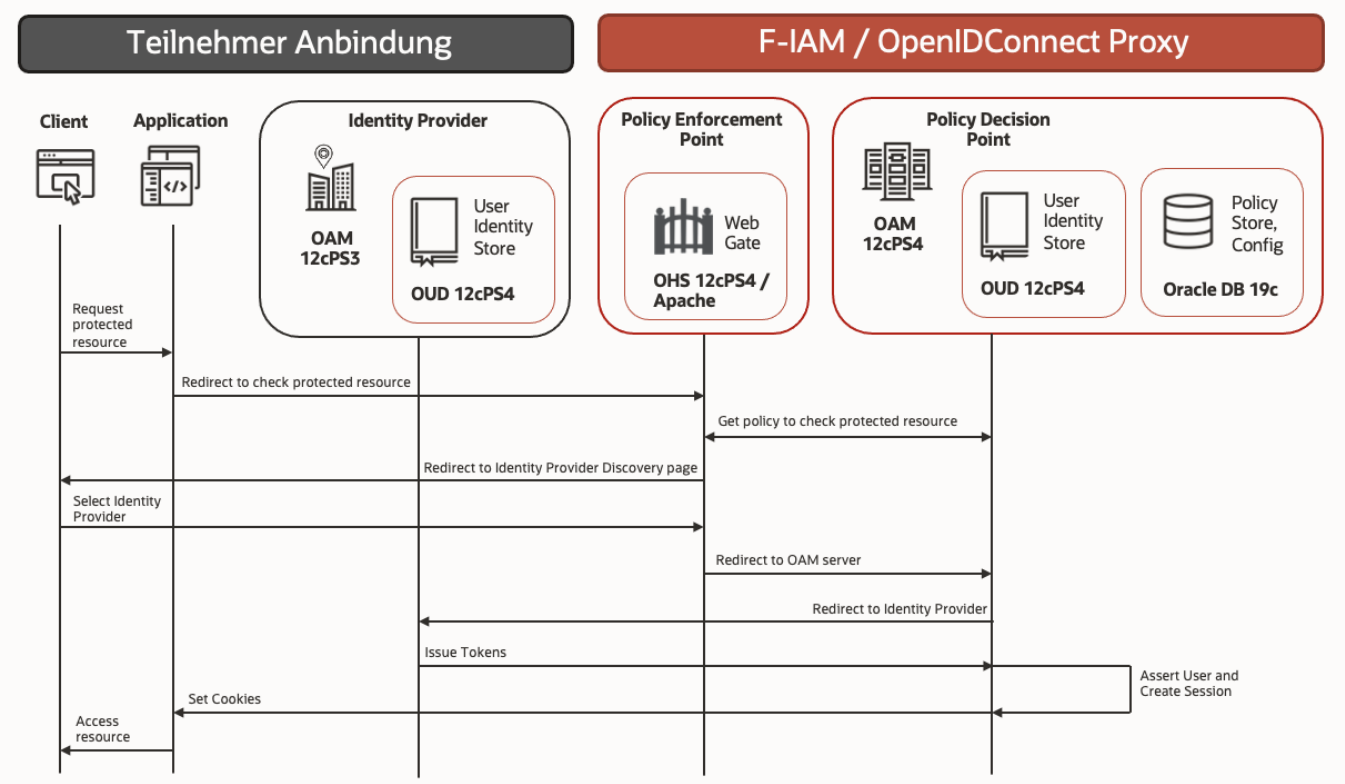


Zielbild

Dieses Dokument beschreibt Konzept und Anwendungsbeispiele um über Oracle Access Management eine Benutzer Authentifizierung via OpenIDConnect/OAuth2.0 dynamisch an einen befähigten OpenIDConnect Identitätsanbieter zu delegieren, welcher unterschiedlich pro Teilnehmer ausgeprägt sein kann. Standardmäßig stellt OAM eine Föderation via SAML zur Verfügung. Um für OpenIDConnect die gleiche „Proxy"-Funktionalität anbieten zu können, wurden zusätzlich Konfigurationen und Erweiterungen vorgenommen.

Der Föderale IAM basierend auf OAM in der aktuell verfügbaren Version 12cPS4 wechselt zwischen der Rolle eines OpenIDConnect fähigen Identitäts- und Serviceanbieters. Bei der Anforderung einer geschützten Applikation wird an einen ausgewählten bzw. übergebenen Identitätsanbieter des Teilnehmers weitergeleitet um die entsprechenden OpenIDConnect/OAuth2.0 JWT-Tokens seitens OAM als Serviceanbieter zur Verfügung zu stellen.

Bild 2. OpenIDConnect Sequenzdiagramm in einer Teststellung

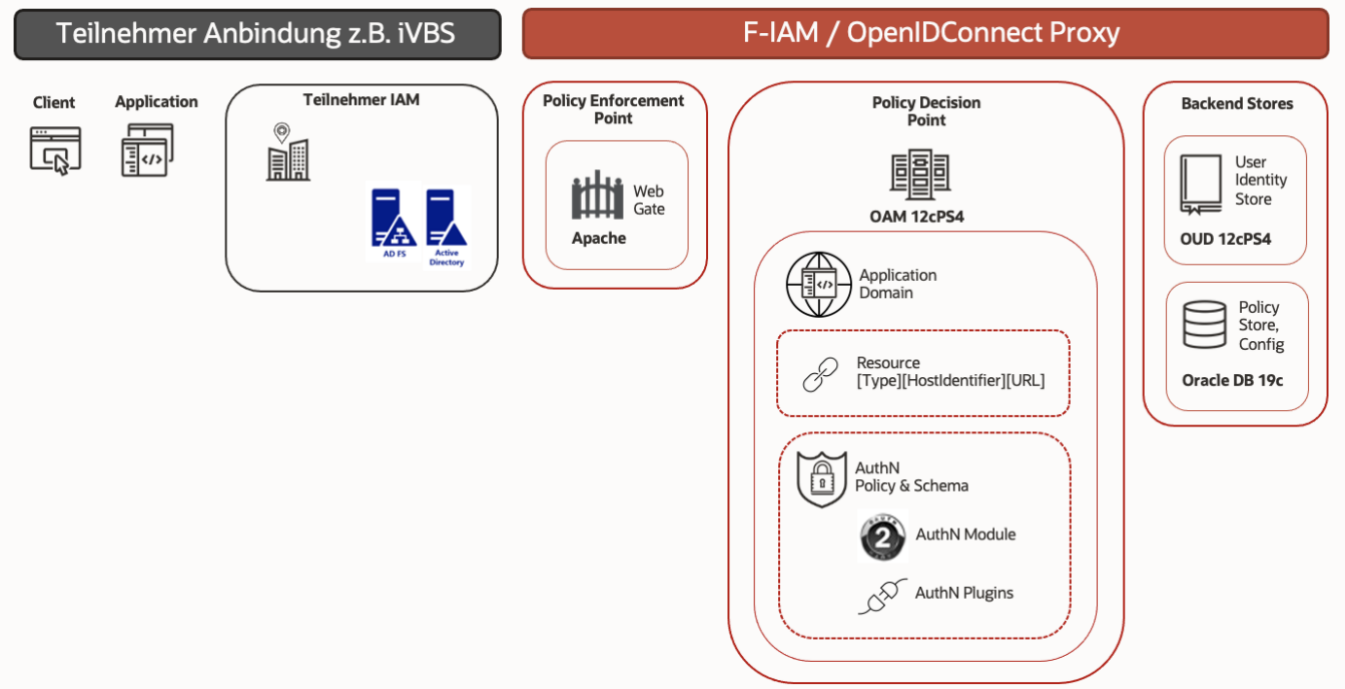


Ablauf mit OAM als Teilnehmer Identitätsanbieter und Serviceanbieter

In einer Teststellung wurde die Benutzer Authentifizierung in folgenden Schritten durchgeführt:

1. Benutzer Client (Browser) fordert Zugriff an auf geschützte Ressource (Applikation z.B. HTML-Seite).
2. Zur Überprüfung von Richtlinien zur Zugangskontrolle leitet der Policy Enforcement Point basierend auf OHS / Apache WebGate die Anfrage um zum Policy Decision Point basierend auf OAM.
3. Die entsprechenden Richtlinien für die geschützte Ressource werden ermittelt und festgestellt.
4. Es erfolgt die Umleitung auf eine Auswahl Seite mit allen verfügbaren Identitätsanbietern.
5. Der ausgewählte Identitätsanbieter wird übermittelt an das WebGate.
6. Das WebGate leitet die Anfrage an den OAM-Server mit dem ausgewählten Identitätsanbieter als URL-Parameter.
7. Die Anfrage zur Authentifizierung wird an den entsprechenden Identitätsanbieter weitergeleitet.
8. Nach erfolgter Benutzer Authentifizierung durch den Identitätsanbieter wird die Anforderung zur Ausstellung der Tokens weitergeleitet an den OAM-Server.
9. Der OAM-Server bestätigt den Benutzer und erstellt die Session.
10. Die für die Session erforderlichen Cookies werden zur Verfügung gestellt.
11. Der Zugriff auf die angeforderte Ressource ist dadurch ermöglicht.

Bild 3. OpenIDConnect Komponenten für die Teststellung



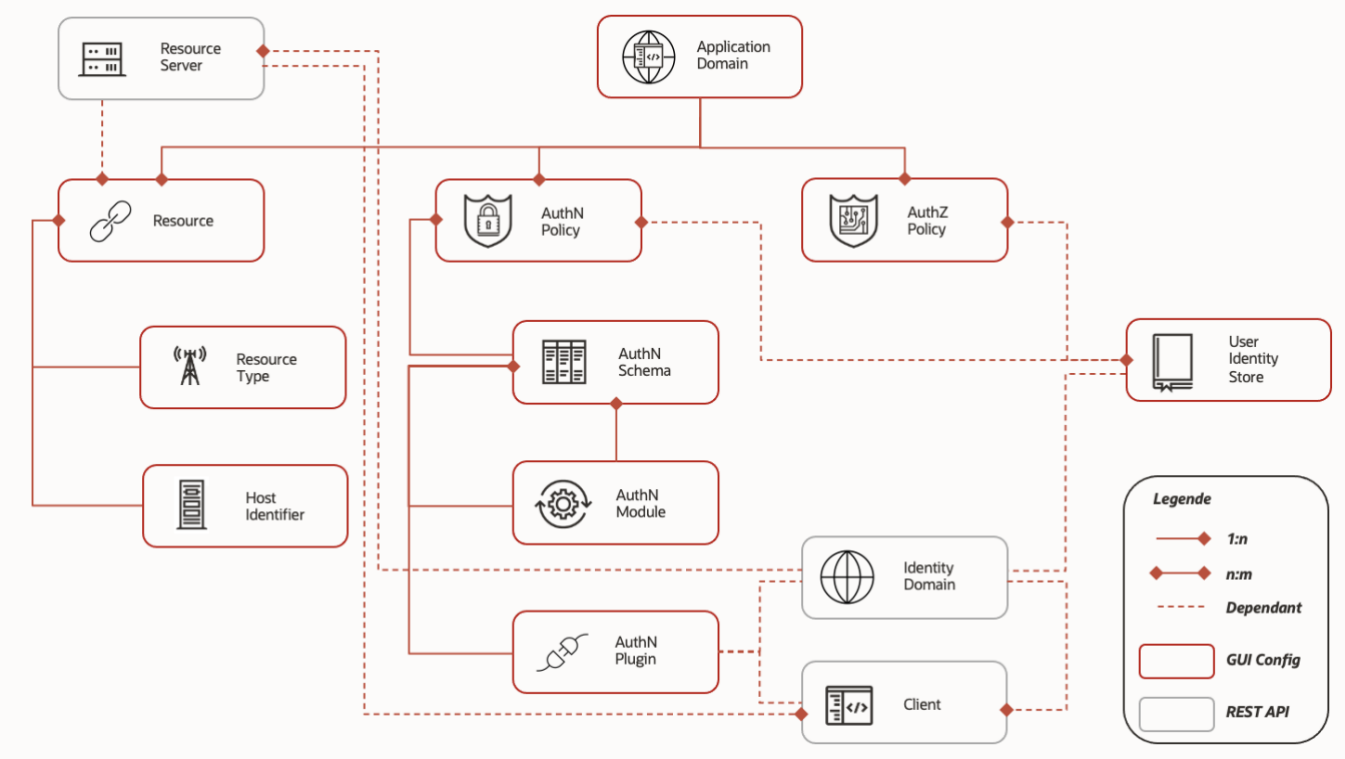
Teststellungsumgebung

In der BKA eigenen PSP-Umgebung stehen folgende Komponenten zwecks Verprobung zur Verfügung:

1. F-IAM basierend auf OAM 12cPS4 als Serviceanbieter, konfiguriert mit entsprechenden OpenIDConnect/OAuth2.0 Artefakten
   a. um über die entwickelten AuthN Plugins “IDPNameReadPlugin“ und “IDPDiscoveryPlugin“ die Anforderung an die Identitätsanbieter spezifische Instanz des Standard AuthN Plugins “OpenIDConnectPlugin“ mit ausgewähltem URL-Parameter delegieren zu können.
   b. um über das Standard AuthN Plugin “OpenIDConnectPlugin“ die Anfrage zur Authentifizierung via OpenIDConnect an den ausgewählten Identitätsanbieter weiterleiten zu können.
   c. mit einer verfügbaren Auswahl Seite für Identitätsanbieter und geschützter Applikation (Welcome Page).
2. Teilnehmer IAM basierend auf OAM 12cPS3 als Identitätsanbieter.

Die Zugang URLs können nur auf Anfrage mitgeteilt werden und sind daher in diesem Dokument nicht explizit angeführt.

Bild 4. OAM Komponenten zwecks OpenIDConnect/OAuth2.0 Konfiguration



OAM Komponenten Konfiguration

Oracle Access Management ermöglicht die Implementierung des OAuth2.0-Webautorisierungsprotokoll indem ein Client den Zugriff anfordert auf durch Oracle Access Manager (OAM) geschützte Ressourcen, die einem anderen Benutzer (d.h. dem Ressourceneigentümer) gehören. OpenIDConnect implementiert die Authentifizierung als Erweiterung des OAuth2.0-Autorisierungsprozesses. Es bietet die Generierung von ID-Tokens, welche von Clients mit OAuth2.0-Flows abgerufen werden können.

Für die Verwendung der OpenIDConnect/OAuth2.0-Services müssen folgende Komponenten über die **Oracle Access Management Console** konfiguriert werden:

- **Application Domain**
  stellt einen logischen Container dar für **Resources** und die zugehörigen **AuthN** und **AuthZ Policies** bezugnehmend auf Richtlinien die vorgeben, wer auf welche geschützten Ressourcen zugreifen kann.

- **Authentication Schema**
  regelt den Abfragemechanismus zur Benutzer Authentifizierung für den Zugriff auf Ressourcen basierend auf vordefinierten **AuthN Modulen** und **AuthN Plug-Ins**.

- **Resource**
  einheitlich innerhalb einer Application Domain definiert und abgeleitet aus **Resource Type**, z.B. standardmäßig von Oracle bereitgestellt “http“ für Webanwendungen mit Zugriff über Internetprotokolle (http oder https), und **Host Identifier**, welcher Webserver Host Name und Port referenziert.

Für die Integration mit den Oracle Access Management OpenIDConnect/OAuth2.0-Services müssen folgende Artefakte über **REST API** erstellt werden:

Identity Domain

enthält als unabhängige Entität alle Artefakte wie **ResourceServer** und deren **Clients** zur Bereitstellung von Standard-OAuth2.0-Diensten. In Cloud oder mandantenfähigen Umgebungen dient die Identity Domain zur Bereitstellung eines separaten Mandanten.

Wichtige Parameter zur Erstellung einer Identity Domain:

- **identityProvider**
  User Identity Store zum Durchführen der Authentifizierung
- **errorPageURL**
  Benutzerdefinierte Fehlerseite für 3-stufigen OAuth2.0-Flows
- **consentPageURL**
  Seite zur Einholung der Zustimmung für 3-stufigen OAuth2.0-Flows
- **tokenSettings**
  Wenn nicht angegeben werden Standardwerte für ACCESS_TOKEN verwendet

Endpoint für CRUD-Operationen:

```
http://<AdminServerHost:Port>/oam/services/rest/ssa/api/v1/oauthpolicyadmin/oauthidentitydomain
```

Beispiel cURL Befehl zur Ermittlung der vorhandenen Identity Domain:

```
curl -X GET \
'http://<AdminServerHost:Port>/oam/services/rest/ssa/api/v1/oauthpolicyadmin/oauthidentitydomai
n' \
-H 'Authorization: Basic d2VibG9naWM6d21rYWgxbWRraA==' \
-H 'Cache-Control: no-cache'
```

Beispiel Ergebnis zu o.a. Durchführung:

```
Sucessfully retrieved entity - OAuthIdentityDomain, detail - [OAuth Identity Domain :: Name -
OIDCDefaultDomain, Id - 3ee7f1f97940451dadcf8ce577ea3f61, Description - OAuth Domain,
TrustStore Identifiers - [OIDCDefaultDomain], Identity Provider - FederationIdentityStore,
TokenSettings -
[{"tokenType":"ACCESS_TOKEN","tokenExpiry":3600,"lifeCycleEnabled":false,"refreshTokenEnabled":
true,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnabled":false},
{"tokenType":"AUTHZ_CODE","tokenExpiry":3600,"lifeCycleEnabled":false,"refreshTokenEnabled":tru
e,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnabled":false},
{"tokenType":"SSO_LINK_TOKEN","tokenExpiry":86400,"lifeCycleEnabled":false,"refreshTokenEnabled
":true,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnabled":false}], ConsentPageURL -
http://<AdminServerHost:Port>/oamcontext/pages/CustomConsent.jsp, ErrorPageURL -
http://<AdminServerHost:Port>/oam/pages/error.jsp, CustomAttrs - null]
```

Benutzerdefinierte Attribute (Custom Claims) können im **Access Token**, **ID Token** und **UserInfo Endpoint** inkludiert werden und müssen zu diesem Zweck konfiguriert werden über den “Template“ Endpoint:

```
http:<AdminServerHost:Port>/oam/services/rest/ssa/api/v1/template/{CUSTOM_CLAIM_NAME}
```

Für ein benutzerdefiniertes Attribut können u.a. Default Wert, Transformation und Filter vorgesehen werden, siehe Dokumentation unter
https://docs.oracle.com/en/middleware/idm/access-manager/12.2.1.4/aiaag/understanding-openidconnect.html#GUID-E16D8F9D-A50E-44E2-B715-F3C2868A2A5C.

Um benutzerdefinierte Attribute für alle zugeordneten Clients in den entsprechenden Tokens und im UserInfo Endpoint zu inkludieren, sind die entsprechenden Custom Claims unter dem jeweiligen Identity Domain Parameter anzuführen:

- **accessTokenCustomClaims**
- **idTokenCustomClaims**
- **userInfoCustomClaims**

Resource Server

hostet geschützte Ressourcen und nimmt Zugriffsanforderungen für diese unter Verwendung von entsprechenden Tokens an.

Wichtige Parameter zur Erstellung eines Resource Server:

- **Name**
  des Resource Server
- **Scopes**
  Geltungsbereiche, welche eindeutig referenziert werden mit Präfix des Resource Server Namen, jeweils ausgeprägt mit **scopeName** und **description**
- **idDomain**
  Zuordnung zur **Identity Domain**
- **tokenAttributes**
  Liste benutzerdefinierter Attribute, die im Access Token mitversendet werden.
  Für "STATIC" Attribute wird der attributeValue fix gesetzt, während bei "DYNAMIC" der attributeValue bei Einfügen in den Access Token ausgewertet wird.

Endpoint für CRUD-Operationen:

```
http://<AdminServerHost:Port>/oam/services/rest/ssa/api/v1/oauthpolicyadmin/application
```

Beispiel cURL Befehl zur Ermittlung des vorhandenen Resource Server:

```
curl -X GET \
'http://<AdminServerHost:Port>/oam/services/rest/ssa/api/v1/oauthpolicyadmin/application?identi
tyDomainName=OIDCDefaultDomain' \
-H 'Authorization: Basic d2VibG9naWM6d21rYWgxbWRraA==' \
-H 'Cache-Control: no-cache'
```

Beispiel Ergebnis zu o.a. Durchführung:

```
Sucessfully retrieved entity - OAuthResourceServer, detail -
[IdentityDomain="OIDCDefaultDomain,Name="OAMClient",Description="OAM Client
Resource",resourceServerId="b9be1dd6-cc40-4cf6-a3d4-
f10cfd5b8d30",resourceServerNameSpacePrefix="OAMClient.",audienceClaim="null",resServerType="CU
STOM_RESOURCE_SERVER",Scopes="[{"scopeName":"profile","description":"profile"},
{"scopeName":"openid","description":"openid"}, {"scopeName":"email","description":"email"},
{"scopeName":"DefaultScope","description":"DefaultScope"}]",tokenAttributes="[{"attrName":"sess
ionId","attrValue":"$session.id","attrType":STATIC}]]
```

Client

eine Anwendung, die Anfragen stellt auf geschützte Ressourcen als Ressourceneigentümer nach dessen Genehmigung und in Folge das OAuth2.0-Protokoll initiiert. Dafür müssen Client-Profile erstellt werden, zumindest mit Anwendungsname, Client-ID und einer oder mehreren URIs zur Umleitung, nachdem die OAuth2.0-Services Zugriff gewährt oder verweigert haben.
Um einen Access Token anzufordern, holt der Client eine Autorisierung (Grant) vom Eigentümer der Ressource ein. Die OAuth2.0-Spezifikation bietet verschiedene Typen je nach Sicherheitsanwendungsfall, siehe auch http://tools.ietf.org/html/rfc6749#section-1.3.
Für den 3-stufigen OAuth2.0-Flow ist der der Grant-Typ "Authorization Code" erforderlich, wo der Access Token gegen einen Autorisierungscode zusammen mit Client Anmeldeinformationen ausgetauscht wird.

Wichtige Parameter zur Erstellung eines Clients:

- **Name**
  des Clients
- **idDomain**
  Zuordnung zur **Identity Domain**
- **secret**
  Client Secret für Typ CONFIDENTIAL_CLIENT
- **clientType**
  Client Typ, mögliche Werte: CONFIDENTIAL_CLIENT, PUBLIC_CLIENT, MOBILE_CLIENT
- **redirectURIs**
  Auflistung der konfigurierten URIs zur Umleitung für den Client
- **attributes**
  Auflistung der Custom Attribute für den Client
- **grantTypes**
  Auflistung der erlaubten Grant-Typen für den Client, mögliche Werte: AUTHORIZATION_CODE, JWT_BEARER, REFRESH_TOKEN, CLIENT_CREDENTIALS, PASSWORD
- **Scopes**
  Auflistung der Geltungsbereiche, auf welche der Client Zugriff hat,
  jeweils ausgeprägt unter **scopeName**, referenziert durch **<ResourceServerName>.<scopeName>**
- **defaultScope**
  zieht im Runtime-Flow, wenn kein Scope angegeben wurde

Um benutzerdefinierte Attribute in den entsprechenden Tokens und im UserInfo Endpoint zu inkludieren, sind die entsprechenden Custom Claims, wie zuvor unter der Identity Domain definiert, im jeweiligen Parameter anzuführen:

- **accessTokenCustomClaims**
- **idTokenCustomClaims**
- **userInfoCustomClaims**

Endpoint für CRUD-Operationen:

```
http://<AdminServerHost:Port>/oam/services/rest/ssa/api/v1/oauthpolicyadmin/client
```

Beispiel cURL Befehl zur Ermittlung des vorhandenen Clients:

```
curl -X GET \
'http://<AdminServerHost:Port>/oam/services/rest/ssa/api/v1/oauthpolicyadmin/client?identityDom
ainName=OIDCDefaultDomain' \
-H 'Authorization: Basic d2VibG9naWM6d21rYWgxbWRraA==' \
-H 'Cache-Control: no-cache'
```

Beispiel Ergebnis zu o.a. Durchführung:

```
Sucessfully retrieved entity - OAuthClient, detail - [OAuth Client - uid = 44aec113-5baf-4793-
baa6-f805bd0b1c87, name = oam_client, id = oam_client, identityDomain = OIDCDefaultDomain,
description = OAM client, secret = ******, , clientType = CONFIDENTIAL_CLIENT, grantTypes =
[PASSWORD, CLIENT_CREDENTIALS, JWT_BEARER, REFRESH_TOKEN, AUTHORIZATION_CODE], attributes =
[{"attrName":"customeAttr1","attrValue":"CustomValue","attrType":STATIC},
{"attrName":"sessionId","attrValue":"$session.id","attrType":STATIC}], scopes =
[OAMClient.profile, OAMClient.openid, OAMClient.email], defaultScope = OAMClient.openid,
redirectURIs =
[{"url":"https://<AdminServerHost:Port>/oam/server/auth_cred_submit","isHttps":true}]]
```

PKCE-Aktivierung und Code Verifier/Challenge-Generierung

Unter OAM 12cPS4 wird der OAuth2.0-Flow mit Verwendung des Proof Key of Code Exchange (PKCE) unterstützt und muss entweder auf Identity Domain oder Client Ebene aktiviert werden.

Auf **Identity Domain** Ebene ist das **Custom Attribut "usePKCE"** entsprechend auf einen der u.a. Werte zu setzen:

Tabelle 1. PKCE-Aktivierung auf Identity Domain Ebene

WERT	BESCHREIBUNG
ALL_CLIENTS_TYPES	PKCE-Aktivierung erfolgt für alle Client Typen. Werden keine Werte für die Parameter "code_verifier" und "code_challenge" für Grant Type Authorization Code übergeben wird in den 3-stufigen OAuth2.0-Flow ohne PKCE gewechselt.
ALL_CLIENTS_TYPES_STRICT	PKCE-Aktivierung erfolgt verpflichtend für alle Client Typen. Werden keine Werte für die Parameter "code_verifier" und "code_challenge" übergeben, schlägt der OAuth2.0-Flow fehl.
PUBLIC_CLIENTS	PKCE-Aktivierung erfolgt nur für PUBLIC Client Typen. Werden keine Werte für die Parameter "code_verifier" und "code_challenge" für Grant Type Authorization Code übergeben wird in den 3-stufigen OAuth2.0-Flow ohne PKCE gewechselt. Für andere Client Typen werden die Parameter ignoriert.
PUBLIC_CLIENTS_STRICT	PKCE-Aktivierung erfolgt verpflichtend für PUBLIC Client Typen Werden keine Werte für die Parameter "code_verifier" und "code_challenge" übergeben, schlägt der OAuth2.0-Flow fehl.

Auf **Client** Ebene ist der **Parameter "usePKCE"** entsprechend auf einen der u.a. Werte zu setzen:

Tabelle 2. PKCE-Aktivierung auf Client Ebene

WERT	BESCHREIBUNG
STRICT	PKCE-Aktivierung erfolgt verpflichtend. Werden keine Werte für die Parameter "code_verifier" und "code_challenge" übergeben, schlägt der OAuth2.0-Flow fehl.
NON_STRICT	PKCE-Aktivierung erfolgt mit der Option, dass bei nicht vorhandenen Werten für die Parameter "code_verifier" und "code_challenge" in den 3-stufigen OAuth2.0-Flow ohne PKCE gewechselt wird.

Als URL-Parameter sind zu übergeben
der **code_verifier** als kryptographisch zufällig generierte Zeichenkette

- bestehend aus den Zeichen A-Z, a-z, 0-9
- Interpunktionszeichen Bindestrich, Punkt, Unterstrich und Tilde
- einer Anzahl zwischen 43 und 128 Stellen

Der code_verifier muss eindeutig für jede Autorisierungsanfrage sein und wird für die Ausstellung des Access Token benötigt. Beispiel für code_verifier:

```
SjpAT6aKtUfkHoREgJCoyH.SIfvx~JzdMkaepEzMpEcD44xxNY0l9tf-
GcYZSvMOhQSvVV3RNeGhh7upDKsiKZZMVqMLh1j7usBguvWWrQvV
```

Die **code_challenge** bezieht sich auf die umgewandelte Base64-kodierte Zeichenfolge des SHA256-Hashwertes von code_verifier. Diese wird bei der Autorisierungsanfrage für den Autorisierungscode zur späteren Gegenprüfung mit dem code_verifier auf dem Autorisierungsserver gespeichert. Nur wenn z.B. der Hash-Wert der gespeicherten code_challenge mit dem code_verifier übereinstimmt wird der Access Token ausgestellt.
Beispiel für code_challenge basierend auf o.a. code_verifier:

```
hYC4Bw_F1SYRt8UENhHrwaQzLazEvTfnVaGqCgWV1kk
```

Referenz: https://tonyxu-io.github.io/pkce-generator

Die **code_challenge_method**, welche zur Gegenprüfung der code_challenge zu verwenden ist, kann eine der folgenden Möglichkeiten enthalten

- **Plain** setzt code_verifier gleich mit code_challenge anstelle des code_verifier Hash-Wertes
- **S256** wird empfohlen und gibt den Hash-Wert des code_verifiers als code_challenge an

Bild 5. 3-stufiger OAuth2.0-Flow inkl PKCE

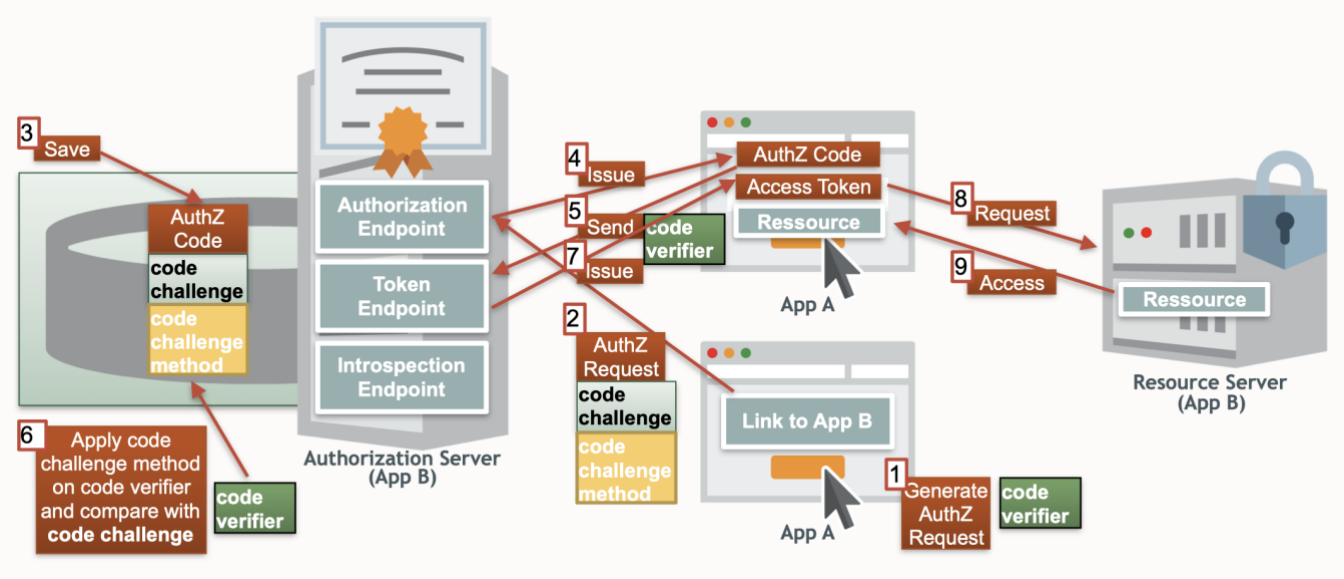
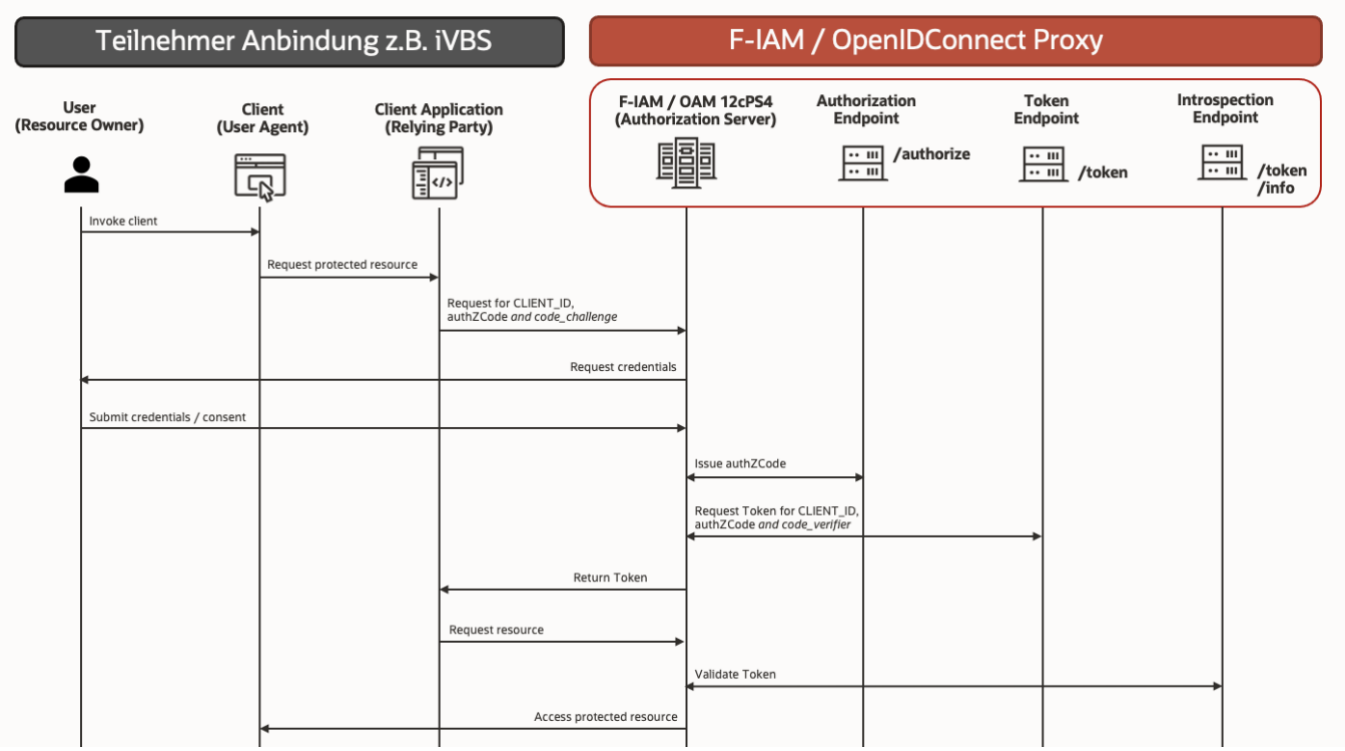


Bild 6. OpenIDConnect Sequenzdiagramm basierend auf 3-stufigem OAuth2.0-Flow



OpenIDConnect-Flow

Oracle Access Manager unterstützt OpenIDConnect sowohl 2-stufig (Implicit Grant) als auch 3-stufig mit Authorization Code. Da das 2-stufige Verfahren veraltet ist und Sicherheitsrisiken birgt, wird empfohlen Authorization Code mit der Option PKCE zu verwenden.

OpenIDConnect stellt zusätzlich eine Identitätsebene auf dem zugrundeliegenden OAuth2.0-Protokoll dar und ermöglicht:

- Überprüfung der Identität des Endbenutzers über die Authentifizierung durch einen Autorisierungsserver
- Abrufen von Profil Informationen basierend auf interoperabler REST API

Teil 1: 3-stufiger OAuth2.0-Flow (Authorization Code)

Der Autorisierungsserver erhält vom Client eine Autorisierungsanfrage mit folgenden Parametern im Query-String:

- **response_type** enthält “code“
- **domain** enthält Identity Domain Namen
- **client_id** enthält Client Kennung
- **redirect_uri** enthält beim Client registrierte Redirect URI an welche die Antwort gesendet wird
- **scope** enthält durch Pluszeichen (+) getrennte Liste der angeforderten Geltungsbereich Berechtigungen, welche jeweils referenziert werden mit <ResourceServerName>.<ScopeName>
  Für OpenIDConnect muss "openid" angegeben werden. Ist dieser nicht angegeben, wird der Flow wie ein Standard OAuth2.0-Flow behandelt.
- **state** optional und empfohlen, um den Status zwischen Anfrage und Antwort zu überprüfen.

- **nonce** verwendeter Wert, um eine Client Session mit einem ID-Token zu verknüpfen und Wiederholungsangriffe abzuschwächen.

Nachdem alle Parameter vom Autorisierungsserver erfolgreich validiert wurden, wird der Benutzer aufgefordert sich anzumelden und den Client Zugriff zu genehmigen. Nach Genehmigung leitet der Autorisierungsserver an die Redirect URI um mit folgenden Parametern im Query-String:

- **code** enthält Autorisierungscode
- **state** enthält Wert aus der ursprünglichen Anfrage. Dieser Wert dient zum Vergleich mit dem in der Session gespeicherten Wert, um sicherzustellen, dass der erhaltene Autorisierungscode passend als Antwort auf die entsprechende Anfrage vom selben Client (und nicht von einem anderen Client) stammt.

Teil 2: Token Ausstellung (inkl ID-Token)

Der Autorisierungsserver erhält vom Client eine POST-Anfrage mit folgenden Parametern:

- **Authorization Header** enthält Base64-kodiert Client ID und Secret
- **oauth-identity-domain-name** als Header Parameter enthält Identity Domain Namen
- **grant_type** enthält "AUTHORIZATION_CODE"
- **code** als Query-String enthält Autorisierungscode
- **redirect_uri** enthält beim Client registrierte Redirect URI an welche die Antwort gesendet wird

Der Autorisierungsserver antwortet mit einem JSON-Objekt mit folgenden Properties:

- **token_type** enthält "Bearer"
- **expires_in** enthält "Time-to-live" Integer-Wert für den Access Token
- **access_token** enthält den Access Token als JWT, signiert mit dem Private Key der Identity Domain. Zwecks Bestätigung wird der X5T-Wert (X509 Zertifikat) aus dem Header und der zugehörige Public Key abgerufen, um den JWT zu verifizieren.
- **refresh_token** enthält verschlüsseltes Payload zwecks Erneuerung des Access Token, wenn dieser abgelaufen ist.
- **id_token** enthält JWT mit Claims über die Authentifizierung eines Endbenutzers durch den Autorisierungsserver.

Beispiel aus der Teststellung

Im ersten Schritt / Teil 1 wird der Autorisierungscode ermittelt

Option A ohne Identitätsanbieter, wodurch umgeleitet wird an IdP Auswahl Seite:

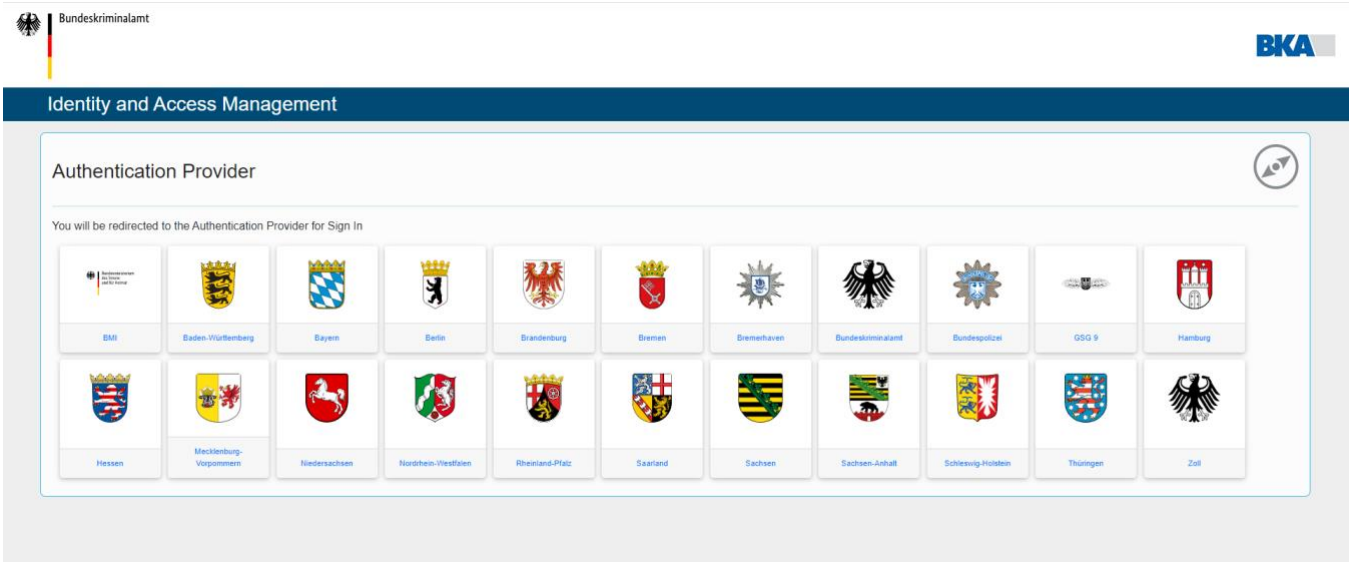
```
https://<AdminServerHost:Port>/oauth2/rest/authorize?response_type=code&domain=OIDCDefaultDomai
n&client_id=oam_client&scope=OAMClient.openid&redirect_uri=https://<Host:Port>/test/index.html
```

Option B mit ausgewähltem Identitätsanbieter als Query-String Parameter **idp_name**:

```
https://<AdminServerHost:Port>/oauth2/rest/authorize?idp_name=
OIDCIdpOAM&response_type=code&domain=OIDCDefaultDomain&client_id=oam_client&scope=OAMClient.ope
nid&redirect_uri=https://<Host:Port>/test/index.html
```

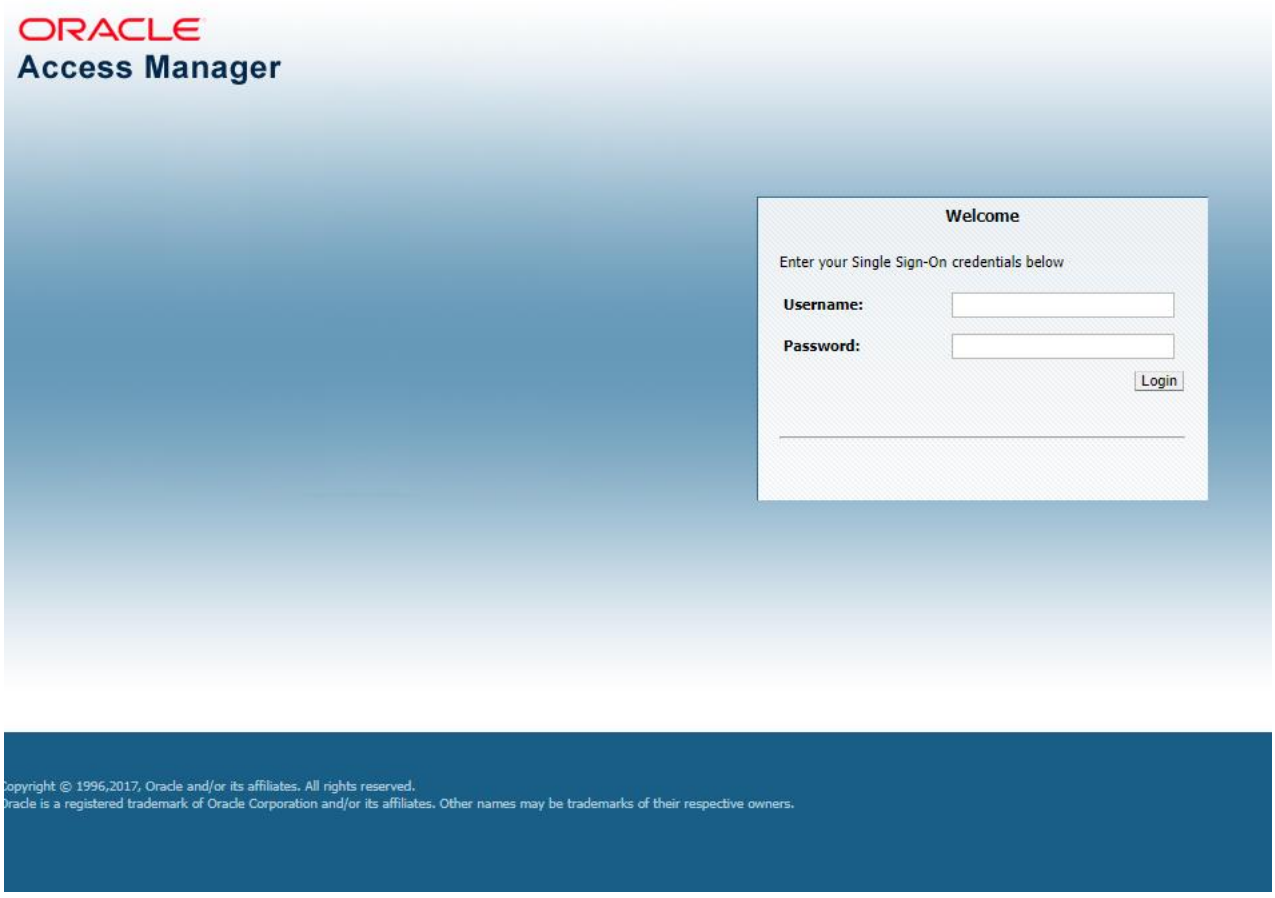
Umleitung bei Option A zur IdP Auswahl Seite seitens OpenIDConnect Proxy:

Bild 7. OpenIDConnect Proxy / IdP Auswahl Seite



Umleitung zu ausgewähltem Teilnehmer Identitätsanbieter, z.B. OAM Benutzer Anmeldung:

Bild 8. Beispiel Teilnehmer IdP / OAM Benutzer Anmeldung Seite



Nach erfolgter Benutzer Authentifizierung Umleitung zur geschützten Applikation mit Autorisierungscode:

```
https://<Host:Port>/test/index.html?code=
MVNHYXl2NjNtYzFKTGhya3hzQlVXdz09fml2K3VHWi9oN2R2eU94K0lxUW0wWnRCOWQzd2tGelpPRFlhenRPSTIrcEhEM1V
5TkNlMTNqcjZSTFgzQXl1WlBRNVBNV3NtQW5hUUs0dHY4Z1RxN04yR2ZGa1hNT2cwN2dXSm1MMUc2UE5PbDJFa3R6MEpMKz
dlbUdxNEhsZ1o1dlBsVk9XWFVtbnJycUlINE00dDdwdmhITjA2K1ZRMS9YWFY2VnFDL2VsQzB5Ulc0QVhnMjZZZnNNb1g5R
E9oeDBibCtreEU5eU0rQXVUdWp2OHpPcTNtZ1pEbmV6ckIwS0tHZlhoaW1VekJoUmxHbDFVK1FySnIyMXl6Q3Z2R3l3Si85
elplWkZqZ0RoVEJaOFBxMFlQZUpxOTJwMnBIcTlaVk9FTVlrMU1QTlI2WnVaVUt0T3Npa3lpdTY0RWVIZDI4ekNtbGZ4QjN
rSDd6OWJqYjZXWElYTzVLa1YyY090eVpEWllMbU1qdEx1bXgvWHhhSFpUV2dXZEtJUmg1VmtJWnZNZDZ0blhwNkptNkdWSk
UzWDFMVzNGTkpWYUpGajBIRFZzSTBEa0kyUU1nUlp5VnV5T1F3dERYQWQ4QTVoR3lMWHhzdXd3cUI2OHBTRWhIMzBiZ2RIc
1BYZzBiajNsWU85dVNjMlZ4RmlRR1hGa3loVWVoODlvN2tOc3Z2cjRsZlRkSXBkNDdtVFFBcFZ1ajBSZzZKWE1sWU1RV3dk
NHYrV1puWkwwb21rRGhWUzQvbEVtNHNNS3BDRjVwd1hqcWhQVlBkQlR2azdOSlp4RW9ITFo5VGVFd3NQeU5ZbHdraXVJek9
SVGFEM3pXU3lwWEZXOUtTSGxqSHpYdS8=
```

Im zweiten Schritt / **Teil 2** können Access und ID-Token ermittelt werden.

Beispiel cURL Befehl zur Token Ermittlung anhand des Autorisierungscode:

```
curl --location --request POST 'http://<AdminServerHost:Port>/oauth2/rest/token' \
--header 'cache-control: no-cache' \
--header 'content-type: application/x-www-form-urlencoded' \
--header 'x-oauth-identity-domain-name: OIDCDefaultDomain' \
--header 'Authorization: Basic b2FtX2NsaWVudDpXZWxjb211lMQ==' \
--data-urlencode 'grant_type=AUTHORIZATION_CODE' \
--data-urlencode
'code=MVNHYXl2NjNtYzFKTGhya3hzQlVXdz09fml2K3VHWi9oN2R2eU94K0lxUW0wWnRCOWQzd2tGelpPRFlhenRPSTIrc
EhEM1V5TkNlMTNqcjZSTFgzQXl1WlBRNVBNV3NtQW5hUUs0dHY4Z1RxN04yR2ZGa1hNT2cwN2dXSm1MMUc2UE5PbDJFa3R6
MEpMKzdlbUdxNEhsZ1o1dlBsVk9XWFVtbnJycUlINE00dDdwdmhITjA2K1ZRMS9YWFY2VnFDL2VsQzB5Ulc0QVhnMjZZZnN
Nb1g5RE9oeDBibCtreEU5eU0rQXVUdWp2OHpPcTNtZ1pEbmV6ckIwS0tHZlhoaW1VekJoUmxHbDFVK1FySnIyMXl6Q3Z2R3
l3Si85elplWkZqZ0RoVEJaOFBxMFlQZUpxOTJwMnBIcTlaVk9FTVlrMU1QTlI2WnVaVUt0T3Npa3lpdTY0RWVIZDI4ekNtb
GZ4QjNrSDd6OWJqYjZXWElYTzVLa1YyY090eVpEWllMbU1qdEx1bXgvWHhhSFpUV2dXZEtJUmg1VmtJWnZNZDZ0blhwNkpt
NkdWSkUzWDFMVzNGTkpWYUpGajBIRFZzSTBEa0kyUU1nUlp5VnV5T1F3dERYQWQ4QTVoR3lMWHhzdXd3cUI2OHBTRWhIMzB
iZ2RIc1BYZzBiajNsWU85dVNjMlZ4RmlRR1hGa3loVWVoODlvN2tOc3Z2cjRsZlRkSXBkNDdtVFFBcFZ1ajBSZzZKWE1sWU
1RV3dkNHYrV1puWkwwb21rRGhWUzQvbEVtNHNNS3BDRjVwd1hqcWhQVlBkQlR2azdOSlp4RW9ITFo5VGVFd3NQeU5ZbHdra
XVJek9SVGFEM3pXU3lwWEZXOUtTSGxqSHpYdS8=' \
--data-urlencode 'redirect_uri=https://<AdminServerHost:Port>/oam/server/auth_cred_submit' \
--data-urlencode 'scope=OAMClient.openid'
```

Beispiel Ergebnis zu o.a. Durchführung:


```
{"access_token":"eyJraWQiOiJPSURDRGVmYXVsdERvbWFpbiIsIng1dCI6Ims3U2Q5ckpycDR0WHd2S0x5Zm42U0hDQm
k4ayIsImFsZyI6IlJTMjU2In0.eyJpc3MiOiJodHRwOi8vdndhc2UwNTEuemRzLmJrYS5idW5kLmRlOjc4MDcvb2F1dGgyI
iwiYXVkIjoib2FtX2NsaWVudCIsImV4cCI6MTY3NTY4MjI2OSwianRpIjoiV2pkZXk2ejk4TE5aRFVFSzFScG9JdyIsImlh
dCI6MTY3NTY3ODY2OSwic3ViIjoiYmtiazQ3MTEzODgiLCJjdXN0b21lQXR0cjEiOiJDdXN0b21WYWx1ZSIsInNlc3Npb25
JZCI6IjE2NmEyYWZiLWIyODQtNDNhMC05M2FjLWQ0ZWU2ZTI5ZDNiZXxBcUxpa3ZiZlV1MUdvVlZRd0g0OERlc2JwbDgraF
NYdmxxajNQMVZUeGw0PSIsImNsaWVudCI6Im9hbV9jbGllbnQiLCJzY29wZSI6WyJvcGVuaWQiLCJlbWFpbCJdLCJkb21ha
W4iOiJPSURDRGVmYXVsdERvbWFpbiJ9.Oy_MWlg8nAGzPQTL3Ps1tltbvs7vxQPTxzSL_P9byfjVKbhYTyqgr4_bO3RPfIG
VwdCaMyLNFHs736yomhTdGD99RaEABPBQJjfa8iDsRTIt1KuXXRdIQKIXjarBjuJPiXNK2_N98A-
kTjylelLtxhW488w5L96bZbxOwOFgo_QaoEPHsXQp5PRJre9IJ3gbATZCBRsAOcrfO8JExV0N_m1XibL-
VKtg4lhXmaNxyYICWwn0zn_7ptJyPc88ieHT3QldfhlUdU5WjN5Icng8qW6TjJnh7HJmqck9skyXFgHlOBEyNoEplCtOduV
5GrFY1vPh3VVRmX_Is8PzgH0vOA","token_type":"Bearer","expires_in":3600,"refresh_token":"WQRS3dOcG
3qaWP6CAOzbkg%3D%3D%7EP%2BVNeEgdmqW9ccOCnhNAamH7KhzsIWBcfYnXNh5278Yll51A6CZWnUyPTtGXcsfarpW2thS
31801k23aF38u5wV%2FHifxexS1x4FeSU1TFCzLXvHMoHMnwK42V10QQq9bH1kxQZKsTUHgZWj%2BJiz5m5ci5ocmht0W9K
Ta9s5FHP3DvzpPAwKBMtJ1ppOiRs5K%2FJhj9ngyW5GnhZSZVjcTlxEJBLU0McS7ALHxAIu%2FrWfWjcIMIrxSkN%2FV50P
r6bL%2FcqPTgFMA%2BOJNqK2nZFgk3w%3D%3D","id_token":"eyJraWQiOiJPSURDRGVmYXVsdERvbWFpbiIsIng1dCI6
Ims3U2Q5ckpycDR0WHd2S0x5Zm42U0hDQmk4ayIsImFsZyI6IlJTMjU2In0.eyJpc3MiOiJodHRwOi8vdndhc2UwNTEuemR
zLmJrYS5idW5kLmRlOjc4MDcvb2F1dGgyIiwic3ViIjoiYmtiazQ3MTEzODgiLCJhdWQiOlsib2FtX2NsaWVudCIsImh0dH
A6Ly92d2FzZTA1MS56ZHMuYmthLmJ1bmQuZGU6NzgwNy9vYXV0aDIiXSwiZXhwIjoxNjc1NjgyMjY5LCJpYXQiOjE2NzU2N
zg2NjksImp0aSI6IjZZbEZoNk5LdkdLV2JUX0xidFBaY3ciLCJhdF9oYXNoIjoiT1djRkItU3J2aXBVel82Y2dwd0lNUSIs
ImF6cCI6Im9hbV9jbGllbnQiLCJhY3IiOiIyIiwic2lkIjoiTDBRZ2tnM3BzRkJlcEpUMjBVWmdTQT09flVkWldlLzR4Zmx
BUitnNXV6MTJsVFByb3pqMk9pK0s0eDhZNzBlWGM4UmQ5bmh0aGUxbnVaaWNwTHl1UVQrNjVDd0plRTR3dGN0a3F2TkNyWE
E0cUxOeGlLSURKR05iQWhZMzBCZDFBMGxTNllSR3lPYXpWdXlPUFRHcEhKNDRHIiwiYXV0aF90aW1lIjoiMTY3NTY3ODYxO
Dg2NSIsImFtciI6WyJwd2QiXSwiZW1haWwiOnsiZW1haWxfdmVyaWZpZWQiOiJOIiwiZW1haWwiOiJhbGZvbnMuemlv0dGVy
QHZtLm9yYWNsZS5jb20ifX0.OCOMY7_HcF89HbhXJomjt1vHCytHmD_ncNcZBJwvU2bvGdTnpJd8dQn3PHaKnmLemmlv4u2
ETFCFjkpsxOof7W2AZp-npWMuXt78qQ5kB7r-yh1FeL6Wk58RASPEcoAtKtK3SouIrYYqX1llUzynPZ-
2WYPiA3nfJg4l3P7Wz1ywy6DF1_bACQq50cAjKz0GSdhbtOemWYzJRl-A-sBX4DOqj4s6FOrT5UoTXBJWaeJSHc-
HhI7VXLFCcP9KlYprh5rib-jwqHXfEfooigTUxvslJEPWfT36Yz89rxVfj18x-
enaUK4zzsfAhkQKlnd1BvM7NzCK1Q4FsDXrmPNnvw"}
```


Dekodierte Versionen von ID- und Access Token (https://jwt.io):

Bild 9. Beispiel ID-Token

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "kid": "OIDCDefaultDomain",
  "x5t": "k7Sd9rJrp4tXwvKLyfn6SHCBi8k",
  "alg": "RS256"
}
```


PAYLOAD: DATA


```
{
  "iss": "http://                              /oauth2",
  "sub": "bkbk4711388",
  "aud": [
    "oam_client",
    "http://                              /oauth2"
  ],
  "exp": 1675682269,
  "iat": 1675678669,
  "jti": "6YlFh6NKvGKWbT_LbtPZcw",
  "at_hash": "OWcFB-SrvipUz_6cgpwIMQ",
  "azp": "oam_client",
  "acr": "2",
  "sid":
"L0Qgkg3psFBepJT20UZgSA==~UdZWe/4xflAR+g5uz12lTProzj2Oi+
K4x8Y70eXc8Rd9nhthe1nuZicpLyuQT+65CwJeE4wtctkqvNCrXA4qLN
xiKIDJGNbAhY30Bd1A0lS6YRGyOazVuyOPTGpHJ44G",
  "auth_time": "1675678618865",
  "amr": [
    "pwd"
  ],
  "email": {
    "email_verified": "N",
    "email": "alfons.zitter@vm.oracle.com"
  }
}
```


Bild 10. Beispiel Access Token

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "kid": "OIDCDefaultDomain",
  "x5t": "k7Sd9rJrp4tXwvKLyfn6SHCBi8k",
  "alg": "RS256"
}
```


PAYLOAD: DATA


```
{
  "iss": "http://                              /oauth2",
  "aud": "oam_client",
  "exp": 1675682269,
  "jti": "Wjdey6z98LNZDUEK1RpoIw",
  "iat": 1675678669,
  "sub": "bkbk4711388",
  "customeAttr1": "CustomValue",
  "sessionId": "166a2afb-b284-43a0-93ac-
d4ee6e29d3be|AqLikvbfUu1GoVVQwH48Desbpl8+hSXvlqj3P1VTxl4
=",
  "client": "oam_client",
  "scope": [
    "openid",
    "email"
  ],
  "domain": "OIDCDefaultDomain"
}
```

Access Token

Zusätzlich zu den u.a. standardmäßig vorhandenen Claims (Attributen) können benutzerdefinierte Attribute im Access Token aufgenommen werden. Diese können entweder
a) bei der Erstellung des Resource Server (**tokenAttributes**)
b) auf Identity Domain Ebene für alle zugeordneten Clients (**accessTokenCustomClaims**) oder
c) in einem bestimmten Client (**accessTokenCustomClaims**) konfiguriert werden.

Tabelle 3. Access Token Standard Claims

CLAIM NAME	BEISPIEL WERT	BESCHREIBUNG
HEADER		
kid	OIDCDefaultDomain	Schlüssel-ID mit eindeutiger Kennung des Unterzeichners, um die JSON-Websignatur (JWS) des Tokens zu validieren.
x5t	k7Sd9rJrp4tXwvKLyfn6SHCBi8k	X.509-Zertifikats-Thumbprint liefert Base64URL-kodiert den SHA-256-Thumbprint für die DER-Kodierung eines X.509-Zertifikats, welcher zum Zertifikats-Abgleich verwendet werden kann.
alg	RS256	Der zum Signieren des Tokens verwendete Algorithmus.
PAYLOAD (Standard)		
iss	http://<AdminServerHost:Port>/oauth2	Aussteller-ID, welcher die Antwort zurück liefert.
aud	oam_client	Resource Server Name gemäß OAM Konfiguration.
exp	1675682269	Ablaufzeit-Zeitstempel, ab welchem der JWT NICHT mehr zur Verarbeitung angenommen werden MUSS. Siehe EPOCH-Zeitformat: https://www.unixtimestamp.com.
jti	Wjdey6z98LNZDUEK1Rpolw	JWT-ID enthält eindeutige Kennung für den JWT.
iat	1675678669	Ausstellungs-Zeitstempel, ab welcher der JWT zur Verfügung gestellt wurde.
sub	bkbk4711388	Auftraggeber bzw. Gegenstand (Subject) des JWT. Beinhält im CLIENT CREDENTIALS Flow die Anwendung (Client) selbst.
client	oam_client	Client Kennung gemäß OAM Konfiguration.
scope	["openid","email"]	Geltungsbereich.
domain	OIDCDefaultDomain	Identity Domain gemäß OAM Konfiguration.

ID Token

Zusätzlich zu den u.a. standardmäßig vorhandenen Claims (Attributen) können benutzerdefinierte Attribute im ID Token aufgenommen werden. Diese können entweder
a) auf Identity Domain Ebene für alle zugeordneten Clients (**idTokenCustomClaims**) oder
b) in einem bestimmten Client (**idTokenCustomClaims**) konfiguriert werden.

Tabelle 4. ID Token Standard Claims

CLAIM NAME	BEISPIEL WERT	BESCHREIBUNG
HEADER		
kid	OIDCDefaultDomain	Schlüssel-ID mit eindeutiger Kennung des Unterzeichners, um die JSON-Websignatur (JWS) des Tokens zu validieren.
x5t	k7Sd9rJrp4tXwvKLyfn6SHCBi8k	X.509-Zertifikats-Thumbprint liefert Base64URL-kodiert den SHA-256-Thumbprint für die DER-Kodierung eines X.509-Zertifikats, welcher zum Zertifikats-Abgleich verwendet werden kann.
alg	RS256	Der zum Signieren des Tokens verwendete Algorithmus.
PAYLOAD (Standard)		
iss	http://<AdminServerHost:Port>/oauth2	Aussteller-ID, welcher die Antwort zurück liefert.
sub	bkbk4711388	Auftraggeber bzw. Gegenstand (Subject) des JWT.
aud	["oam_client", "http://<AdminServerHost:Port>/oauth2"]	Zielgruppe, für welche das ID Token bestimmt ist.
exp	1675682269	Ablaufzeit-Zeitstempel, ab welchem der JWT NICHT mehr zur Verarbeitung angenommen werden MUSS. Siehe EPOCH-Zeitformat: https://www.unixtimestamp.com.
iat	1675678669	Ausstellungs-Zeitstempel, ab welcher der JWT zur Verfügung gestellt wurde.
jti	6YlFh6NKvGKWbT_LbtPZcw	JWT-ID enthält eindeutige Kennung für den JWT.
at_hash	OWcFB-SrvipUz_6cgpwIMQ	Base64URL-kodierter Hashwert der linken Hälfte (128 Bit) des SHA-256-Hashwert für den Token.
azp	oam_client	Client Kennung, die das Zugangs-Token verwenden und Ressourcen anfordern soll.
acr	2	"Authentication Context Class Reference" Wert bezugnehmend auf die Authentifizierungsebene.
sid	L0Qgkg3psFBepJT20UZgSA==~ UdZWe/4xflAR+g5uz12ITProzj2O i+K4x8Y70eXc8Rd9nhthe1nuZicp LyuQT+65CwJeE4wtctkqvNCrXA 4qLNxiKIDJGNbAhY30Bd1A0lS6Y RGyOazVuyOPTGpHJ44G	Enthält verschlüsselt Session-ID und Details.
auth-time	1675678618865	Zeitpunkt zu dem die Endbenutzer Authentifizierung erfolgt ist.
amr	pwd	"Authentication Method Reference" Wert bezugnehmend auf die Authentifizierungsmethode gemäß Spezifikation https://www.rfc-editor.org/rfc/rfc8176#section-2.

Discovery und UserInfo Endpoint

- **Discovery Endpoint**

  Steht zur Verfügung als OAuth2.0-Ressource, die Metadaten über den Autorisierungsserver zurückgibt.

```
http://<AdminServerHost:Port>/.well-known/opendid-configuration
```

  Beispiel Ergebnis:

```
{"issuer":"http://<AdminServerHost:Port>/oauth2",
"authorization_endpoint":"http://<AdminServerHost:Port>/oauth2/rest/authorize",
"token_endpoint":"http://<AdminServerHost:Port>/oauth2/rest/token",
"userinfo_endpoint":"http://<AdminServerHost:Port>/oauth2/rest/userinfo",
"introspect_endpoint":"http://<AdminServerHost:Port>/oauth2/rest/token/info",
"jwks_uri":"http://<AdminServerHost:Port>/oauth2/rest/security",
"end_session_endpoint":"http://<AdminServerHost:Port>/oauth2/rest/userlogout",
"scopes_supported":["openid","profile","email","address","phone"],
"response_types_supported":["code","token","id token","token id token"],
"grant_types_supported":["client_credentials","password","refresh_token","authorization_code","
implicit",
"urn:ietf:params:oauth:grant-type:jwt-bearer"],
"subject_types_supported":["public"],
"id_token_signing_alg_values_supported":["RS256"],
"userinfo_signing_alg_values_supported":["none"],
"token_endpoint_auth_methods_supported":["client_secret_basic","client_secret_jwt"],
"token_endpoint_auth_signing_alg_values_supported":["RS256"],
"claims_supported":["aud","exp","iat","iss","jti","sub"],
"ui_locales_supported":["en"]}
```

- **UserInfo Endpoint**

  Steht zur Verfügung als OAuth2.0-Ressource, die Angaben oder Informationen über den authentifizierten Benutzer gemäß **F-IAM / Autorisierungsserver** zurückgibt.
  Der Client kann mit übergebenem Access Token an den UserInfo Endpoint des Autorisierungsservers Angaben über den Endbenutzer anfordern.

  Beispiel cURL Befehl:

```
curl -X GET \
'http://<AdminServerHost:Port>/oauth2/rest/userinfo' \
-H 'Authorization: Bearer
eyJraWQiOiJPSURDRGVmYXVsdERvbWFpbiIsIng1dCI6Ims3U2Q5ckpycDR0WHd2S0x5Zm42U0hDQmk4ayIsImFsZyI6IlJ
TMjU2In0.eyJpc3MiOiJodHRwOi8vdndhc2UwNTEuemRzLmJrYS5idW5kLmRlOjc4MDcvb2F1dGgyIiwiYXVkIjoib2FtX2
NsaWVudCIsImV4cCI6MTY3NTY4MjI2OSwianRpIjoiV2pkZXk2ejk4TE5aRFVFSzFScG9JdyIsImlhdCI6MTY3NTY3ODY2O
Swic3ViIjoiYmtiazQ3MTEzODgiLCJjdXN0b21lQXR0cjEiOiJDdXN0b21lWYWx1ZSIsInNlc3Npb25JZCI6IjE2NmEyYWZi
LWIyODQtNDNhMC05M2FjLWQ0ZWU2ZTI5ZDNiZXxBcUxpa3ZiZlV1MUdvVlZRd0g0OERlc2JwbDgraFNYdmxxajNQMVZUeGw
0PSIsImNsaWVudCI6Im9hbV9jbGllbnQiLCJzY29wZSI6WyJvcGVuaWQiLCJlbWFpbCJdLCJkb21haW4iOiJPSURDRGVmYX
VsdERvbWFpbiJ9.Oy_MWlg8nAGzPQTL3Ps1tltbvs7vxQPTxzSL_P9byfjVKbhYTyqgr4_bO3RPfIGVwdCaMyLNFHs736yom
hTdGD99RaEABPBQJjfa8iDsRTIt1KuXXRdIQKIXjarBjuJPiXNK2_N98A-
kTjylelLtxhW488w5L96bZbxOwOFgo_QaoEPHsXQp5PRJre9IJ3gbATZCBRsAOcrfO8JExV0N_m1XibL-
VKtg4lhXmaNxyYICWwn0zn_7ptJyPc88ieHT3QldfhlUdU5WjN5Icng8qW6TjJnh7HJmqck9skyXFgHlOBEyNoEplCtOduV
5GrFY1vPh3VVRmX_Is8PzgH0vOAd2VibG9naWM6d21rYWgxbWRraA=='
```

  Beispiel Ergebnis zu o.a. Befehl:

```
{
    "sub": "Zitter, Alfons",
    "email": "alfons.zitter@vm.oracle.com",
    "email_verified": false
}
```

Dokumentation

Tabelle 5. Referenzen auf Produkt Dokumentation

THEMA	URL
OAM / OAuth & OpenIDConnect Services	https://docs.oracle.com/en/middleware/idm/access-manager/12.2.1.4/aiaag/managing-oracle-access-management-oauth-service-and-openidconnect.html
OAM / PKCE-Aktivierung	https://docs.oracle.com/en/middleware/idm/access-manager/12.2.1.4/aiaag/configuring-oauth-services-12c.html#GUID-D48FC8CC-653B-44AF-9E09-9182C7973D63
OAM / REST API	https://docs.oracle.com/en/middleware/idm/access-manager/12.2.1.4/oroau/index.html
OAM / OpenIDConnect ID-Token	https://docs.oracle.com/en/middleware/idm/access-manager/12.2.1.4/aiaag/understanding-openidconnect.html#GUID-47D0FE7E-1142-4F4A-AB9D-0106FC4AC984
OpenIDConnect Claims	https://docs.oracle.com/en/middleware/idm/access-manager/12.2.1.4/aiaag/understanding-openidconnect.html#GUID-08397858-DD58-410D-80F9-E6249C269236