4. TechEx2019 - Attribute Management

Attribute Definitions

Section Topics

- OIDC user attributes possible locations
- OIDC Attribute resolving principle, authorization, token and userinfo endpoints
- OIDC attribute encoders

OIDC user attributes possible locations

User attributes may be returned either in ID Token http://openid.net/specs/openid-connect-core-1_0.html#IDToken or in UserInfo response http://openid.net/specs/openid-connect-core-1_0.html#UserInfoResponse.

ID Token - Depending on response type is returned from Authentication or from Token endpoint.

```
{
 kid: "1e9gdk7",
 alg: "RS256"
}.
{
 iss: "http://server.example.com",
 sub: "248289761001",
 aud: "s6BhdRkqt3",
 nonce: "n-0S6_WzA2Mj",
 exp: 1311281970,
 iat: 1311280970,
 name: "Jane Doe",
 given_name: "Jane",
 family_name: "Doe",
 gender: "female",
 birthdate: "0000-10-31",
 email: "janedoe@example.com",
 picture: "http://example.com/janedoe/me.jpg"
}.
[signature]
```

UserInfo response - Always from UserInfo endpoint.

```
 {
   "sub": "248289761001",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "preferred_username": "j.doe",
   "email": "janedoe@example.com",
   "picture": "http://example.com/janedoe/me.jpg"
  }
```

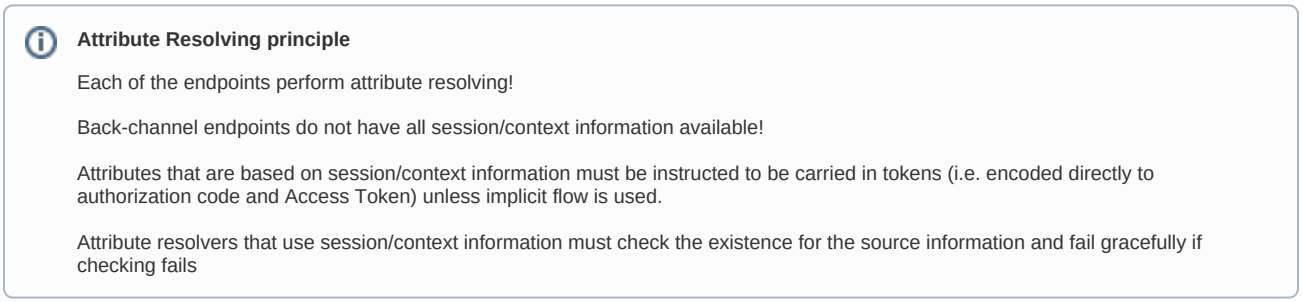
Whether the attribute ends up to the ID Token or UserInfo response depends on authentication request.

- If claim is requested by standard scope parameter, attribute will be set to userinfo response unless response type is "id_token" (i.e. UserInfo endpoint cannot be accessed).
- Client may specifically request claim to be returned either in ID Token or in UserInfo response
- Third option is of course some agreement between RP and OP about non standard scope or some out of band agreement.

> How clients request for claims is handled in more detail in section about attribute filtering

OIDC Attribute resolving principle, authorization, token and userinfo endpoints.

- Authentication endpoint. Front-channel endpoint responsible of authenticating the user are asking for consent. May return depending on response type ID Token, authorization code or Access Token or any combination of the three items.
- Token endpoint. Back-channel endpoint that is responsible of authenticating the client RP. Swaps authorization code(or Refresh Token) to Access Token, ID Token and to Refresh Token.
- Userinfo endpoint. Back-channel endpoint that can be accessed with Access Token returning UserInfo response.



Attribute Resolving principle

Each of the endpoints perform attribute resolving!

Back-channel endpoints do not have all session/context information available!

Attributes that are based on session/context information must be instructed to be carried in tokens (i.e. encoded directly to authorization code and Access Token) unless implicit flow is used.

Attribute resolvers that use session/context information must check the existence for the source information and fail gracefully if checking fails

Resolver example

```
    <AttributeDefinition id="password" xsi:type="ScriptedAttribute" dependencyOnly="true" language="
nashorn">
        <Script><![CDATA[
            logger = Java.type("org.slf4j.LoggerFactory").getLogger("net.shibboleth.idp.script.
password");
            subjectCtx = profileContext.getSubcontext("net.shibboleth.idp.authn.context.SubjectContext");
            subject = subjectCtx.getSubjects()[0];
            if (!subject.getPrivateCredentials().isEmpty()){
                password.addValue(subject.getPrivateCredentials().toArray()[0].getName());
            }
    ]]></Script>
    </AttributeDefinition>
```

OIDC attribute encoders

- oidcext:OIDCString, for encoding an IdPAttribute with simple string values as JSON Object.
- oidcext:OIDCScopedString, for encoding an IdPAttribute with scoped string values as JSON Object.
- oidcext:OIDCByte, for encoding an IdPAttribute with binary values as JSON Object.

Encoder formatting options

Default

```
<AttributeEncoder xsi:type="oidcext:OIDCString" name="affiliation" />

Input: IdPAttribute["member", "staff"]
Output: "affiliation": "member staff"
```

asArray

```
<AttributeEncoder xsi:type="oidcext:OIDCString" asArray="true" name="affiliation" />

Input: IdPAttribute["member", "staff"]
Output: "affiliation":["member","staff"]
```

asInt

```
<AttributeEncoder xsi:type="oidcext:OIDCString" asInt="true" name="updated_at" />

Input: IdPAttribute["1536143427"]
Output: "updated_at": 1536143427
```

asBoolean

```
<AttributeEncoder xsi:type="oidcext:OIDCString" asBoolean="true" name="email_verified" />

Input: IdPAttribute["true"]
Output: "email_verified": true
```

asObject

```
<AttributeDefinition id="address" xsi:type="ScriptedAttribute">
    <Dependency ref="staticAttributes" />
    <Script><![CDATA[address.addValue("{\"street_address\":\""+street_address.getValues().get(0) + "\","
        +"\"locality\":\""+locality.getValues().get(0) + "\","
        +"\"region\":\""+region.getValues().get(0) + "\","
        +"\"postal_code\":\""+postal_code.getValues().get(0) + "\","
        +"\"country\":\""+country.getValues().get(0) + "\"}");]]></Script>
    <AttributeEncoder xsi:type="oidcext:OIDCString" asObject="true" name="address" />
</AttributeDefinition>


Output: "address":{"street_address":"234 Hollywood Blvd.","country":"US","locality":"Los Angeles","region":"
CA","postal_code":"90210"}
```

Encoder delivery options

placeToIDToken & denyUserinfo

```
<!-- This demonstrates a claim that is placed always to id token -->
<AttributeDefinition id="email_idtoken" xsi:type="Simple" sourceAttributeID="email">
    <Dependency ref="email" />
    <AttributeEncoder xsi:type="oidcext:OIDCString" placeToIDToken="true" denyUserinfo="true" name="email" />
</AttributeDefinition>
```

setToToken

Attribute not resolvable in token and UserInfo endpoints must be carried in tokens if necessary.

Resolver example

```
     <AttributeDefinition id="password" xsi:type="ScriptedAttribute" dependencyOnly="true" language="nashorn">
        <Script><![CDATA[
            logger = Java.type("org.slf4j.LoggerFactory").getLogger("net.shibboleth.idp.script.password");
            subjectCtx = profileContext.getSubcontext("net.shibboleth.idp.authn.context.SubjectContext");
            <!-- Subject does not have credentials populated in Token and UserInfo endpoints, have to give up
in such case -->
            if (subjectCtx!=null){
                subject = subjectCtx.getSubjects()[0];
                password.addValue(subject.getPrivateCredentials().toArray()[0].getName());
            }
        ]]></Script>
     <AttributeEncoder xsi:type="oidcext:OIDCString" setToToken="true" name="password"/>
    </AttributeDefinition>
```

Subject Identifier

Locally unique and never reassigned identifier within the Issuer for the End-User, which is intended to be consumed by the Client. It MUST NOT exceed 255 ASCII characters in length. The value is a case sensitive string.

Two types of Subject Identifiers, public and pairwise, http://openid.net/specs/openid-connect-core-1_0.html#SubjectIDTypes

Different RPs may belong to same pairwise group by sharing the sector_identifier_uri - value

Placed always in ID Token and UserInfo response, whether requested or not.

Subject Identifier Generation

Extension provides default resolver and a configuration for it.

oidc-subject.properties

```
# The source attribute used in generating the subject
idp.oidc.subject.sourceAttribute = uid

# The digest algorithm used in generating the subject
#idp.oidc.subject.algorithm = SHA

# The encoding used in generating the subject
#idp.oidc.subject.encoding = BASE32

# The salt used in generating the subject
# Do *NOT* share the salt with other people, it's like divulging your private key.
idp.oidc.subject.salt = this_too_should_be_ch4ng3dExercises
```

attribute-resolver.xml

```
    <!-- Subject Identifier is a attribute that must always be resolved.
    There has to be exactly one resolved and filtered attribute that would be encoded as 'sub'.
    This example attribute (the data connector actually ) will generate public or pairwise 'sub' depending on
client registration data.  -->

    <AttributeDefinition id="subject" xsi:type="Simple" activationConditionRef="SubjectRequired">
        <InputDataConnector ref="computedSubjectId" attributeNames="subjectId"/>
        <AttributeEncoder xsi:type="oidcext:OIDCString" name="sub" />
    </AttributeDefinition>

    <!--
    Subject Identifier is a attribute that must always be resolved.
    There has to be exactly one resolved and filtered attribute that would be encoded as 'sub'.

    Use activation conditions and filters to ensure the requirement is met if you have need for several
different kind of formats for 'sub'.

    <AttributeDefinition id="subject-public" xsi:type="Simple" sourceAttributeID="uid" activationConditionRef="
PublicRequired">
        <Dependency ref="uid" />
        <AttributeEncoder xsi:type="oidcext:OIDCString" name="sub" />
    </AttributeDefinition>

    <AttributeDefinition id="subject-pairwise" xsi:type="Simple" activationConditionRef="PairwiseRequired">
        <InputDataConnector ref="computedSubjectId" attributeNames="subjectId"/>
        <AttributeEncoder xsi:type="oidcext:OIDCString" name="sub" />
    </AttributeDefinition>
    -->


   <!-- Data Connector for generating 'sub' claim.
         The connector may be used to generate both public and pairwise subject values -->
    <DataConnector id="computedSubjectId" xsi:type="ComputedId"
            generatedAttributeID="subjectId"
            sourceAttributeID="%{idp.oidc.subject.sourceAttribute}"
            salt="%{idp.oidc.subject.salt}"
            algorithm="%{idp.oidc.subject.algorithm:SHA}"
            encoding="%{idp.oidc.subject.encoding:BASE32}">
            <Dependency ref="%{idp.oidc.subject.sourceAttribute}"/>
    </DataConnector>
```

attribute-filter.xml

```
<AttributeFilterPolicy id="OPENID_SCOPE">
    <PolicyRequirementRule xsi:type="oidcext:OIDCScope" value="openid" />
    <AttributeRule attributeID="subject">
        <PermitValueRule xsi:type="ANY" />
    </AttributeRule>
</AttributeFilterPolicy>
```

Attribute Filtering

Section Topics

- Requesting Attributes in OIDC
- Filtering attributes for OIDC RPs

Requesting Attributes in OIDC

There are two main mechanisms for RP to ask for attributes,

- Authentication request scope - parameter
  - There is a set of standard scope values profile, email, address and phone.
  - The claims requested by the standard scope values are expected to be returned from UserInfo endpoint unless response type is "id_token"
  - The claims requested by the standard scope values are treated as voluntary
  - You may define your own scope values and rules for them. This has to be agreed by both parties of course.
- Authentication request claims - parameter
  - You must declare the target for the claim, ID Token or UserInfo response
  - You may list the claim as essential claim
  - You may list a expected value for the claim

Claims Request

```
{
   "userinfo":
    {
     "given_name": {"essential": true},
     "nickname": null,
     "picture": null,
     "http://example.info/claims/groups": null
    },
   "id_token":
    {
     "email": {"essential": true},
     "email_verified": {"essential": true},
     "sub": {"value": "248289761001"}
     "auth_time": {"essential": true},
     "acr": {"essential": true,
             "values": ["urn:mace:incommon:iap:silver",
                        "urn:mace:incommon:iap:bronze"]}
    }
  }
```

Filtering attributes for OIDC RPs

It is your job to resolve and filter attributes to match the requests. OIDC extension has reserved the right to control some claims you should not try to resolve or filter.

oidcext:OIDCScope

PolicyRule which returns true if any of the scope values in the authentication request matches a supplied string.

OIDCScope

```
 <!-- This demonstrates a rule that releases email claims in response to all oidc authentication requests
having scope
        email. The requester needs to have scope email as a registered scope. -->

<AttributeFilterPolicy id="OPENID_SCOPE_EMAIL">
    <PolicyRequirementRule xsi:type="oidcext:OIDCScope" value="email" />
    <AttributeRule attributeID="email">
        <PermitValueRule xsi:type="ANY" />
    </AttributeRule>
    <AttributeRule attributeID="email_verified">
        <PermitValueRule xsi:type="ANY" />
    </AttributeRule>
</AttributeFilterPolicy>
```

oidcext:AttributeInOIDCRequestedClaims

Matcher which returns attribute values after comparing them to requested claims parameter of oidc authentication request.

AttributeInOIDCRequestedClaims

```
 <!-- This demonstrates a rule that releases email in id token if specifically asked to be released as
essential for id token -->

<AttributeFilterPolicy id="REQUESTED_CLAIMS">
    <PolicyRequirementRule xsi:type="ANY" />
    <AttributeRule attributeID="email_idtoken">
        <PermitValueRule xsi:type="oidcext:AttributeInOIDCRequestedClaims" matchOnlyIDToken="true"
onlyIfEssential="true" />
    </AttributeRule>
</AttributeFilterPolicy>
```

Exercises

Exercise 4.1

Define a new scope "campus". Target is to have scope "campus" that includes a claim "campus_id" to ID Token.

1. Add "campusId" attribute definition to attribute resolver and add a value for it in staticAttributes data connector. Add a new filtering rules that releases campusId when scope "campus" is requested.

   Hints, Tips and Result

   ```
   nano /opt/shibboleth-idp/conf/attribute-resolver.xml

   <AttributeDefinition id="campusId" xsi:type="Simple" sourceAttributeID="campusId">
       <Dependency ref="staticAttributes" />
       <AttributeEncoder xsi:type="oidcext:OIDCString" name="campus_id" />
   </AttributeDefinition>

   <!-- Find the existing data connector element and add a new attribute -->
   <DataConnector id="staticAttributes" xsi:type="Static">
   ....
       <Attribute id="campusId">
           <Value>New Campus</Value>
       </Attribute>
   ....
   </DataConnector>


   nano /opt/shibboleth-idp/conf/attribute-filter.xml

   <AttributeFilterPolicy id="OPENID_SCOPE_CAMPUS">
       <PolicyRequirementRule xsi:type="oidcext:OIDCScope" value="campus" />
       <AttributeRule attributeID="campusId">
           <PermitValueRule xsi:type="ANY" />
       </AttributeRule>
   </AttributeFilterPolicy>
   ```

2. Modify client RP to request for scope "campus".

   ```
   nano +643 /etc/httpd/conf.d/auth_openidc.conf

   OIDCScope "openid campus"

   service httpd restart
   ```

3. Authenticate the user. Verify from the logs that scope "campus" is being requested. Find out from the logs why it is not being released.

Hints, Tips and Result

```
[root@gn43-oidcshibop-devel conf]# grep campus  /opt/shibboleth-idp/logs/idp-process.log
2018-09-24 05:27:27,048 - INFO [net.shibboleth.idp.attribute.resolver.spring.BaseResolverPluginParser:
63] - Parsing configuration for AttributeDefinition plugin with id: campusId
    scope:openid campus
2018-09-24 05:40:41,506 - DEBUG [org.geant.idpextension.oidc.decoding.impl.
OIDCAuthenticationRequestDecoder:69] - Decoded inbound request query string login_hint=teppo%
40192.168.0.150&scope=openid+campus&claims=%7B%22id_token%22%3A%7B%22acr%22%3A%7B%22values%22%3A%5B%
22urn%3Amace%3Aincommon%3Aiap%3Asilver%22%2C%22urn%3Amace%3Aincommon%3Aiap%3Abronze%22%5D%2C%
22essential%22%3Atrue%7D%7D%7D&response_type=code&redirect_uri=https%3A%2F%2F192.168.0.150%3A8443%
2Fprotected%
2Fredirect_uri&state=GfF7nJQf2lEQZIvFOj9UZMCxLYY&nonce=7XWKLlplp2TNt5gHCHS4QGg9XgglPzXGSMNcXyRT1IM&clie
nt_id=_6abd8ce1be06caf6c65b79934dfd5a01
2018-09-24 05:40:41,768 - WARN [org.geant.idpextension.oidc.profile.impl.ValidateScope:93] - Profile
Action ValidateScope: removing requested scope campus for rp _6abd8ce1be06caf6c65b79934dfd5a01 as it
is not a registered one
```

4. Add scope "campus" for the RP as a valid scope value.

Hints, Tips and Result

```
nano /opt/shibboleth-idp/metadata/oidc-client.json

[
  {
    "scope":"openid phone campus",
    "redirect_uris":["https://demorp.example.com/redirect_uri"],
    "client_id":"demo_rp",
    "client_secret":"topsecret",
    "response_types":["code"],
    "grant_types":["authorization_code","implicit","refresh_token"]
  }
]
```

5. Authenticate the user. Verify from the logs that scope "campus" is being requested. Find out from the logs is it being released to ID Token or to UserInfo response.

Hints, Tips and Result

```
grep id_token /opt/shibboleth-idp/logs/idp-process.log

Content:{"access_token":"AAdzZWNyZXQx6kaWFa5m2rBYllmrWmypRG6o8acq7EsttE-kkEqpF_llEOeZ-RWze1B-
p18f3rVWUH4dC_DQP8OHkoYbJuCBHpKdEOSNsmjCvO7MElb_p5VeiiyUqtVAcYnjWe0CAUF0w_lGHKndleYiWjzO_EhGuvH-zwui-
HgvS65A3VNUYtfbUt6fxm-
yC8KoVBfN7TjWhIu6wV_lhUFZuNWpI4HMhwLhV4vqcpAyVwqPmUJzcDqOT2ltktMGImeZmDl3a_rAvfZhF2VZc6K9xdEcu6NEMbasbQ
gCutWJkvrTfkBgMspdGn2_YYapxOWImkMS4s0fYEEdcwMB8nxQRKMpP_JmiV5hIMMFiZSg4qa5J9N2t5oI_F-
vO2nG5Q7bBMyPmvLCcBcXPOVbP2fqZyu_hCcxBwYjnwURYlp1jN-
V9dtG_vdRa5Ddn3P0QcrJQDTQDrAsnkn51NB24b56CiQjj4kVFrc4qNpeDhQ7dgG3M19BKD1rdoFMaNTStN0_Br3YD2G0B4GDJX3I6U
JOM4ksmbqDb5aknDgSMxnJHDt96LQqoNmDNI-RugE3Ombit9ZAUaRuAXMAKvR6gXK5gfhr8LzqflF8k2DTVLPyvI1q","
refresh_token":"AAdzZWNyZXQxvJQv2UDhK1QGie4smtxaYDD9uwFMk09syFcROFj851zrNb1a7UXBSmp6VYJjYHsXyyIBe0-
BQ4_vA7WL-Yuc1u-rJ951w3uKytERn87s0ZGTv45mCVpvlisLpL_XAu6x6Jh7DpOCP7b1FZYH2ctWIYx-
Vo9QLya1b1Zys3fohU6Rg5O24ZvtTuUJugFPyDOdaCjO5tTP2eSVSEGrm8-
NFz_z0VLWdqwCCjhPKq3FKSXmDo6UX1wyxLtH0DVLoBrAJrZtHMrjsNI1jJQY6IeNl_yxN6Hl_0FNmhmbEgD8zMT0N9-
zlrno4w7NPHBQK5ytFCXixmXEK9xgjlf3uqoLQMVMuTuMvee6hQNgRzawcYU3R5jN9KtCNN_NR0v5n8-
R3QTlHlYn7Agdhm1UpEuKomXFCy9fo9Psq2haOj588Yt0YaHqp877bofP5SjaNVSixC7TECwAjc11yg2JhRSPGUHQwlRzFuVhUdpVgd
NZfSD70nrdAMUv_--XVEdzg9vnCNjPk7XE3jjVijvugaVZvUV2HO6RnhQRWCg12lfOROmBq0bUlaMPaC87zwBlEI9GjQOG_e1fw-
K3BfcC7WdQ9DVEtPuOrRh0qA0","id_token":"eyJraWQiOiJ0ZXN0a2V5UlMiLCJhbGciOiJSUzI1NiJ9.
eyJhdF9oYXNoIjoib1RBc3dNX2pxR0pncHo4WDkzRWNydyIsInN1YiI6IlZVRzQ3NzdZUDNOTVU1S1JGRVNYNlNLUkFQWExFNE1JIiw
iYXVkIjoiX2ZhYmJjNmY5MGExMmZmYjNmZWUwMDEwZDRjMmFiZGVlIiwiYWNyIjoidXJuOm1hY2U6aW5jb21tb246aWFwOnNpbHZlci
IsImF1dGhfdGltZSI6MTUzNzc3Mjc4NiwiaXNzIjoiaHR0cHM6XC9cLzE5Mi4xNjguMC4xNTAiLCJleHAiOjE1Mzc3NzYzODcsImlhd
CI6MTUzNzc3Mjc4Nywibm9uY2UiOiJ3NnJzdkh2d05UX0FuSVZ3OFh1UTlucG5iLXFBZ18yenEySTdtUEJMNkdNIn0.
Iqraic8ugywq2_saSoIGIBCZFJIArgUiVvnRQxyjemmZdUG7hxU7lJ06GogBGNmnKDwhOqiA-
Ck1SB2iVkvA1YbK3xGgCBO4kkYQSWGjQ1Mq1R-
3J0_OPZ5SU5pjcpo4YUBGqYpOjv2WDAftfpBgGr0HmXRjTd_CLxcm8AM5XMwokCJufqoP7l6fjAhxcKweQNPnGEASCG200wTNHIEA82
wvOHxZ9BrjB65kyM2lODL40T9NJ5UVvAGHzMzrcw2PUHrhnBGKLmGe89oSO0ww3KfxDiU7vg2jg36xnoMUVwN8RFad7-
fNzH3JwNEMzekQBuusDxrxAHlaVHntTZndrw","token_type":"Bearer","expires_in":600}

Decode id token, you will get something like:

{
 kid: "testkeyRS",
 alg: "RS256"
}.{
 at_hash: "oTAswM_jqGJgpz8X93Ecrw",
 sub: "VUG4777YP3NMU5KRFESX6SKRAPXLE4MI",
 aud: "_fabbc6f90a12ffb3fee0010d4c2abdee",
 acr: "password",
 auth_time: 1537772786,
 iss: "https:\/\/192.168.0.150",
 exp: 1537776387,
 iat: 1537772787,
 nonce: "w6rsvHvwNT_AnIVw8XuQ9npnb-qAg_2zq2I7mPBL6GM"
}.[signature]

Check now userinfo response. It is one of the last lines in the log now

tail /opt/shibboleth-idp/logs/idp-process.log

Content:{"sub":"VUG4777YP3NMU5KRFESX6SKRAPXLE4MI","campus_id":"Y2FtcHVzSWQ="}

You notice the campus_id is only in the userinfo response.
```

6. Modify "campusId" attribute resolver to encode the claim always and only to ID Token.

Hints, Tips and Result

```
<AttributeDefinition id="campusId" xsi:type="Simple" sourceAttributeID="campusId">
    <Dependency ref="staticAttributes" />
    <AttributeEncoder xsi:type="oidcext:OIDCString" name="campus_id" placeToIDToken="true"
denyUserinfo="true"/>
</AttributeDefinition>
```

7. Authenticate the user. Verify from the logs that scope "campus" is being requested. Verify from the logs it is being released only to ID Token.

Hints, Tips and Result

```
grep id_token /opt/shibboleth-idp/logs/idp-process.log

Content:{"access_token":"AAdzZWNyZXQxX6FZjPLkXdPUi2UBqUNnkcNwayEbDbrsL_LHumwkbTpLM2W8QVEOGhHRsh-
sAtKeaIFx1568_AAESKeHT7qFSJYKOavGxj17wVBJ_S8mQGW7YcIIfL2hAC-LnHtvCgY8n_r-
WsD_HVJN3UqQ0GijbPeX4NHEENfBgG18oEO1bbSMHR0XhBCY0sLcubfaXl7N0E17COObm992Su5jFsvZptfgmYWgkV3OYzX4684bNL2
F_8-6w1TblGpNgXIi0ORci7FSXP0oMQfxb_AeAfADe3n-tl_4dUeJ-rrKP-oG-rUsnkEQQqHUgtBZEIV-
YmIYhA3UP1Ux35XpFYwrbMYSb7yZ2zeH-
fiukZGiStMxoifVmldzeKTZ8euO69lcZqv1WZGWueMmZ3ltfCWH3V5XI26KjYJgnxaX9FGVG4wRBm_yCclgtuplscs8Ixvmrr4HyaSQ
ZOGOS0d8Ri4q6KnKqaob4zCAX2GGkBCWCtYQpTKfotLWWwvNorxhPlyVoorOLDKe2eTl3en4H9RGl_xvaYZTIMyu0bB-
eUJgeXyskz_TEjDpnnBoSSBSO2daBNZPnsQUaLnz5E0RK_C7it73hc2TWrNXWkA7pA","refresh_token":"
AAdzZWNyZXQxjINhPCFGNntDfeWtxJhGEToRHe3CwPT3WTQJO-
7GQBmNHSGiK3dzGPILulNKHI8Akhf7CZiYePeB5AEyV2mFvF5142QjmG0hao_pyH08WyCI9m9XSrGR7QxhzYPdKgShAAoJUyLkNBjsw
V2B0HsW2dg6tys4WZW7bMPOSmJ_o6mIFuuG6umzWUJkKSyqDcuvfUBQdNKZ73UHYMDheuTeZ8VrF9-
6CcUsBkG7EEEwIogpJ4tXGUcstZ-3lKeEf3RR68Ag7Okoy7G-
QqBL28KtCOKTery5iqB5ZbKGqxoGDfHEe1c5TqVbRp7pvvISTMVx8EgDuFoUnUC0IGAoFoFMf4WH0fCwN4kQL2WRfsYiyoHliC0aw2n
fI2gV-rciO9K26kj6mmxBf71urV2HKmjdJg8cR-V3m5T80D94e14jJFewyFs1_3XYgbkDu5CeeptesgObLBYy0oYCLEqxt4ht_-
fxNqBmsrnganoIn4tYRzXDyTkUXJv8w-
foFfkiC484crWlutAEbXzAKipDn2t43k9HOHFdM3nofw4CnY4YHeRMsOphIFEHXSYOgBa73zr4dzdR5If9ivLcuuRNe7lfmaEEhwzcq
llG-FDU9s0GQA","id_token":"eyJraWQiOiJ0ZXN0a2V5UlMiLCJhbGciOiJSUzI1NiJ9.
eyJhdF9oYXNoIjoiWjlPNjRkYVFUbXhKeE9obmZtdkl2USIsInN1YiI6IlZVRzQ3NzdZUDNOTVU1S1JGRVNYNlNLUkFQWExFNE1JIiw
iYXVkIjoiX2ZhYmJjNmY5MGExMmZmYjNmZWUwMDEwZDRjMmFiZGVlIiwiYWNyIjoidXJuOm1hY2U6aW5jb21tb246aWFwOnNpbHZlci
IsImF1dGhfdGltZSI6MTUzNzc3MzIyMCwiaXNzIjoiaHR0cHM6XC9cLzE5Mi4xNjguMC4xNTAiLCJleHAiOjE1Mzc3NzY4MjQsImlhd
CI6MTUzNzc3MzIyNCwibm9uY2UiOiJfcDdCX3lWeDlsQUhEelJQcGhnWkNRa2JYdmNnMC1UR0JqTm9ONVZvdTlrIiwiY2FtcHVzX2lk
IjoiWTJGdGNIVnpTV1E9In0.
ejxpf9N3LW4UqMlbB92F2_DUSNyVwHUckCPYNGO11WxkGRFd9mOKDC1m217f6ezQmZceyTT7rckkmWhWGDDQeJoxKJGzji0tmSHzZU6
2FAm32zxRL49rlmRT-cBpfH7cpAMjmhi8ORjci8pER7M205IL2ANXzM_q8RnLgtS9r4ztm9Xpmu83bPea-
y4Qqm1phpsejPbIanDNZ7WmrChD26_lrSid8dOLu_VTta3Mgh_BcF0sQbnvxSpz5VQIJBHp7vlfxuipfHPMrc96Pqj_6pksQHQjvBOG
r9yLIXF6lO6ja7-8dTRMlOhvTae2tbec62ptepLK5_OSgOFT_GwhEQ","token_type":"Bearer","expires_in":600}

Decode id token, you will get something like:

{
 kid: "testkeyRS",
 alg: "RS256"
}.{
 at_hash: "Z9O64daQTmxJxOhnfmvIvQ",
 sub: "VUG4777YP3NMU5KRFESX6SKRAPXLE4MI",
 aud: "_fabbc6f90a12ffb3fee0010d4c2abdee",
 acr: "password",
 auth_time: 1537773220,
 iss: "https:\/\/192.168.0.150",
 exp: 1537776824,
 iat: 1537773224,
 nonce: "_p7B_yVx9lAHDzRPphgZCQkbXvcg0-TGBjNoN5Vou9k",
 campus_id: "New Campus"
}.[signature]

Check now userinfo response. It is one of the last lines in the log now

tail /opt/shibboleth-idp/logs/idp-process.log

Content:{"sub":"VUG4777YP3NMU5KRFESX6SKRAPXLE4MI"}

You notice the campus_id is only in the id token.
```

Exercise 4.2

Define attribute release rules to release "campusId" attribute to be released if asked to be released for ID Token as essential claim.

1. Make sure "campusId" is not requested anymore by scope.

```
nano +643 /etc/httpd/conf.d/auth_openidc.conf

OIDCScope "openid"
```

2. Modify RP to ask "campusId" as essential ID Token claim.

```
nano +417 /etc/httpd/conf.d/auth_openidc.conf

OIDCAuthRequestParams claims=%7B%22id_token%22%3A%7B%22campus_id%22%3A+%7B%22essential%22%3A+true%7D%7D%7D

service httpd restart
```

3. Remove or comment the campus scope filter rule that you added in exercise 4.1.1. Add a new filtering rule that will release "campusId" as a claim only if requested to be released as essential ID Token claim

Hints, Tips and Result

```
<AttributeFilterPolicy id="REQUESTED_CAMPUS_CLAIMS">
    <PolicyRequirementRule xsi:type="ANY" />
    <AttributeRule attributeID="campusId">
        <PermitValueRule xsi:type="oidcext:AttributeInOIDCRequestedClaims" matchOnlyIDToken="true"
onlyIfEssential="true" />
    </AttributeRule>
</AttributeFilterPolicy>
```

4. Authenticate the user and verify from the logs the attribute is released. At this point you should be able to do it without hints and tips.

Exercises

Exercise 4.3

Available contexts in Endpoints

1. Add the following attribute definition to attribute-resolver.xml. Resolver changes require restarting Shibboleth IdP

```
nano /opt/shibboleth-idp/conf/attribute-resolver.xml

<AttributeDefinition id="scriptedAuthenticationFlowId" xsi:type="ScriptedAttribute" language="nashorn">
    <Script><![CDATA[
    logger = Java.type("org.slf4j.LoggerFactory").getLogger("net.shibboleth.idp.script.password");
    authnCtx = profileContext.getSubcontext("net.shibboleth.idp.authn.context.AuthenticationContext");
    scriptedAuthenticationFlowId.addValue(authnCtx.getAuthenticationResult().
getAuthenticationFlowId());
    ]]></Script>
    <AttributeEncoder xsi:type="oidcext:OIDCString" setToToken="true" name="flow_id"/>
</AttributeDefinition>
```

2. Authenticate user using with this new configuration. We are not creating release rules for the attribute, yet. See log for any errors related to the attribute.

Hints, Tips and Result

```
2018-09-20 09:50:48,070 - ERROR [net.shibboleth.idp.profile.impl.ResolveAttributes:314] - Profile
Action ResolveAttributes: Error resolving attributes
net.shibboleth.idp.attribute.resolver.ResolutionException: Attribute Definition
'scriptedAuthenticationFlowId':Script did not run successfully
    at net.shibboleth.idp.attribute.resolver.ad.impl.
ScriptedAttributeDefinition$AttributeDefinitionScriptEvaluator.execute(ScriptedAttributeDefinition.
java:228)
Caused by: java.lang.RuntimeException: javax.script.ScriptException: TypeError: null has no such
function "getAuthenticationResult" in <eval> at line number 3
    at net.shibboleth.utilities.java.support.scripting.AbstractScriptEvaluator.evaluate
(AbstractScriptEvaluator.java:193)
Caused by: javax.script.ScriptException: TypeError: null has no such function
"getAuthenticationResult" in <eval> at line number 3
    at jdk.nashorn.api.scripting.NashornScriptEngine.throwAsScriptException(NashornScriptEngine.java:
470)
Caused by: jdk.nashorn.internal.runtime.ECMAException: TypeError: null has no such function
"getAuthenticationResult"
```

3. Change the response type to "id_token". It means we are using only authentication endpoint.

```
nano +636 /etc/httpd/conf.d/auth_openidc.conf

OIDCResponseType "id_token"

service httpd restart
```

4. Authenticate user using with this new configuration. Notice there are not errors related to resolving the attribute. Can you explain why?
5. Change the response type back to "code". Correct the attribute to not cause errors. Correct also the attribute to be carried in tokens.

Hints, Tips and Result

```
<AttributeDefinition id="scriptedAuthenticationFlowId" xsi:type="ScriptedAttribute" language="nashorn">
    <Script><![CDATA[
    logger = Java.type("org.slf4j.LoggerFactory").getLogger("net.shibboleth.idp.script.password");
    authnCtx = profileContext.getSubcontext("net.shibboleth.idp.authn.context.AuthenticationContext");
    if (authnCtx != null){
        scriptedAuthenticationFlowId.addValue(authnCtx.getAuthenticationResult().
getAuthenticationFlowId());
    }
    ]]></Script>
    <AttributeEncoder xsi:type="oidcext:OIDCString" setToToken="true" name="flow_id"/>
</AttributeDefinition>
```

Exercise 4.4

Formatting attribute

1. Create a new attribute attribute called "manipe" with static values "zero", "1", "3" and "two".

   Hints, Tips and Result

```
...

    <AttributeDefinition id="manipe" xsi:type="Simple" sourceAttributeID="manipe">
        <Dependency ref="staticAttributes" />
        <AttributeEncoder xsi:type="oidcext:OIDCString" name="manipe" />
    </AttributeDefinition>

...

   <DataConnector id="staticAttributes" xsi:type="Static">
        <Attribute id="manipe">
            <Value>zero</Value>
            <Value>1</Value>
            <Value>3</Value>
            <Value>two</Value>
        </Attribute>
...
```

2. Update the filtering rules to release it all RPs asking for profile scope.

   Hints, Tips and Result

```
...

    <AttributeFilterPolicy id="OPENID_SCOPE_PROFILE">
        <PolicyRequirementRule xsi:type="oidcext:OIDCScope" value="profile" />
        <AttributeRule attributeID="manipe">
            <PermitValueRule xsi:type="ANY" />
        </AttributeRule>
...
```

3. Authenticate the user. Locate "manipe" claim from the UserInfo response.

   Hints, Tips and Result

```
Headers:
    Content-Type:application/json; charset=UTF-8
Content:{"sub":"VUG4777YP3NMU5KRFESX6SKRAPXLE4MI","zoneinfo":"America\/Los_Angeles","website":"https:\
/\/www.facebook.com\/officialtomcruise\/","birthdate":"1962","address":{"street_address":"234
Hollywood Blvd.","country":"US","locality":"Los Angeles","region":"CA","postal_code":"90210"},"
email_verified":false,"gender":"male","profile":"https:\/\/fi.wikipedia.org\/wiki\/Tom_Cruise","
phone_number_verified":true,"preferred_username":"ttester","locale":"en-US","given_name":"Teppo
Matias","middle_name":"Matias","manipe":"zero 1 3 two","picture":"https:\/\/pixabay.com\/fi\/pentu-
kissa-kukka-potin-tabby-pentu-2766820\/","updated_at":1509450347,"nickname":"TT","name":"Mr.Teppo
Matias Testaaja","phone_number":"+1 (604) 555-1234;ext=5678","family_name":"Testaaja","email":"
teppo@example.org"}
```

4. Instruct "manipe" to be encoded as an array. Verify the result from the UserInfo response.

Hints, Tips and Result


```
Headers:
    Content-Type:application/json; charset=UTF-8
Content:{"sub":"VUG4777YP3NMU5KRFESX6SKRAPXLE4MI","zoneinfo":"America\/Los_Angeles","website":"https:\
/\/www.facebook.com\/officialtomcruise\/","birthdate":"1962","address":{"street_address":"234
Hollywood Blvd.","country":"US","locality":"Los Angeles","region":"CA","postal_code":"90210"},"
email_verified":false,"gender":"male","profile":"https:\/\/fi.wikipedia.org\/wiki\/Tom_Cruise","
phone_number_verified":true,"preferred_username":"ttester","locale":"en-US","given_name":"Teppo
Matias","middle_name":"Matias","manipe":["zero","1","3","two"],"picture":"https:\/\/pixabay.com\/fi\
/pentu-kissa-kukka-potin-tabby-pentu-2766820\/","updated_at":1509450347,"nickname":"TT","name":"Mr.
Teppo Matias Testaaja","phone_number":"+1 (604) 555-1234;ext=5678","family_name":"Testaaja","email":"
teppo@example.org"}
```


5. Instruct "manipe" to be encoded as an array of integers. Verify the result from the UserInfo response.

Hints, Tips and Result


```
Headers:
    Content-Type:application/json; charset=UTF-8
Content:{"sub":"VUG4777YP3NMU5KRFESX6SKRAPXLE4MI","zoneinfo":"America\/Los_Angeles","website":"https:\
/\/www.facebook.com\/officialtomcruise\/","birthdate":"1962","address":{"street_address":"234
Hollywood Blvd.","country":"US","locality":"Los Angeles","region":"CA","postal_code":"90210"},"
email_verified":false,"gender":"male","profile":"https:\/\/fi.wikipedia.org\/wiki\/Tom_Cruise","
phone_number_verified":true,"preferred_username":"ttester","locale":"en-US","given_name":"Teppo
Matias","middle_name":"Matias","manipe":[1,3],"picture":"https:\/\/pixabay.com\/fi\/pentu-kissa-kukka-
potin-tabby-pentu-2766820\/","updated_at":1509450347,"nickname":"TT","name":"Mr.Teppo Matias
Testaaja","phone_number":"+1 (604) 555-1234;ext=5678","family_name":"Testaaja","email":"teppo@example.
org"}
```

Exercise 4.5

Configuring Subject claim

1. Modify the registration data of the RP to list subject of type pairwise.

```
nano /opt/shibboleth-idp/metadata/oidc-client.json

{
  "scope":"openid info profile email address phone",
  "redirect_uris":["https://192.168.0.150:8443/protected/redirect_uri"],
  "client_id":"test_rp",
  "client_secret":"testSecret1234",
  "response_types":["id_token","code"],
  "grant_types":["authorization_code","implicit","refresh_token"],
  "subject_type":"pairwise"
}
```

2. Authenticate the user. Verify from landing page or from logs that the value of subject is different from previously used public value.

Hints, Tips and Result

```
[OIDC_CLAIM_sub] => DQ3YFEXBF65XMAULUJHBAI34IVRR3GT5
```

3. Activate the attributes "subject-public" and "subject-pairwise" by modifying the comment markers. Ensure the filtering rules are such that these attributes are not filtered out but the attribute "subject" is filtered out for your RP.
4. Modify the registration data of the rp to request subject of type public.

```
nano /opt/shibboleth-idp/metadata/oidc-client.json

{
  "scope":"openid info profile email address phone",
  "redirect_uris":["https://IP_ADDRESS:8443/protected/redirect_uri"],
  "client_id":"test_rp",
  "client_secret":"testSecret1234",
  "response_types":["id_token","code"],
  "grant_types":["authorization_code","implicit","refresh_token"],
  "subject_type":"public"
}
```

5. Authenticate the user. Verify from landing page or from logs that the value of subject is now this plain text value.

Hints, Tips and Result

```
[OIDC_CLAIM_sub] => teppo
```