Scripting

AM provides a scripting engine for you to develop scripts for authentication, as well as for policy conditions, handling OpenID Connect claims, and others.

Get Started

Learn about how to use scripts in AM to modify the default behavior.

Scripting API

Discover which behaviors of AM you can modify with scripts.

Sample Scripts

Learn by example.

ForgeRock® Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

Scripting environment

AM supports scripts written in either JavaScript, or Groovy. Scripts used for client-side authentication must be written in JavaScript.

▼ How to determine the JavaScript Engine Version

You can use a script to check the version of the JavaScript engine AM is using. You could temporarily add the following script to a Scripted Decision node, for example, to output the engine version to the debug log:

```
var rhino = JavaImporter(
  org.mozilla.javascript.Context
)

var currentContext = rhino.Context.getCurrentContext()
var rhinoVersion = currentContext.getImplementationVersion()

logger.error("JS Script Engine: " + rhinoVersion)

outcome = "true"
```

NOTE

Ensure the following are listed in the **Java class whitelist** property of the scripting engine.

- `org.mozilla.javascript.Context`
- `org.forgerock.openam.scripting.timeouts.*`

To view the Java class allowlist, go to **Configure > Global Services > Scripting > Secondary Configurations**. Select the script type, and on the **Secondary Configurations** tab, click **engineConfiguration**.

AM uses the Mozilla Rhino JavaScript engine.

▼ How to determine the Groovy engine version

You can use a script to check the version of the Groovy scripting engine AM is using. You could temporarily add the following script to a Scripted Decision node, for example, to output the engine version to the debug log:

```
logger.error("Groovy Script Engine: " + GroovySystem.version)

outcome = "true"
```

NOTE

NOTE

Ensure the following are listed in the **Java class whitelist** property of the scripting engine.

- `groovy.lang.GroovySystem`

To view the Java class allowlist, go to **Configure > Global Services > Scripting > Secondary Configurations**. Select the script type, and on the **Secondary Configurations** tab, click **engineConfiguration**.

AM uses the Apache Groovy engine.

To access the functionality AM provides, import the required Java class or package, as follows:

JavaScript | Groovy

```
var fr = JavaImporter(
    org.forgerock.openam.auth.node.api,
    javax.security.auth.callback.NameCallback
);
with (fr) {
    ...
}
```

You may need to allowlist the classes you use in scripts. See Security.

You can use scripts to modify default AM behavior in the following situations, also known as *contexts*:

Client-side authentication
: Scripts that are executed on the client during authentication.

Client-side scripts must be in JavaScript.

Server-side authentication
: Scripts are included in an authentication module within a chain and are executed on the server during authentication.

Authentication trees
: Scripts are included in an authentication node within a tree and are executed on the server during authentication.

Policy conditions
: Scripts used as conditions within policies.

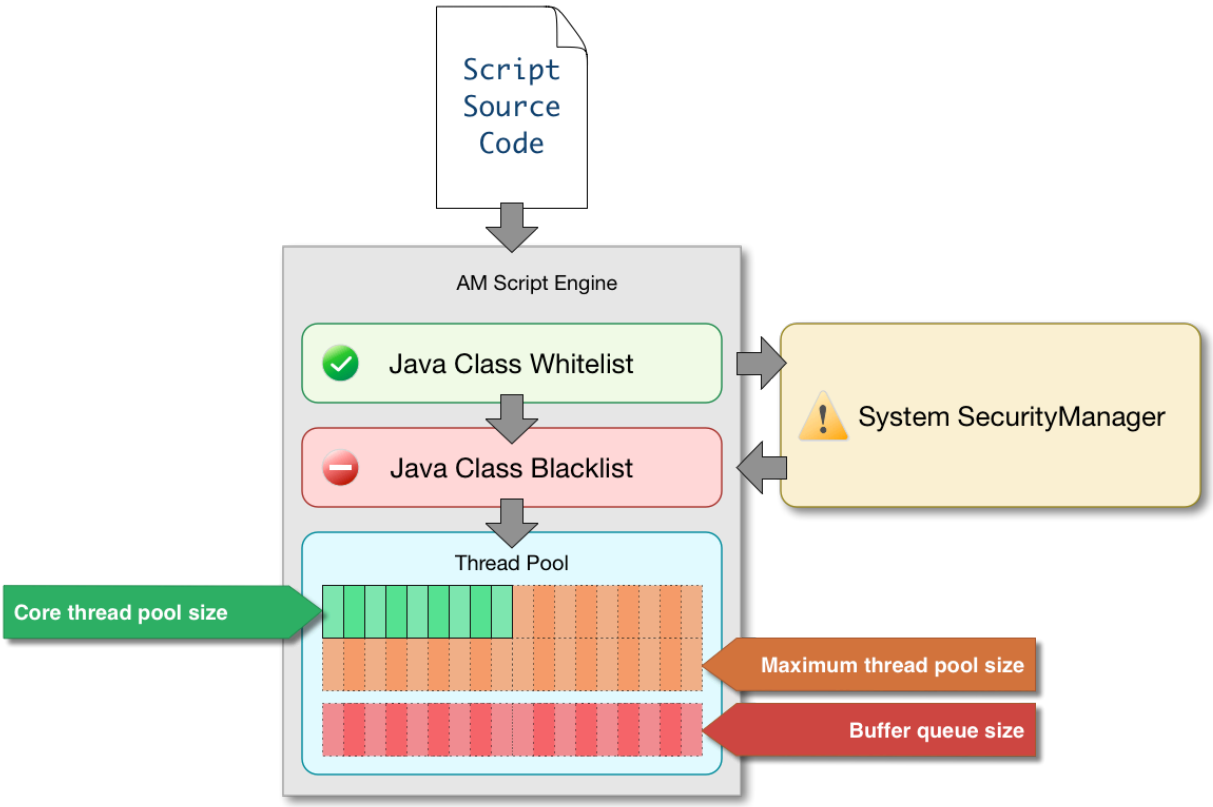
OIDC claims

Scripts that gather and populate the claims in a request when issuing an ID token or making a request to the `userinfo` endpoint.

OAuth 2.0 access tokens
Scripts that modify the key-value pairs contained within access tokens before they are issued to a client.

AM implements a configurable scripting engine for each of the context types that are executed on the server.

The scripting engines in AM have two main components: security settings, and the thread pool.



Security

AM scripting engines provide security features for ensuring that malicious Java classes are not directly called. The engines validate scripts by checking all directly-called Java classes against a configurable denylist and allowlist, and, optionally, against the JVM SecurityManager, if it is configured.

Allowlists and denylists contain class names that are allowed or denied execution respectively. Specify classes in allowlists and denylists by name or by using regular expressions.

Classes called by the script are checked against the allowlist first, and must match at least one pattern in the list. The denylist is applied after the allowlist, and classes

matching any pattern are disallowed.

You can also configure the scripting engine to make an additional call to the JVM security manager for each class that is accessed. The security manager throws an exception if a class being called is not allowed to execute.

For more information on configuring script engine security, see Scripting.

Important Points About Script Engine Security

The following points should be considered when configuring the security settings within each script engine:

The scripting engine only validates directly accessible classes.
The security settings only apply to classes that the script *directly* accesses. If the script calls `Foo.a()` and then that method calls `Bar.b()`, the scripting engine will be unable to prevent it. You must consider the whole chain of accessible classes.

> NOTE
>
> *Access* includes actions such as:
>
> - Importing or loading a class.
> - Accessing any instance of that class. For example, passed as a parameter to the script.
> - Calling a static method on that class.
> - Calling a method on an instance of that class.
> - Accessing a method or field that returns an instance of that class.

Potentially dangerous Java classes are denylisted by default.
All Java reflection classes (`java.lang.Class`, `java.lang.reflect.*`) are denylisted by default to avoid bypassing the security settings.

The `java.security.AccessController` class is also blacklisted by default to prevent access to the `doPrivileged()` methods.

> CAUTION
>
> You should not remove potentially dangerous Java classes from the Denylist.

The allowlists and denylists match class or package names only.
The allowlist and denylist patterns apply only to the exact class or package names involved. The script engine does not know anything about inheritance, so it is best to allowlist known, specific classes.

Thread Pools

Each script is executed in an individual thread. Each scripting engine starts with an initial number of threads available for executing scripts. If no threads are available for execution, AM creates a new thread to execute the script, until the configured maximum number of threads is reached.

If the maximum number of threads is reached, pending script executions are queued in a number of buffer threads, until a thread becomes available for execution. If a created thread has completed script execution and has remained idle for a configured amount of time, AM terminates the thread, shrinking the pool.

For more information on configuring script engine thread pools, see Scripting.

Scripting API

AM provides the following functionality and artifacts for scripting:

Scripted module API
(*Authentication chains only*)

Access authentication state data, user profile data, request data, and information gathered by client-side scripts.

Scripted decision node API
(*Authentication trees only*)

Access data in request headers, shared state, and user session data.

Policy Condition Script API
Access the authorization state data, the information pertaining a session, and the user's profile data in authorization policies.

Customize OAuth 2.0 with plugins
Extend authorization server behavior with the OAuth 2.0 plugins:

- **Access token modification plugin** Modify the key-value pairs contained within an OAuth 2.0 access token.
- **Authorize endpoint data provider plugin** Return additional data from an authorization request.
- **Scope evaluator plugin** Evaluate and return an OAuth2 access token's scope information.
- **Scope validator plugin** Customize the set of requested scopes for authorize, access token, refresh token and back channel authorize requests.

- **User info claims plugin** Map scopes to claims and data for OpenID Connect ID tokens.

Token exchange scripting API
Add the `may_act` claim when performing OAuth 2.0 token exchange.

Accessing HTTP Services
Configure the parameters for the HTTP client object in any server-side script.

Debug logging
Add debug logging to any server-side script.

Configuring AM for token exchange
Add `may_act` claims to OAuth 2.0/OpenID Connect exchanged tokens.

Reference substituted properties in scripts

The `systemEnv` binding, available to all AM script types, includes an instance of the `ScriptPropertyResolver` class. This interface exposes the following methods:

JavaScript | Groovy

```
String getProperty(String propertyName);
String getProperty(String propertyName, String defaultValue);
```

where:

1. `propertyName` is the configuration expression (without the ampersand braces)
2. `defaultValue` is the default value set for that property in the configuration expression

To reference a substituted property in a script, the property name *must* include a specific prefix. This prefix decreases the risk that random property values are resolved in scripts. Because property value substitution is often used to change secrets, you must be intentional about which properties you want to be able to resolve in scripts. The default prefix, for all script types, is `script`. You can change this prefix in the script type configuration. Select **Configure > Global Services > Scripting > Secondary Configurations > *Script Type* > Secondary Configurations > Engine Configuration > Property Name Prefix**.

Example: Property value resolution in a scripted decision node

These examples assume that the property name prefix `script` has been set in the script engine configuration for this script type. The scripts are used in a scripted decision

node to get the values of the user's `email`, `hostname`, `port`, and so on. The scripts also show type transformation where the type is not a string.

JavaScript | Groovy

```
// Properties should get resolved (set in AM)
var email =
systemEnv.getProperty('script.tree.decision.node.email');
var name =
systemEnv.getProperty('script.tree.decision.node.hostname',
'defaultHostname');
var port =
systemEnv.getProperty('script.tree.decision.node.port',
'587', java.lang.Integer);
var double =
systemEnv.getProperty('script.tree.decision.node.double,
'2.0', java.lang.Double);
var hasPort =
systemEnv.getProperty('script.tree.decision.node.hasPort',
'false', java.lang.Boolean);
var map =
systemEnv.getProperty('script.tree.decision.node.map',
'{"defaultKey":"defaultValue"}', java.util.Map);
var list =
systemEnv.getProperty('script.tree.decision.node.list',
'defaultValue', java.util.List);

// Properties should get resolved to their defaults (not set
in AM)
var defaultName =
systemEnv.getProperty('script.tree.decision.node.hostname.unr
esolved', 'defaultHostname');
var defaultPort =
systemEnv.getProperty('script.tree.decision.node.port.unresol
ved', '587', java.lang.Integer);
var defaultDouble =
systemEnv.getProperty('script.tree.decision.node.double.unres
olved', '2.0', java.lang.Double);
var defaultHasPort =
systemEnv.getProperty('script.tree.decision.node.hasPort.unre
solved', 'false', java.lang.Boolean);
var defaultMap =
systemEnv.getProperty('script.tree.decision.node.map.unresolv
ed', '{"defaultKey":"defaultValue"}', java.util.Map);
var defaultList =
```

```
systemEnv.getProperty('script.tree.decision.node.list.unresol
ved', 'defaultFirstValue,defaultSecondValue',
java.util.List);

// Assert all property values - set the appropriate outcome
if (email === 'test@example.com' && name === 'testHostname'
&& port === 25 && double === 1.0 && hasPort === true
    && map.get('testKey') == 'testValue' && list ==
'[testFirstValue, testSecondValue]'
    && defaultName === 'defaultHostname' && defaultPort ===
587 && defaultDouble === 2.0 && defaultHasPort === false
    && defaultMap.get('defaultKey') == 'defaultValue' &&
defaultList == '[defaultFirstValue, defaultSecondValue]') {
outcome = 'true';
} else {
outcome = 'false';
}
```

Reference ESVs in scripts

Use the `esv` prefix to reference ESVs from scripts. This prefix lets you create generic scripts that can be customized in different environments. For example:

```
var host = systemEnv.getProperty(esv.my.host);
```

Accessing HTTP Services

AM passes an HTTP client object, `httpClient`, to server-side scripts. Server-side scripts can call HTTP services with the `httpClient.send` method. The method returns an `HttpClientResponse` object.

Configure the parameters for the HTTP client object by using the `org.forgerock.http.protocol` package. This package contains the `Request` class, which has methods for setting the URI and type of request.

The following example, taken from the default server-side Scripted authentication module script, uses these methods to call an online API to determine the longitude and latitude of a user based on their postal address:

```
function getLongitudeLatitudeFromUserPostalAddress() {

    var request = new org.forgerock.http.protocol.Request();
```

```
request.setUri("http://maps.googleapis.com/maps/api/geocode/json?
address=" + encodeURIComponent(userPostalAddress));
    request.setMethod("GET");

    var response = httpClient.send(request).get();
    logResponse(response);

    var geocode = JSON.parse(response.getEntity());
    var i;

    for (i = 0; i < geocode.results.length; i++) {
        var result = geocode.results[i];
        latitude = result.geometry.location.lat;
        longitude = result.geometry.location.lng;

        logger.message("latitude:" + latitude + " longitude:" +
longitude);
    }
}
```

HTTP client requests are synchronous and blocking until they return. You can, however, set a global timeout for server-side scripts. For details, see Scripted Authentication Module Properties.

Server-side scripts can access response data by using the methods listed in the table below.

HTTP Client Response Methods

Method	Parameters	Return Type	Description
`HttpClientResponse.getCookies`	`Void`	`Map<String, String>`	Get the cookies for the returned response, if any exist.
`HttpClientResponse.getEntity`	`Void`	`String`	Get the entity of the returned response.
`HttpClientResponse.getHeaders`	`Void`	`Map<String, String>`	Get the headers for the returned response, if any exist.

Method	Parameters	Return Type	Description
`HttpClientResponse.getReasonPhrase`	`Void`	`String`	Get the reason phrase of the returned response.
`HttpClientResponse.getStatusCode`	`Void`	`Integer`	Get the status code of the returned response.
`HttpClientResponse.hasCookies`	`Void`	`Boolean`	Indicate whether the returned response had any cookies.
`HttpClientResponse.hasHeaders`	`Void`	`Boolean`	Indicate whether the returned response had any headers.

Debug logging

Server-side scripts can write messages to AM debug logs by using the `logger` object.

NOTE

The scripting API does not use the logback logger that is used by the rest of AM. Instead, it uses an instance of the custom AM `Debug` logger class.

AM does not log debug messages from scripts by default. To configure AM to log script messages, set the debug log level for the `amScript` service.

The following table lists the `logger` methods.

Logger methods

Method	Parameters	Return Type	Description
`logger.error`	*`Error Message`* (type: `String`)	`Void`	Write *Error Message* to AM debug logs if ERROR level logging is enabled.

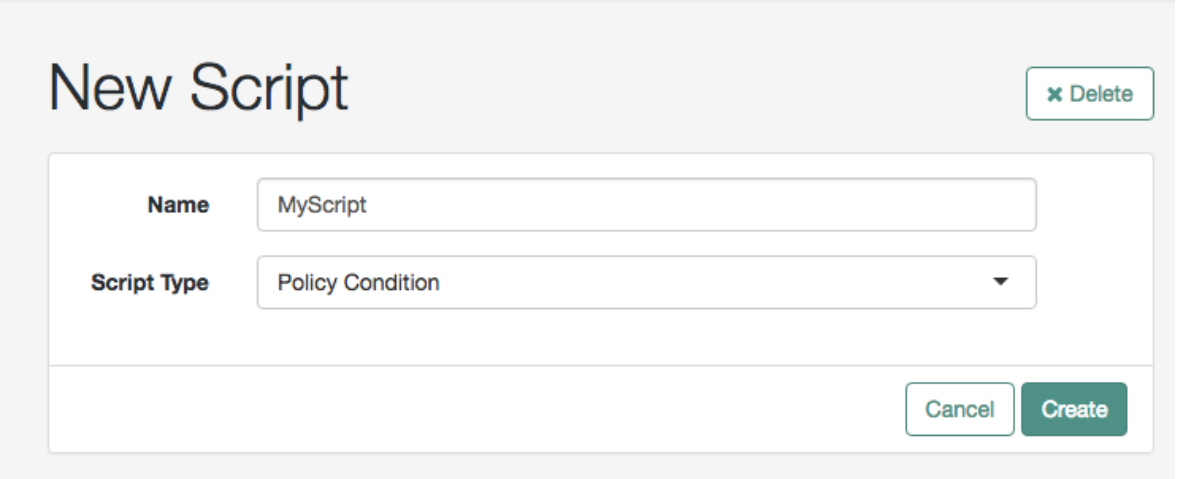
Method	Parameters	Return Type	Description
`logger.errorEnabled`	`Void`	`Boolean`	Return `true` when ERROR level debug messages are enabled.
`logger.message`	*`Message`* (type: `String`)	`Void`	Write *Message* to AM debug logs if MESSAGE level logging is enabled.
`logger.messageEnabled`	`Void`	`Boolean`	Return `true` when MESSAGE level debug messages are enabled.
`logger.warning`	*`Warning Message`* (type: `String`)	`Void`	Write *Warning Message* to AM debug logs if WARNING level logging is enabled.
`logger.warningEnabled`	`Void`	`Boolean`	Return `true` when WARNING level debug messages are enabled.

Manage scripts (UI)

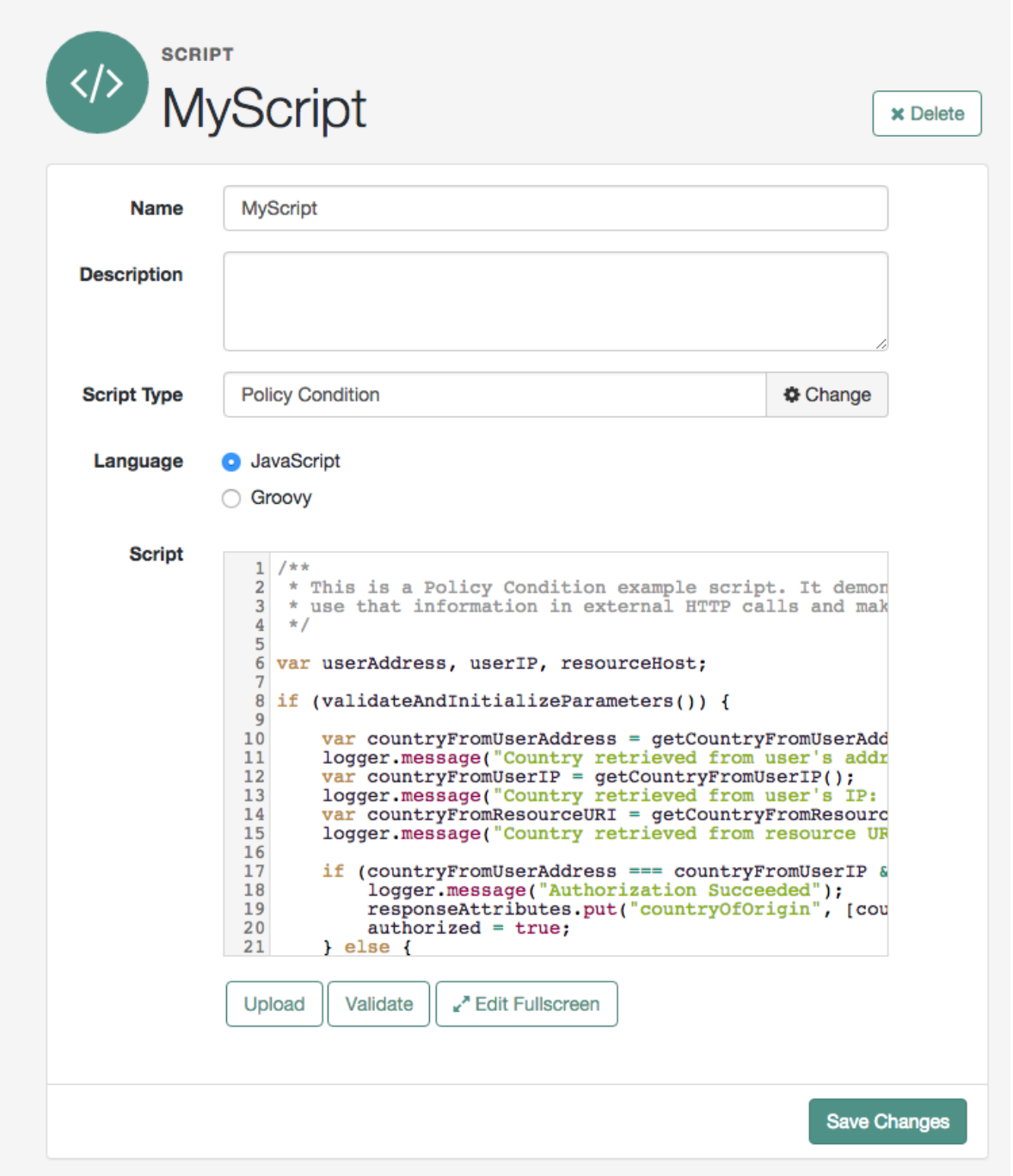
The following procedures describe how to create, modify, and delete scripts using the AM admin UI.

Create a script

1. In the AM admin UI, go to **Realms** > ***Realm Name*** > **Scripts**, and click **New Script**.



2. Specify a name for the script.
3. Select the type of script from the **Script Type** drop-down list.
4. Click **Create**.



5. Enter values on the *Script Name* page as follows:
    - Enter a description of the script.
    - Choose the script language, either JavaScript or Groovy. Note that not every script type supports both languages.
    - Enter the source code in the Script field.

      On supported browsers, you can click Upload, go to the script file, and click Open to upload the contents to the Script field.
    - Click Validate to check for compilation errors in the script.

      Correct any compilation errors, and revalidate the script until all errors have been fixed.
    - Save your changes.

Modify a script

1. In the AM admin UI, go to Realms > *Realm Name* > Scripts.
2. Select the script you want to modify from the list of scripts.
3. Modify values on the *Script Name* page as needed.

   Note that if you change the Script Type, existing code in the script is replaced.
4. If you modified the code in the script, click Validate to check for compilation errors.

   Correct any compilation errors, and revalidate the script until all errors have been fixed.
5. Save your changes.

Delete a script

1. In the AM admin UI, go to Realms > *Realm Name* > Scripts.
2. Choose one or more scripts to delete by activating the checkboxes in the relevant rows.

   Note that you can only delete user-created scripts—you cannot delete the global sample scripts provided with AM.
3. Click Delete.

Manage scripts (REST)

AM provides the `scripts` endpoint to manage scripts using REST calls.

The following actions are available:

User-created scripts are realm-specific, hence the URI for the scripts' API can contain a realm component, such as `/json{/realm}/scripts`. If the realm is not specified in the URI, the top level realm is used.

TIP

AM includes some global example scripts that can be used in any realm.

Scripts are represented in JSON and take the following form. Scripts are built from standard JSON objects and values (strings, numbers, objects, sets, arrays, `true`, `false`, and `null`). Each script has a system-generated *universally unique identifier* (UUID), which must be used when modifying existing scripts. Renaming a script will not affect the UUID:

```
{
  "_id": "7e3d7067-d50f-4674-8c76-a3e13a810c33",
  "name": "Scripted Module - Server Side",
  "description": "Default global script for server side Scripted
Authentication Module",
  "script": "dmFyIFNUQVJUX1RJ...",
  "language": "JAVASCRIPT",
  "context": "AUTHENTICATION_SERVER_SIDE",
  "createdBy":
"id=dsameuser,ou=user,dc=openam,dc=forgerock,dc=org",
  "creationDate": 1433147666269,
  "lastModifiedBy":
"id=dsameuser,ou=user,dc=openam,dc=forgerock,dc=org",
  "lastModifiedDate": 1433147666269
}
```

The values for the fields shown in the example are explained below:

_id
: The UUID that AM generates for the script.

name
: The name provided for the script.

description
: An optional text string to help identify the script.

script
: The source code of the script. The source code is in UTF-8 format and encoded into Base64.

  For example, a script such as the following:

```
var a = 123;
var b = 456;
```

  When encoded into Base64 becomes:

```
dmFyIGEgPSAxMjM7IA0KdmFyIGIgPSA0NTY7
```

language
: The language the script is written in - `JAVASCRIPT` or `GROOVY`.

  Language Support per Context

Script Context	Supported Languages
`POLICY_CONDITION`	`JAVASCRIPT, GROOVY`
`AUTHENTICATION_SERVER_SIDE`	`JAVASCRIPT, GROOVY`
`AUTHENTICATION_CLIENT_SIDE`	`JAVASCRIPT`
`OIDC_CLAIMS`	`JAVASCRIPT, GROOVY`
`AUTHENTICATION_TREE_DECISION_NODE`	`JAVASCRIPT, GROOVY`

`context`
: The context type of the script.

  Supported values are:

  `POLICY_CONDITION`
  : Policy Condition

  `AUTHENTICATION_SERVER_SIDE`
  : Server-side Authentication

  `AUTHENTICATION_CLIENT_SIDE`
  : Client-side Authentication

    NOTE

    Client-side scripts must be written in JavaScript.

  `OIDC_CLAIMS`
  : OIDC Claims

  `AUTHENTICATION_TREE_DECISION_NODE`
  : Authentication scripts used by Scripted Tree Decision authentication nodes.

`createdBy`
: A string containing the universal identifier DN of the subject that created the script.

`creationDate`
: An integer containing the creation date and time, in ISO 8601 format.

`lastModifiedBy`
: A string containing the universal identifier DN of the subject that most recently updated the resource type.

  If the script has not been modified since it was created, this property will have the same value as `createdBy`.

lastModifiedDate

A string containing the last modified date and time, in ISO 8601 format.

If the script has not been modified since it was created, this property will have the same value as `creationDate`.

Querying Scripts

To list all the scripts in a realm, as well as any global scripts, perform an HTTP GET to the `/json{/realm}/scripts` endpoint with a `_queryFilter` parameter set to `true`.

NOTE

If the realm is not specified in the URL, AM returns scripts in the top level realm, as well as any global scripts.

The `iPlanetDirectoryPro` header is required and should contain the SSO token of an administrative user, such as `amAdmin`, who has access to perform the operation.

```
$ curl \
--header "iPlanetDirectoryPro: AQIC5…" \
--header "Accept-API-Version: resource=1.1" \
https://openam.example.com:8443/openam/json/realms/root/realms/myrealm/scripts?_queryFilter=true
{
    "result": [
        {
            "_id": "9de3eb62-f131-4fac-a294-7bd170fd4acb",
            "name": "Scripted Policy Condition",
            "description": "Default global script for Scripted Policy Conditions",
            "script": "LyoqCiAqIFRoaXMg…",
            "language": "JAVASCRIPT",
            "context": "POLICY_CONDITION",
            "createdBy": "id=dsameuser,ou=user,dc=openam,dc=forgerock,dc=org",
            "creationDate": 1433147666269,
            "lastModifiedBy": "id=dsameuser,ou=user,dc=openam,dc=forgerock,dc=org",
            "lastModifiedDate": 1433147666269
        },
        {
            "_id": "7e3d7067-d50f-4674-8c76-a3e13a810c33",
            "name": "Scripted Module - Server Side",
            "description": "Default global script for server side
```

```
Scripted Authentication Module",
            "script": "dmFyIFNUQVJUX1RJ…",
            "language": "JAVASCRIPT",
            "context": "AUTHENTICATION_SERVER_SIDE",
            "createdBy":
"id=dsameuser,ou=user,dc=openam,dc=forgerock,dc=org",
            "creationDate": 1433147666269,
            "lastModifiedBy":
"id=dsameuser,ou=user,dc=openam,dc=forgerock,dc=org",
            "lastModifiedDate": 1433147666269
        }
    ],
    "resultCount": 2,
    "pagedResultsCookie": null,
    "remainingPagedResults": -1
}
```

Supported _queryFilter Fields andOperators

Field	Supported Operators
`_id`	Equals (eq), Contains (co), Starts with (sw)
`name`	Equals (eq), Contains (co), Starts with (sw)
`description`	Equals (eq), Contains (co), Starts with (sw)
`script`	Equals (eq), Contains (co), Starts with (sw)
`language`	Equals (eq), Contains (co), Starts with (sw)
`context`	Equals (eq), Contains (co), Starts with (sw)

Reading a Script

To read an individual script in a realm, perform an HTTP GET using the `/json{/realm}/scripts` endpoint, specifying the UUID in the URL.

TIP

To read a script in the Top Level Realm, or to read a built-in global script, do not specify a realm in the URL.

The `iPlanetDirectoryPro` header is required and should contain the SSO token of an administrative user, such as `amAdmin`, who has access to perform the operation.

```
$ curl \
--header "iPlanetDirectoryPro: AQIC5…" \
--header "Accept-API-Version: resource=1.1" \
https://openam.example.com:8443/openam/json/realms/root/realms/myrealm/scripts/9de3eb62-f131-4fac-a294-7bd170fd4acb
{
    "_id": "9de3eb62-f131-4fac-a294-7bd170fd4acb",
    "name": "Scripted Policy Condition",
    "description": "Default global script for Scripted Policy Conditions",
    "script": "LyoqCiAqIFRoaXMg…",
    "language": "JAVASCRIPT",
    "context": "POLICY_CONDITION",
    "createdBy": "id=dsameuser,ou=user,dc=openam,dc=forgerock,dc=org",
    "creationDate": 1433147666269,
    "lastModifiedBy": "id=dsameuser,ou=user,dc=openam,dc=forgerock,dc=org",
    "lastModifiedDate": 1433147666269
}
```

Validating a Script

To validate a script, perform an HTTP POST using the `/json{/realm}/scripts` endpoint, with an `_action` parameter set to `validate`. Include a JSON representation of the script and the script language, `JAVASCRIPT` or `GROOVY`, in the POST data.

The value for `script` must be in UTF-8 format and then encoded into Base64.

The `iPlanetDirectoryPro` header is required and should contain the SSO token of an administrative user, such as `amAdmin`, who has access to perform the operation.

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "iPlanetDirectoryPro: AQIC5…" \
```

```
--header "Accept-API-Version: resource=1.1" \
--data '{
    "script": "dmFyIGEgPSAxMjM7dmFyIGIgPSA0NTY7Cg==",
    "language": "JAVASCRIPT"
}' \
https://openam.example.com:8443/openam/json/realms/root/realms/myrealm/scripts/?_action=validate
{
    "success": true
}
```

If the script is valid the JSON response contains a `success` key with a value of `true`.

If the script is invalid the JSON response contains a `success` key with a value of `false`, and an indication of the problem and where it occurs, as shown below:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "iPlanetDirectoryPro: AQIC5…" \
--header "Accept-API-Version: resource=1.1" \
--data '{
    "script":
"dmFyIGEgPSAxMjM7dmFyIGIgPSA0NTY7ID1WQUxJREFUSU9OIFNIT1VMRCBGQUlMPQo=",
    "language": "JAVASCRIPT"
}' \
https://openam.example.com:8443/openam/json/realms/root/realms/myrealm/scripts/?_action=validate
{
    "success": false,
    "errors": [
        {
            "line": 1,
            "column": 27,
            "message": "syntax error"
        }
    ]
}
```

Creating a Script

To create a script in a realm, perform an HTTP POST using the `/json{/realm}/scripts` endpoint, with an `_action` parameter set to `create`. Include a JSON representation of the script in the POST data.

The value for `script` must be in UTF-8 format and then encoded into Base64.

NOTE

If the realm is not specified in the URL, AM creates the script in the top level realm.

The `iPlanetDirectoryPro` header is required and should contain the SSO token of an administrative user, such as `amAdmin`, who has access to perform the operation.

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "iPlanetDirectoryPro: AQIC5…" \
--header "Accept-API-Version: resource=1.1" \
--data '{
    "name": "MyJavaScript",
    "script": "dmFyIGEgPSAxMjM7CnZhciBiID0gNDU2Ow==",
    "language": "JAVASCRIPT",
    "context": "POLICY_CONDITION",
    "description": "An example script"
}' \
https://openam.example.com:8443/openam/json/realms/root/realms/myrealm/scripts/?_action=create
{
    "_id": "0168d494-015a-420f-ae5a-6a2a5c1126af",
    "name": "MyJavaScript",
    "description": "An example script",
    "script": "dmFyIGEgPSAxMjM7CnZhciBiID0gNDU2Ow==",
    "language": "JAVASCRIPT",
    "context": "POLICY_CONDITION",
    "createdBy":
"id=amadmin,ou=user,dc=openam,dc=forgerock,dc=org",
    "creationDate": 1436807766258,
    "lastModifiedBy":
"id=amadmin,ou=user,dc=openam,dc=forgerock,dc=org",
    "lastModifiedDate": 1436807766258
}
```

Updating a Script

To update an individual script in a realm, perform an HTTP PUT using the `/json{/realm}/scripts` endpoint, specifying the UUID in both the URL and the PUT body. Include a JSON representation of the updated script in the PUT data, alongside the UUID.

NOTE

If the realm is not specified in the URL, AM uses the top level realm.

The `iPlanetDirectoryPro` header is required and should contain the SSO token of an administrative user, such as `amAdmin`, who has access to perform the operation.

```
$ curl \
--header "iPlanetDirectoryPro: AQIC5…" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.1" \
--request PUT \
--data '{
    "name": "MyUpdatedJavaScript",
    "script": "dmFyIGEgPSAxMjM7CnZhciBiID0gNDU2Ow==",
    "language": "JAVASCRIPT",
    "context": "POLICY_CONDITION",
    "description": "An updated example script configuration"
}' \
https://openam.example.com:8443/openam/json/realms/root/realms/myrealm/scripts/0168d494-015a-420f-ae5a-6a2a5c1126af
{
    "_id": "0168d494-015a-420f-ae5a-6a2a5c1126af",
    "name": "MyUpdatedJavaScript",
    "description": "An updated example script configuration",
    "script": "dmFyIGEgPSAxMjM7CnZhciBiID0gNDU2Ow==",
    "language": "JAVASCRIPT",
    "context": "POLICY_CONDITION",
    "createdBy":
"id=amadmin,ou=user,dc=openam,dc=forgerock,dc=org",
    "creationDate": 1436807766258,
    "lastModifiedBy":
"id=amadmin,ou=user,dc=openam,dc=forgerock,dc=org",
    "lastModifiedDate": 1436808364681
}
```

Deleting a Script

To delete an individual script in a realm, perform an HTTP DELETE using the `/json{/realm}/scripts` endpoint, specifying the UUID in the URL.

NOTE

If the realm is not specified in the URL, AM uses the top level realm.

The `iPlanetDirectoryPro` header is required and should contain the SSO token of an administrative user, such as `amAdmin`, who has access to perform the operation.

```
$ curl \
--request DELETE \
--header "iPlanetDirectoryPro: AQIC5…" \
--header "Accept-API-Version: resource=1.1" \
https://openam.example.com:8443/openam/json/realms/root/realms/myrealm/scripts/0168d494-015a-420f-ae5a-6a2a5c1126af
{}
```

Manage scripts (ssoadm)

Use the **`ssoadm`** command's **`create-sub-cfg`**, **`get-sub-cfg`**, and **`delete-sub-cfg`** subcommands to manage AM scripts.

Create an AM script as follows:

1. Create a script configuration file, for example, `/path/to/myScriptConfigurationFile.txt`, containing the following:

```
script-file=/path/to/myScriptFile.js
language=JAVASCRIPT ①
name=My New Script
context=AUTHENTICATION_SERVER_SIDE ②
```

① Possible values for the `language` property are:

- `JAVASCRIPT`
- `GROOVY`

② Possible values for the `context` property are:

- `POLICY_CONDITION`
- `AUTHENTICATION_SERVER_SIDE`
- `AUTHENTICATION_CLIENT_SIDE`
- `OIDC_CLAIMS`
- `AUTHENTICATION_TREE_DECISION_NODE`

2. Run the **`ssoadm create-sub-cfg`** command.

   The `--datafile` argument references the script configuration file you created in the previous step:

```
$ ssoadm \
create-sub-cfg \
--realm /myRealm \
--adminid uid=amAdmin,ou=People,dc=openam,dc=forgerock,dc=org
\
--password-file /tmp/pwd.txt \
--servicename ScriptingService \
--subconfigname scriptConfigurations/scriptConfiguration \
--subconfigid myScriptID \
--datafile /path/to/myScriptConfigurationFile.txt
Sub Configuration scriptConfigurations/scriptConfiguration was
added to realm /myRealm
```

   To list the properties of a script, run the **`ssoadm get-sub-cfg`** command:

```
$ ssoadm \
get-sub-cfg \
--realm /myRealm \
--adminid uid=amAdmin,ou=People,dc=openam,dc=forgerock,dc=org
\
--password-file /tmp/pwd.txt \
--servicename ScriptingService \
--subconfigname scriptConfigurations/myScriptID
createdBy=
lastModifiedDate=
lastModifiedBy=
name=My New Script
context=AUTHENTICATION_SERVER_SIDE
description=
language=JAVASCRIPT
```

```
creationDate=
script=…Script output follows…
```

To delete a script, run the **`ssoadm delete-sub-cfg`** command:

```
$ ssoadm \
delete-sub-cfg \
--realm /myRealm \
--adminid uid=amAdmin,ou=People,dc=openam,dc=forgerock,dc=org
\
--password-file /tmp/pwd.txt \
--servicename ScriptingService \
--subconfigname scriptConfigurations/myScriptID
Sub Configuration scriptConfigurations/myScriptID was deleted
from realm /myRealm
```

Sample scripts

The following sample scripts demonstrate how to extend AM.

Groovy Samples

- amazon-profile-normalization.groovy
- apple-profile-normalization.groovy
- facebook-profile-normalization.groovy
- google-profile-normalization.groovy
- instagram-profile-normalization.groovy
- itsme-profile-normalization.groovy
- linkedIn-profile-normalization.groovy
- microsoft-profile-normalization.groovy
- normalized-profile-to-identity.groovy
- normalized-profile-to-managed-user.groovy
- oauth2-access-token-modification.groovy
- oauth2-may-act.groovy
- oidc-claims-extension.groovy
- salesforce-profile-normalization.groovy
- social-idp-profile-transformation.groovy

- twitter-profile-normalization.groovy
- vkontakte-profile-normalization.groovy
- wechat-profile-normalization.groovy
- wordpress-profile-normalization.groovy
- yahoo-profile-normalization.groovy

amazon-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
        field("id", rawProfile.user_id),
        field("displayName", rawProfile.name),
        field("email", rawProfile.email),
        field("username", rawProfile.email)))
```

Open amazon-profile-normalization.groovy in your browser.

apple-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2021-2022 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
```

```
 * to such license between the licensee and ForgeRock AS.
 *
 * In some common default configurations, the following keys are
required to be not empty:
 * username, givenName, familyName, email.
 *
 * From RFC4517: A value of the Directory String syntax is a
string of one or more
 * arbitrary characters from the Universal Character Set (UCS).
 * A zero-length character string is not permitted.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

String email = "change@me.com"
String subjectId = rawProfile.sub
String firstName = " "
String lastName = " "
String username = subjectId
String name

if (rawProfile.isDefined("email") &&
rawProfile.email.isNotNull()){ // User can elect to not share
their email
    email = rawProfile.email.asString()
    username = email
}
if (rawProfile.isDefined("name") && rawProfile.name.isNotNull())
{
    if (rawProfile.name.isDefined("firstName") &&
rawProfile.name.firstName.isNotNull()) {
        firstName = rawProfile.name.firstName.asString()
    }
    if (rawProfile.name.isDefined("lastName") &&
rawProfile.name.lastName.isNotNull()) {
        lastName = rawProfile.name.lastName.asString()
    }
}

name = (firstName?.trim() ? firstName : "") + (lastName?.trim()
? ((firstName?.trim() ? " " : "") + lastName) : "")
name =  (!name?.trim()) ? " " : name
```

```
return json(object(
        field("id", subjectId),
        field("displayName", name),
        field("email", email),
        field("givenName", firstName),
        field("familyName", lastName),
        field("username", username)))
```

Open apple-profile-normalization.groovy in your browser.

facebook-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
        field("id", rawProfile.id),
        field("displayName", rawProfile.name),
        field("givenName", rawProfile.first_name),
        field("familyName", rawProfile.last_name),
        field("photoUrl", rawProfile.picture.data.url),
        field("email", rawProfile.email),
        field("username", rawProfile.email)))
```

Open facebook-profile-normalization.groovy in your browser.

google-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
```

```
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
        field("id", rawProfile.sub),
        field("displayName", rawProfile.name),
        field("givenName", rawProfile.given_name),
        field("familyName", rawProfile.family_name),
        field("photoUrl", rawProfile.picture),
        field("email", rawProfile.email),
        field("username", rawProfile.email),
        field("locale", rawProfile.locale)))
```

Open google-profile-normalization.groovy in your browser.

instagram-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
```

```
        field("id", rawProfile.id),
        field("username", rawProfile.username)))
```

Open instagram-profile-normalization.groovy in your browser.

itsme-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020-2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

import org.forgerock.json.JsonValue

JsonValue managedUser = json(object(
        field("id", rawProfile.sub),
        field("displayName", rawProfile.name),
        field("givenName", rawProfile.given_name),
        field("familyName", rawProfile.family_name),
        field("username", rawProfile.email),
        field("email", rawProfile.email)))
return managedUser
```

Open itsme-profile-normalization.groovy in your browser.

linkedIn-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
```

```
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
        field("id", rawProfile.id),
        field("givenName",
rawProfile.firstName.localized.get(0)),
        field("familyName",
rawProfile.lastName.localized.get(0)),
        field("photoUrl",
rawProfile.profilePicture.displayImage),
        field("email",
rawProfile.elements.get(0).get("handle~").emailAddress),
        field("username",
rawProfile.elements.get(0).get("handle~").emailAddress)))
```

Open linkedIn-profile-normalization.groovy in your browser.

microsoft-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
        field("id", rawProfile.id),
        field("displayName", rawProfile.displayName),
```

```
        field("givenName", rawProfile.givenName),
        field("familyName", rawProfile.surname),
        field("email", rawProfile.userPrincipalName),
        field("username", rawProfile.userPrincipalName)))
```

Open microsoft-profile-normalization.groovy in your browser.

normalized-profile-to-identity.groovy

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

import org.forgerock.json.JsonValue

JsonValue identity = json(object(
        field("givenName", normalizedProfile.givenName),
        field("sn", normalizedProfile.familyName),
        field("mail", normalizedProfile.email),
        field("cn", normalizedProfile.displayName),
        field("userName", normalizedProfile.username),
        field("iplanet-am-user-alias-list", selectedIdp + '-' +
normalizedProfile.id.asString())))

return identity
```

Open normalized-profile-to-identity.groovy in your browser.

normalized-profile-to-managed-user.groovy

▼ View script

```
/*
 * Copyright 2020-2022 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

import org.forgerock.json.JsonValue

JsonValue managedUser = json(object(
        field("givenName", normalizedProfile.givenName),
        field("sn", normalizedProfile.familyName),
        field("mail", normalizedProfile.email),
        field("userName", normalizedProfile.username)))

if (normalizedProfile.postalAddress.isNotNull())
managedUser.put("postalAddress",
normalizedProfile.postalAddress)
if (normalizedProfile.addressLocality.isNotNull())
managedUser.put("city", normalizedProfile.addressLocality)
if (normalizedProfile.addressRegion.isNotNull())
managedUser.put("stateProvince",
normalizedProfile.addressRegion)
if (normalizedProfile.postalCode.isNotNull())
managedUser.put("postalCode", normalizedProfile.postalCode)
if (normalizedProfile.country.isNotNull())
managedUser.put("country", normalizedProfile.country)
if (normalizedProfile.phone.isNotNull())
managedUser.put("telephoneNumber", normalizedProfile.phone)

// if the givenName and familyName is null or empty
// then add a boolean flag to the shared state to indicate names
are not present
// this could be used elsewhere
// for eg. this could be used in a scripted decision node to by-
pass patching
// the user object with blank values when givenName  and
familyName is not present
```

```
boolean noGivenName = normalizedProfile.givenName.isNull() ||
(!normalizedProfile.givenName.asString()?.trim())
boolean noFamilyName = normalizedProfile.familyName.isNull() ||
(!normalizedProfile.familyName.asString()?.trim())
sharedState.put("nameEmptyOrNull", noGivenName && noFamilyName)


return managedUser
```

Open normalized-profile-to-managed-user.groovy in your browser.

oauth2-access-token-modification.groovy

▼ View script

```
/*
 * Copyright 2019-2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import org.forgerock.http.protocol.Request
import org.forgerock.http.protocol.Response

import com.iplanet.sso.SSOException

import groovy.json.JsonSlurper

/**
 * Defined variables:
 * accessToken - The access token to be updated. Mutable object,
all changes to the access token will be reflected.
 * httpClient - always present, the HTTP client that can be used
to make external HTTP requests
 * identity - always present, the identity of the resource owner
 * logger - always present, corresponding log files will be
prefixed with: scripts.OAUTH2_ACCESS_TOKEN_MODIFICATION.
 * scopes - always present, the requested scopes
 * session - present if the request contains the session cookie,
the user's session object
 * scriptName - always present, the display name of the script
```

```
 * requestProperties - always present, contains a map of request
properties:
 *                     requestUri - the request URI
 *                     realm - the realm that the request
relates to
 *                     requestParams - a map of the request
params and/or posted data. Each value is a list of one or
 *                     more properties. Please note that these
should be handled in accordance with OWASP best
 *                     practices.
 * clientProperties - present if the client specified in the
request was identified, contains a map of client
 *                    properties:
 *                    clientId - the client's Uri for the
request locale
 *                    allowedGrantTypes - list of the allowed
grant types (org.forgerock.oauth2.core.GrantType)
 *                                         for the client
 *                    allowedResponseTypes - list of the allowed
response types for the client
 *                    allowedScopes - list of the allowed scopes
for the client
 *                    customProperties - A map of the custom
properties of the client.
 *                                       Lists or maps will be
included as sub-maps, e.g:
 *                                       testMap[Key1]=Value1
will be returned as testmap -> Key1 -> Value1
 *
 * No return value - changes shall be made to the accessToken
parameter directly.
 *
 * The changes made to OAuth2 access tokens will directly impact
the size of the CTS tokens, and similarly the size of
 * the JWTs if client based OAuth2 tokens are utilised.
 * When adding/updating fields make sure that the token size
remains within client/user-agent limits.
 */

/*
//Field to always include in token
accessToken.setField("hello", "world")

//Obtain additional values by performing a REST call to an
external service
```

```
try {
    Response response = httpClient.send(new Request()
            .setUri("https://third.party.app/hello.jsp")
            .setMethod("POST")
            .modifyHeaders({ headers -> headers.put("Content-
Type", "application/json;charset=UTF-8") })
//          .setEntity('foo=bar&hello=world'))
            .setEntity([foo: 'bar']))
            .getOrThrow()
    if (response.status.successful) {
        def result = new
JsonSlurper().parseText(response.entity.string)
        accessToken.setFields(result.get("updatedFields"))
    } else {
        logger.error("Unable to obtain access token
modifications: {}, {}", response.status,
response.entity.toString())
    }
} catch (InterruptedException ex) {
    logger.error("The request processing was interrupted", ex)
    Thread.currentThread().interrupt()
    //The access token request will fail with HTTP 500 error in
this case.
    throw new RuntimeException("Unable to obtain response from
")
}

//Add new fields containing identity attribute values
def attributes = identity.getAttributes(["mail",
"telephoneNumber"].toSet())
accessToken.setField("mail", attributes["mail"])
accessToken.setField("phone", attributes["telephoneNumber"])

//Add new fields containing session property values
if (session != null) { // session is not available for resource
owner password credentials grant
    try {
        accessToken.setField("ipAddress",
session.getProperty("Host"))
    } catch (SSOException ex) {
        logger.error("Unable to retrieve session property
value", ex)
    }
}
```

```
// Remove a native field from the token entry, that was set by
AM. For complete list of remove* methods see the JavaDoc
// for org.forgerock.oauth2.core.AccessToken class.
accessToken.removeTokenName()
*/
```

Open oauth2-access-token-modification.groovy in your browser.

oauth2-may-act.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/**
 * Defined variables:
 * token - The access token to be updated. Mutable object, all
changes to the access token will be reflected.
 * logger - always present, corresponding log files will be
prefixed with: scripts.OAUTH2_ACCESS_TOKEN_MODIFICATION.
 * scriptName - always present, the display name of the script
 * session - present if the request contains the session cookie,
the user's session object
 * requestProperties - always present, contains a map of request
properties:
 *                     requestUri - the request URI
 *                     realm - the realm that the request
relates to
 *                     requestParams - a map of the request
params and/or posted data. Each value is a list of one or
 *                     more properties. Please note that these
should be handled in accordance with OWASP best
 *                     practices.
 * clientProperties - present if the client specified in the
request was identified, contains a map of client
 *                    properties:
 *                     clientId - the client's Uri for the
```

```
request locale
 *                    allowedGrantTypes - list of the allowed
grant types (org.forgerock.oauth2.core.GrantType)
 *                                         for the client
 *                    allowedResponseTypes - list of the allowed
response types for the client
 *                    allowedScopes - list of the allowed scopes
for the client
 *                    customProperties - A map of the custom
properties of the client.
 *                                       Lists or maps will be
included as sub-maps, e.g:
 *                                       testMap[Key1]=Value1
will be returned as testmap -> Key1 -> Value1
 *
 * identity - always present, the identity of the resource owner
 * scopes - always present, the requested scopes
 */
/*
import org.forgerock.json.JsonValue

token.setMayAct(
    JsonValue.json(JsonValue.object(
        JsonValue.field("client_id", "myClient"),
        JsonValue.field("sub", "(usr!myActor)"))))
*/
```

Open oauth2-may-act.groovy in your browser.

oidc-claims-extension.groovy

▼ View script

```
/*
 * Copyright 2014-2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */
import com.iplanet.sso.SSOException
import com.sun.identity.idm.IdRepoException
import
```

```
org.forgerock.oauth2.core.exceptions.InvalidRequestException
import org.forgerock.oauth2.core.UserInfoClaims
import org.forgerock.openidconnect.Claim

/*
* Defined variables:
* logger - always presents, the "OAuth2Provider" debug logger
instance
* claims - always present, default server provided claims -
Map<String, Object>
* claimObjects - always present, default server provided claims
- List<Claim>
* session - present if the request contains the session cookie,
the user's session object
* identity - always present, the identity of the resource owner
* scopes - always present, the requested scopes
* scriptName - always present, the display name of the script
* requestProperties - always present, contains a map of request
properties:
*                     requestUri - the request URI
*                     realm - the realm that the request relates
to
*                     requestParams - a map of the request
params and/or posted data. Each value is a list of one or
*                     more properties. Please note that these
should be handled in accordance with OWASP best practices.
* clientProperties - present if the client specified in the
request was identified, contains a map of client
*                   properties:
*                   clientId - the client's Uri for the request
locale
*                   allowedGrantTypes - list of the allowed
grant types (org.forgerock.oauth2.core.GrantType)
*                                       for the client
*                   allowedResponseTypes - list of the allowed
response types for the client
*                   allowedScopes - list of the allowed scopes
for the client
*                   customProperties - A map of the custom
properties of the client.
*                                     Lists or maps will be
included as sub-maps, e.g:
*                                     testMap[Key1]=Value1
will be returned as testmap -> Key1 -> Value1
* requestedClaims - Map<String, Set<String>>
```

```
*                   always present, not empty if the request
contains a claims parameter and server has enabled
*                   claims_parameter_supported, map of requested
claims to possible values, otherwise empty,
*                   requested claims with no requested values
will have a key but no value in the map. A key with
*                   a single value in its Set indicates this is
the only value that should be returned.
* requestedTypedClaims - List<Claim>
*                       always present, not empty if the request
contains a claims parameter and server has enabled
*                       claims_parameter_supported, list of
requested claims with claim name, requested possible values
*                       and if claim is essential, otherwise
empty,
*                       requested claims with no requested
values will have a claim with no values. A claims with
*                       a single value indicates this is the
only value that should be returned.
* claimsLocales - the values from the 'claims_locales' parameter
- List<String>
* Required to return a Map of claims to be added to the id_token
claims
*
* Expected return value structure:
* UserInfoClaims {
*    Map<String, Object> values; // The values of the claims for
the user information
*    Map<String, List<String>> compositeScopes; // Mapping of
scope name to a list of claim names.
* }
*/

// user session not guaranteed to be present
boolean sessionPresent = session != null

/*
 * Pulls first value from users profile attribute
 *
 * @param claim The claim object.
 * @param attr The profile attribute name.
 */
def fromSet = { claim, attr ->
    if (attr != null && attr.size() == 1){
        attr.iterator().next()
```

```
    } else if (attr != null && attr.size() > 1){
        attr
    } else if (logger.warningEnabled()) {
        logger.warning("OpenAMScopeValidator.getUserInfo(): Got
an empty result for claim=$claim");
    }
}

// ---vvvvvvvvvv--- EXAMPLE CLAIM ATTRIBUTE RESOLVER FUNCTIONS -
--vvvvvvvvvv---
/*
 * Claim resolver which resolves the value of the claim from its
requested values.
 *
 * This resolver will return a value if the claim has one
requested values, otherwise an exception is thrown.
 */
defaultClaimResolver = { claim ->
    if (claim.getValues().size() == 1) {
        [(claim.getName()): claim.getValues().iterator().next()]
    } else {
        [:]
    }
}

/*
 * Claim resolver which resolves the value of the claim by
looking up the user's profile.
 *
 * This resolver will return a value for the claim if:
 * # the user's profile attribute is not null
 * # AND the claim contains no requested values
 * # OR the claim contains requested values and the value from
the user's profile is in the list of values
 *
 * If no match is found an exception is thrown.
 */
userProfileClaimResolver = { attribute, claim, identity ->
    if (identity != null) {
        userProfileValue = fromSet(claim.getName(),
identity.getAttribute(attribute))
        if (userProfileValue != null && (claim.getValues() ==
null || claim.getValues().isEmpty() ||
claim.getValues().contains(userProfileValue))) {
            return [(claim.getName()): userProfileValue]
```

```
        }
    }
    [:]
}

/*
 * Claim resolver which resolves the value of the claim of the
user's address.
 *
 * This resolver will return a value for the claim if:
 * # the value of the address is not null
 *
 */
userAddressClaimResolver = { claim, identity ->
    if (identity != null) {
        addressFormattedValue = fromSet(claim.getName(),
identity.getAttribute("postaladdress"))
        if (addressFormattedValue != null) {
            return [
                    "formatted" : addressFormattedValue
            ]
        }
    }
    [:]
}

/*
 * Claim resolver which resolves the value of the claim by
looking up the user's profile.
 *
 * This resolver will return a value for the claim if:
 * # the user's profile attribute is not null
 * # AND the claim contains no requested values
 * # OR the claim contains requested values and the value from
the user's profile is in the list of values
 *
 * If the claim is essential and no value is found an
InvalidRequestException will be thrown and returned to the user.
 * If no match is found an exception is thrown.
 */
essentialClaimResolver = { attribute, claim, identity ->
    if (identity != null) {
        userProfileValue = fromSet(claim.getName(),
identity.getAttribute(attribute))
        if (claim.isEssential() && (userProfileValue == null ||
```

```
userProfileValue.isEmpty())) {
            throw new InvalidRequestException("Could not provide
value for essential claim $claim")
        }
        if (userProfileValue != null && (claim.getValues() ==
null || claim.getValues().isEmpty() ||
claim.getValues().contains(userProfileValue))) {
            return [(claim.getName()): userProfileValue]
        }
    }
    return [:]
}

/*
 * Claim resolver which expects the user's profile attribute
value to be in the following format:
 * "language_tag|value_for_language,...".
 *
 * This resolver will take the list of requested languages from
the 'claims_locales' authorize request
 * parameter and attempt to match it to a value from the users'
profile attribute.
 * If no match is found an exception is thrown.
 */
claimLocalesClaimResolver = { attribute, claim, identity ->
    if (identity != null) {
        userProfileValue = fromSet(claim.getName(),
identity.getAttribute(attribute))
        if (userProfileValue != null) {
            localeValues =
parseLocaleAwareString(userProfileValue)
            locale = claimsLocales.find { locale ->
localeValues.containsKey(locale) }
            if (locale != null) {
                return [(claim.getName()):
localeValues.get(locale)]
            }
        }
    }
    return [:]
}

/*
 * Claim resolver which expects the user's profile attribute
value to be in the following format:
```

```
 * "language_tag|value_for_language,...".
 *
 * This resolver will take the language tag specified in the
claim object and attempt to match it to a value
 * from the users' profile attribute. If no match is found an
exception is thrown.
 */
languageTagClaimResolver = { attribute, claim, identity ->
    if (identity != null) {
        userProfileValue = fromSet(claim.getName(),
identity.getAttribute(attribute))
        if (userProfileValue != null) {
            localeValues =
parseLocaleAwareString(userProfileValue)
            if (claim.getLocale() != null) {
                if (localeValues.containsKey(claim.getLocale()))
{
                    return [(claim.getName()):
localeValues.get(claim.getLocale())]
                } else {
                    entry =
localeValues.entrySet().iterator().next()
                    return [(claim.getName() + "#" +
entry.getKey()): entry.getValue()]
                }
            } else {
                entry =
localeValues.entrySet().iterator().next()
                return [(claim.getName()): entry.getValue()]
            }
        }
    }
    return [:]
}

/*
 * Given a string "en|English,jp|Japenese,fr_CA|French Canadian"
will return map of locale -> value.
 */
parseLocaleAwareString = { s ->
    return result = s.split(",").collectEntries { entry ->
        split = entry.split("\\|")
        [(split[0]): value = split[1]]
    }
}
```

```
// ---^^^^^^^^^^---- EXAMPLE CLAIM ATTRIBUTE RESOLVER FUNCTIONS -
--^^^^^^^^^^---

// -------------- UPDATE THIS TO CHANGE CLAIM TO ATTRIBUTE
MAPPING FUNCTIONS ---------------
/*
 * List of claim resolver mappings.
 */
// [ {claim}: {attribute retriever}, ... ]
claimAttributes = [
        "email": userProfileClaimResolver.curry("mail"),
        "address": { claim, identity -> [ "address" :
userAddressClaimResolver(claim, identity) ] },
        "phone_number":
userProfileClaimResolver.curry("telephonenumber"),
        "given_name":
userProfileClaimResolver.curry("givenname"),
        "zoneinfo":
userProfileClaimResolver.curry("preferredtimezone"),
        "family_name": userProfileClaimResolver.curry("sn"),
        "locale":
userProfileClaimResolver.curry("preferredlocale"),
        "name": userProfileClaimResolver.curry("cn")
]


// -------------- UPDATE THIS TO CHANGE SCOPE TO CLAIM MAPPINGS
--------------
/*
 * Map of scopes to claim objects.
 */
// {scope}: [ {claim}, ... ]
scopeClaimsMap = [
        "email": [ "email" ],
        "address": [ "address" ],
        "phone": [ "phone_number" ],
        "profile": [ "given_name", "zoneinfo", "family_name",
"locale", "name" ]
]


// ---------------- UPDATE BELOW FOR ADVANCED USAGES ----------
--------
if (logger.messageEnabled()) {
    scopes.findAll { s -> !("openid".equals(s) ||
```

```
scopeClaimsMap.containsKey(s)) }.each { s ->

logger.message("OpenAMScopeValidator.getUserInfo()::Message:
scope not bound to claims: $s")
    }
}

/*
 * Computes the claims return key and value. The key may be a
different value if the claim value is not in
 * the requested language.
 */
def computeClaim = { claim ->
    try {
        claimResolver = claimAttributes.get(claim.getName(), {
claimObj, identity -> defaultClaimResolver(claim)})
        claimResolver(claim, identity)
    } catch (IdRepoException e) {
        if (logger.warningEnabled()) {
            logger.warning("OpenAMScopeValidator.getUserInfo():
Unable to retrieve attribute=$attribute", e);
        }
    } catch (SSOException e) {
        if (logger.warningEnabled()) {
            logger.warning("OpenAMScopeValidator.getUserInfo():
Unable to retrieve attribute=$attribute", e);
        }
    }
}

/*
 * Converts requested scopes into claim objects based on the
scope mappings in scopeClaimsMap.
 */
def convertScopeToClaims = {
    scopes.findAll { scope -> "openid" != scope &&
scopeClaimsMap.containsKey(scope) }.collectMany { scope ->
        scopeClaimsMap.get(scope).collect { claim ->
            new Claim(claim)
        }
    }
}

// Creates a full list of claims to resolve from requested
scopes, claims provided by AS and requested claims
```

```
def claimsToResolve = convertScopeToClaims() + claimObjects +
requestedTypedClaims

// Computes the claim return key and values for all requested
claims
computedClaims = claimsToResolve.collectEntries() { claim ->
    result = computeClaim(claim)
}

// Computes composite scopes
def compositeScopes = scopeClaimsMap.findAll { scope ->
    scopes.contains(scope.key)
}

return new UserInfoClaims((Map)computedClaims,
(Map)compositeScopes)
```

Open oidc-claims-extension.groovy in your browser.

salesforce-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
        field("id", rawProfile.user_id),
        field("displayName", rawProfile.name),
        field("givenName", rawProfile.given_name),
        field("familyName", rawProfile.family_name),
        field("photoUrl", rawProfile.picture),
        field("email", rawProfile.email),
```

```
        field("username", rawProfile.email),
        field("locale", rawProfile.zoneInfo)))
```

Open salesforce-profile-normalization.groovy in your browser.

social-idp-profile-transformation.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/* Default Social Identity Provider Profile Transformation
script to use as a template for new scripts */
```

Open social-idp-profile-transformation.groovy in your browser.

twitter-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
        field("id", rawProfile.id_str),
        field("displayName", rawProfile.name),
```

```
        field("photoUrl", rawProfile.profile_image_url),
        field("email", rawProfile.email),
        field("username", rawProfile.screen_name)))
```

Open twitter-profile-normalization.groovy in your browser.

vkontakte-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
        field("id", rawProfile.id),
        field("displayName", rawProfile.first_name),
        field("givenName", rawProfile.first_name),
        field("familyName", rawProfile.last_name),
        field("photoUrl", rawProfile.photo_50),
        field("email", rawProfile.email),
        field("username", rawProfile.email)))
```

Open vkontakte-profile-normalization.groovy in your browser.

wechat-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
```

```
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
        field("id", rawProfile.openid),
        field("displayName", rawProfile.nickname),
        field("photoUrl", rawProfile.headimgurl),
        field("username", rawProfile.nickname)))
```

Open wechat-profile-normalization.groovy in your browser.

wordpress-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
        field("id", rawProfile.username),
        field("displayName", rawProfile.display_name),
        field("photoUrl", rawProfile.avatar_URL),
        field("email", rawProfile.email),
        field("username", rawProfile.username)))
```

Open wordpress-profile-normalization.groovy in your browser.

yahoo-profile-normalization.groovy

▼ View script

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

import static org.forgerock.json.JsonValue.field
import static org.forgerock.json.JsonValue.json
import static org.forgerock.json.JsonValue.object

return json(object(
        field("id", rawProfile.sub),
        field("displayName", rawProfile.name),
        field("givenName", rawProfile.given_name),
        field("familyName", rawProfile.family_name),
        field("photoUrl", rawProfile.picture),
        field("email", rawProfile.email),
        field("username", rawProfile.email),
        field("locale", rawProfile.locale)))
```

Open yahoo-profile-normalization.groovy in your browser.

JavaScript Samples

The comments describe the variables available in the execution context of the script, so use them for reference even if your script's function differs from the sample:

- amazon-profile-normalization.js
- apple-profile-normalization.js
- authentication-client-side.js
- authentication-server-side.js
- authentication-tree-decision-node.js
- deviceIdMatch-client-side.js
- deviceIdMatch-server-side.js
- deviceProfileMatch-decision-node.js
- facebook-profile-normalization.js

- fontdetector.js
- google-profile-normalization.js
- instagram-profile-normalization.js
- itsme-profile-normalization.js
- linkedIn-profile-normalization.js
- microsoft-profile-normalization.js
- normalized-profile-to-identity.js
- normalized-profile-to-managed-user.js
- oauth2-access-token-modification.js
- oauth2-authorize-endpoint-data-provider.js
- oauth2-evaluate-scope.js
- oauth2-may-act.js
- oauth2-validate-scope.js
- oidc-claims-extension.js
- policy-condition.js
- saml2-idp-adapter.js
- saml2-idp-attribute-mapper.js
- salesforce-profile-normalization.js
- social-idp-profile-transformation.js
- twitter-profile-normalization.js
- vkontakte-profile-normalization.js
- wechat-profile-normalization.js
- wordpress-profile-normalization.js
- yahoo-profile-normalization.js

amazon-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
```

```
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
```

```
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
```

```
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('user_id'));
    normalizedProfileData.put('displayName',
rawProfile.get('name'));
    normalizedProfileData.put('email', rawProfile.get('email'));
    normalizedProfileData.put('username',
rawProfile.get('email'));

    return normalizedProfileData;
}());
```

Open amazon-profile-normalization.js in your browser.

apple-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021-2022 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
```

```
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
```

```
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required to be not empty:
 *          username, givenName, familyName, email.
 *
 *          From RFC4517: A value of the Directory String syntax
is a string of one or more
 *          arbitrary characters from the Universal Character
Set (UCS).
 *          A zero-length character string is not permitted.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
```

```
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    var email = 'change@me.com';
    var subjectId = rawProfile.get('sub');
    var firstName = ' ';
    var lastName = ' ';
    var name = '';
    var username = subjectId;

    if(rawProfile.isDefined('email') &&
rawProfile.get('email').isNotNull()) { // User can elect to not
share their email
        email = rawProfile.get('email').asString();
        username = email;
    }
    if (rawProfile.isDefined('name') &&
rawProfile.get('name').isNotNull()) {
        if (rawProfile.name.isDefined('firstName') &&
rawProfile.get('firstName').isNotNull()) {
            firstName =
rawProfile.get('name').get('firstName').asString()
        }
        if (rawProfile.name.isDefined('lastName') &&
rawProfile.get('lastName')..isNotNull()) {
            lastName =
rawProfile.get('name').get('lastName').asString()
        }
    }

    var hasFirstName = firstName && firstName.trim().length > 0
```

```
    var hasLastName = lastName && lastName.trim().length > 0
    name = (hasFirstName ? firstName : '') + (hasLastName ?
(hasFirstName ? ' ' : '') + lastName : '')
    name =  name ? name : ' '

    normalizedProfileData.put('id', subjectId);
    normalizedProfileData.put('displayName', name);
    normalizedProfileData.put('email', email);
    normalizedProfileData.put('givenName', firstName);
    normalizedProfileData.put('familyName', lastName);
    normalizedProfileData.put('username', username);

    return normalizedProfileData;
}());
```

Open apple-profile-normalization.js in your browser.

authentication-client-side.js

▼ View script

```
/*
 * Copyright 2016-2022 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */
/* Default Authentication client side script to use as a
template for new scripts */
```

Open authentication-client-side.js in your browser.

authentication-server-side.js

▼ View script

```
/*
 * Copyright 2015-2022 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
```

```
subject
 * to such license between the licensee and ForgeRock AS.
 */

var START_TIME = 9;  // 9am
var END_TIME   = 17; // 5pm
var longitude, latitude;
var localTime;

logger.message("Starting scripted authentication");
logger.message("User: " + username);

var userPostalAddress = getUserPostalAddress();
logger.message("User address: " + userPostalAddress);

getLongitudeLatitudeFromUserPostalAddress();
getLocalTime();

logger.message("Current time at the users location: " +
localTime.getHours());
if (localTime.getHours() < START_TIME || localTime.getHours() >
END_TIME) {
    logger.error("Login forbidden outside work hours!");
    authState = FAILED;
} else {
    logger.message("Authentication allowed!");
    authState = SUCCESS;
}

function getLongitudeLatitudeFromUserPostalAddress() {

    var request = new org.forgerock.http.protocol.Request();

request.setUri("http://maps.googleapis.com/maps/api/geocode/json
?address=" + encodeURIComponent(userPostalAddress));
        request.setMethod("GET");
        //the above URI has to be extended with an API_KEY if
used in a frequent manner
        //see documentation:
https://developers.google.com/maps/documentation/geocoding/intro

    var response = httpClient.send(request).get();
    logResponse(response);

    var geocode = JSON.parse(response.getEntity());
```

```
    var i;
    for (i = 0; i < geocode.results.length; i++) {
        var result = geocode.results[i];
        latitude = result.geometry.location.lat;
        longitude = result.geometry.location.lng;

            logger.message("latitude:" + latitude + "
longitude:" + longitude);
    }
}

function getLocalTime() {

    var now = new Date().getTime() / 1000;
    var location = "location=" + latitude + "," + longitude;
    var timestamp = "timestamp=" + now;

    var request = new org.forgerock.http.protocol.Request();

request.setUri("https://maps.googleapis.com/maps/api/timezone/js
on?" + location + "&" + timestamp);
        request.setMethod("GET");
        //the above URI has to be extended with an API_KEY if
used in a frequent manner
        //see documentation:
https://developers.google.com/maps/documentation/timezone/intro

    var response = httpClient.send(request).get();
    logResponse(response);

    var timezone = JSON.parse(response.getEntity());
    var localTimestamp = parseInt(now) +
parseInt(timezone.dstOffset) + parseInt(timezone.rawOffset);
    localTime = new Date(localTimestamp*1000);
}

function getUserPostalAddress() {
    var userAddressSet = idRepository.getAttribute(username,
"postalAddress");
    if (userAddressSet == null || userAddressSet.isEmpty()) {
        logger.warning("No address specified for user: " +
username);
        return false;
    }
    return userAddressSet.iterator().next()
```

```
}

function logResponse(response) {
    logger.message("User REST Call. Status: " +
response.getStatus() + ", Body: " + response.getEntity());
}
```

Open authentication-server-side.js in your browser.

authentication-tree-decision-node.js

▼ View script

```
/*
  - Data made available by nodes that have already executed are
available in the sharedState variable.
  - The script should set outcome to either "true" or "false".
 */

outcome = "true";
```

Open authentication-tree-decision-node.js in your browser.

config-provider-node.js

▼ View script

```
/*
 * Copyright 2021-2022 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/**
 * The following script is a simplified template for
understanding how to build
 * up a config Map object with custom values. The Config
Provider Node will then
 * provide this config Map to the desired node type. It is
important that the Map
 * you build here is named 'config'.
```

```
 *
 * Defined variables:
 *
 * nodeState - Node State (1)
 *           Always present, this represents the current values
stored in the node state.
 *
 * idRepository - Profile Data (2)
 *           Always present, a repository to retrieve user
information.
 *
 * secrets - Credentials and Secrets (3)
 *           Always present, an interface to access the Secrets
API from a scripting context.
 *
 * requestHeaders (4) - Map (5)
 *           Always present, an object that provides methods for
accessing headers in the login request.
 *
 * logger - Debug Logging (6)
 *          Always present, the debug logger instance.
 *
 * httpClient - HTTP Client (7)
 *          Always present, the HTTP client that can be used to
make external HTTP requests.
 *
 * realm - String (primitive).
 *          Always present, the name of the realm the user is
authenticating to.
 *
 * existingSession - Map<String, String> (5)
 *          Present if the request contains the session cookie,
the user's session object. The returned map from
 *          SSOToken.getProperties() (8)
 *
 * requestParameters - Map (5)
 *          Always present, the object that contains the
authentication request parameters.
 *
 *
 * Outputs:
 *
 * config - Map (5)
 *           Define and fill a Map object named 'config' with
custom values, this will define the configuration for the
```

```
 *           associated node selected in the ConfigProviderNode.
 *
 * Reference:
 * (1) Node State -
https://backstage.forgerock.com/docs/idcloud-
am/latest/authentication-guide/scripting-api-
node.html#scripting-api-node-nodeState
 * (2) Profile Data -
https://backstage.forgerock.com/docs/am/7.1/authentication-
guide/scripting-api-node.html#scripting-api-node-id-repo
 * (3) Credentials and Secrets -
https://backstage.forgerock.com/docs/am/7.1/authentication-
guide/scripting-api-node.html#scripting-api-authn-secrets
 * (4) Request Headers -
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * (5) Map -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/Map.html
 * (6) Debug Logging -
https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * (7) HTTP Client -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
http/Client.html.
 * (8) SSOToken -
https://backstage.forgerock.com/docs/am/7/apidocs/com/iplanet/ss
o/SSOToken.html.
 */

config = {
    "key0": {"subKey": "value0"},
    "key1": "value1"
};
```

Open config-provider-node.js in your browser.

deviceIdMatch-client-side.js

▼ View script

```
/*
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS HEADER.
 *
```

```
 * Copyright (c) 2009 Sun Microsystems Inc. All Rights Reserved
 *
 * The contents of this file are subject to the terms
 * of the Common Development and Distribution License
 * (the License). You may not use this file except in
 * compliance with the License.
 *
 * You can obtain a copy of the License at
 * https://opensso.dev.java.net/public/CDDLv1.0.html or
 * opensso/legal/CDDLv1.0.txt
 * See the License for the specific language governing
 * permission and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL
 * Header Notice in each file and include the License file
 * at opensso/legal/CDDLv1.0.txt.
 * If applicable, add the following below the CDDL Header,
 * with the fields enclosed by brackets [] replaced by
 * your own identifying information:
 * "Portions Copyrighted [year] [name of copyright owner]"
 *
 */
/*
 * Portions Copyrighted 2013 Syntegrity.
 * Portions Copyrighted 2013-2014 ForgeRock AS.
 */

var collectScreenInfo = function () {
        var screenInfo = {};
        if (screen) {
            if (screen.width) {
                screenInfo.screenWidth = screen.width;
            }

            if (screen.height) {
                screenInfo.screenHeight = screen.height;
            }

            if (screen.pixelDepth) {
                screenInfo.screenColourDepth =
screen.pixelDepth;
            }
        } else {
            console.warn("Cannot collect screen information.
screen is not defined.");
```

```
        }
        return screenInfo;
    },
    collectTimezoneInfo = function () {
        var timezoneInfo =  {}, offset = new
Date().getTimezoneOffset();

        if (offset) {
            timezoneInfo.timezone = offset;
        } else {
            console.warn("Cannot collect timezone information.
timezone is not defined.");
        }

        return timezoneInfo;
    },
    collectBrowserPluginsInfo = function () {

        if (navigator && navigator.plugins) {
            var pluginsInfo = {}, i, plugins =
navigator.plugins;
            pluginsInfo.installedPlugins = "";

            for (i = 0; i < plugins.length; i++) {
                pluginsInfo.installedPlugins =
pluginsInfo.installedPlugins + plugins[i].filename + ";";
            }

            return pluginsInfo;
        } else {
            console.warn("Cannot collect browser plugin
information. navigator.plugins is not defined.");
            return {};
        }

    },
// Getting geolocation takes some time and is done
asynchronously, hence need a callback which is called once
geolocation is retrieved.
    collectGeolocationInfo = function (callback) {
        var geolocationInfo = {},
            successCallback = function(position) {
                geolocationInfo.longitude =
position.coords.longitude;
                geolocationInfo.latitude =
```

```
position.coords.latitude;
                callback(geolocationInfo);
            }, errorCallback = function(error) {
                console.warn("Cannot collect geolocation
information. " + error.code + ": " + error.message);
                callback(geolocationInfo);
            };
        if (navigator && navigator.geolocation) {
            // NB: If user chooses 'Not now' on Firefox neither
callback gets called
            //     https://bugzilla.mozilla.org/show_bug.cgi?
id=675533

navigator.geolocation.getCurrentPosition(successCallback,
errorCallback);
        } else {
            console.warn("Cannot collect geolocation
information. navigator.geolocation is not defined.");
            callback(geolocationInfo);
        }
    },
    collectBrowserFontsInfo = function () {
        var fontsInfo = {}, i, fontsList =
["cursive","monospace","serif","sans-
serif","fantasy","default","Arial","Arial Black",
            "Arial Narrow","Arial Rounded MT Bold","Bookman Old
Style","Bradley Hand ITC","Century","Century Gothic",
            "Comic Sans MS","Courier","Courier
New","Georgia","Gentium","Impact","King","Lucida
Console","Lalit",
            "Modena","Monotype
Corsiva","Papyrus","Tahoma","TeX","Times","Times New
Roman","Trebuchet MS","Verdana",
            "Verona"];
        fontsInfo.installedFonts = "";

        for (i = 0; i < fontsList.length; i++) {
            if (fontDetector.detect(fontsList[i])) {
                fontsInfo.installedFonts =
fontsInfo.installedFonts + fontsList[i] + ";";
            }
        }
        return fontsInfo;
    },
    devicePrint = {};
```

```
devicePrint.screen = collectScreenInfo();
devicePrint.timezone = collectTimezoneInfo();
devicePrint.plugins = collectBrowserPluginsInfo();
devicePrint.fonts = collectBrowserFontsInfo();

if (navigator.userAgent) {
    devicePrint.userAgent = navigator.userAgent;
}
if (navigator.appName) {
    devicePrint.appName = navigator.appName;
}
if (navigator.appCodeName) {
    devicePrint.appCodeName = navigator.appCodeName;
}
if (navigator.appVersion) {
    devicePrint.appVersion = navigator.appVersion;
}
if (navigator.appMinorVersion) {
    devicePrint.appMinorVersion = navigator.appMinorVersion;
}
if (navigator.buildID) {
    devicePrint.buildID = navigator.buildID;
}
if (navigator.platform) {
    devicePrint.platform = navigator.platform;
}
if (navigator.cpuClass) {
    devicePrint.cpuClass = navigator.cpuClass;
}
if (navigator.oscpu) {
    devicePrint.oscpu = navigator.oscpu;
}
if (navigator.product) {
    devicePrint.product = navigator.product;
}
if (navigator.productSub) {
    devicePrint.productSub = navigator.productSub;
}
if (navigator.vendor) {
    devicePrint.vendor = navigator.vendor;
}
if (navigator.vendorSub) {
    devicePrint.vendorSub = navigator.vendorSub;
}
```

```
if (navigator.language) {
    devicePrint.language = navigator.language;
}
if (navigator.userLanguage) {
    devicePrint.userLanguage = navigator.userLanguage;
}
if (navigator.browserLanguage) {
    devicePrint.browserLanguage = navigator.browserLanguage;
}
if (navigator.systemLanguage) {
    devicePrint.systemLanguage = navigator.systemLanguage;
}

// Attempt to collect geo-location information and return this
with the data collected so far.
// Otherwise, if geo-location fails or takes longer than 30
seconds, auto-submit the data collected so far.
autoSubmitDelay = 30000;
output.value = JSON.stringify(devicePrint);
collectGeolocationInfo(function(geolocationInfo) {
    devicePrint.geolocation = geolocationInfo;
    output.value = JSON.stringify(devicePrint);
    submit();
});
```

Open deviceIdMatch-client-side.js in your browser.

deviceIdMatch-server-side.js

▼ View script

```
/*
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS HEADER.
 *
 * Copyright (c) 2009 Sun Microsystems Inc. All Rights Reserved
 *
 * The contents of this file are subject to the terms
 * of the Common Development and Distribution License
 * (the License). You may not use this file except in
 * compliance with the License.
 *
 * You can obtain a copy of the License at
 * https://opensso.dev.java.net/public/CDDLv1.0.html or
 * opensso/legal/CDDLv1.0.txt
 * See the License for the specific language governing
```

```
 * permission and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL
 * Header Notice in each file and include the License file
 * at opensso/legal/CDDLv1.0.txt.
 * If applicable, add the following below the CDDL Header,
 * with the fields enclosed by brackets [] replaced by
 * your own identifying information:
 * "Portions Copyrighted [year] [name of copyright owner]"
 *
 */
/*
 * Portions Copyrighted 2013 Syntegrity.
 * Portions Copyrighted 2013-2018 ForgeRock AS.
 */

var ScalarComparator = {}, ScreenComparator = {},
MultiValueComparator = {}, UserAgentComparator = {},
GeolocationComparator = {};

var config = {
    profileExpiration: 30,              //in days
    maxProfilesAllowed: 5,
    maxPenaltyPoints: 0,
    attributes: {
        screen: {
            required: true,
            comparator: ScreenComparator,
            args: {
                penaltyPoints: 50
            }
        },
        plugins: {
            installedPlugins: {
                required: false,
                comparator: MultiValueComparator,
                args: {
                    maxPercentageDifference: 10,
                    maxDifferences: 5,
                    penaltyPoints: 100
                }
            }
        },
        fonts: {
            installedFonts: {
```

```
                required: false,
                comparator: MultiValueComparator,
                args: {
                    maxPercentageDifference: 10,
                    maxDifferences: 5,
                    penaltyPoints: 100
                }
            }
        },
        timezone: {
            timezone: {
                required: false,
                comparator: ScalarComparator,
                args: {
                    penaltyPoints: 100
                }
            }
        },
        userAgent: {
            required: true,
            comparator: UserAgentComparator,
            args: {
                ignoreVersion: true,
                penaltyPoints: 100
            }
        },
        geolocation: {
            required: false,
            comparator: GeolocationComparator,
            args: {
                allowedRange: 100,                      //in
miles
                penaltyPoints: 100
            }
        }
    }
};

//-------------------------------------------------------------
-------------//
//                          Comparator functions
//
//-------------------------------------------------------------
-------------//
```

```
var all, any, calculateDistance, calculateIntersection,
calculatePercentage, nullOrUndefined, splitAndTrim,
    undefinedLocation;

// ComparisonResult

/**
 * Constructs an instance of a ComparisonResult with the given
penalty points.
 *
 * @param penaltyPoints (Number) The penalty points for the
comparison (defaults to 0).
 * @param additionalInfoInCurrentValue (boolean) Whether the
current value contains more information
 *                                                  than the stored
value (defaults to false).
 */
function ComparisonResult() {

    var penaltyPoints = 0,
        additionalInfoInCurrentValue = false;

    if (arguments[0] !== undefined && arguments[1] !==
undefined) {
        penaltyPoints = arguments[0];
        additionalInfoInCurrentValue = arguments[1];
    }

    if (arguments[0] !== undefined && arguments[1] ===
undefined) {
        if (typeof(arguments[0]) === "boolean") {
            additionalInfoInCurrentValue = arguments[0];
        } else {
            penaltyPoints = arguments[0];
        }
    }

    this.penaltyPoints = penaltyPoints;
    this.additionalInfoInCurrentValue =
additionalInfoInCurrentValue;

}

ComparisonResult.ZERO_PENALTY_POINTS = new ComparisonResult(0);
```

```
/**
 * Static method for functional programming.
 *
 * @return boolean true if comparisonResult.isSuccessful().
 */
ComparisonResult.isSuccessful =  function(comparisonResult) {
    return comparisonResult.isSuccessful();
};


/**
 * Static method for functional programming.
 *
 * @return boolean true if
comparisonResult.additionalInfoInCurrentValue.
 */
ComparisonResult.additionalInfoInCurrentValue =
function(comparisonResult) {
    return comparisonResult.additionalInfoInCurrentValue;
};

/**
 * Comparison function that can be provided as an argument to
array.sort
 */
ComparisonResult.compare = function(first, second) {
    if (nullOrUndefined(first) && nullOrUndefined(second)) {
        return 0;
    } else if (nullOrUndefined(first)) {
        return -1;
    } else if (nullOrUndefined(second)) {
        return 1;
    } else {
        if (first.penaltyPoints !== second.penaltyPoints) {
            return first.penaltyPoints - second.penaltyPoints;
        } else {
            return (first.additionalInfoInCurrentValue ? 1 : 0)
- (second.additionalInfoInCurrentValue ? 1 : 0);
        }
    }
};

/**
 * Amalgamates the given ComparisonResult into this
ComparisonResult.
```

```
 *
 * @param comparisonResult The ComparisonResult to include.
 */
ComparisonResult.prototype.addComparisonResult =
function(comparisonResult) {
    this.penaltyPoints += comparisonResult.penaltyPoints;
    if (comparisonResult.additionalInfoInCurrentValue) {
        this.additionalInfoInCurrentValue =
comparisonResult.additionalInfoInCurrentValue;
    }
};

/**
 * Returns true if no penalty points have been assigned for the
comparison.
 *
 * @return boolean true if the comparison was successful.
 */
ComparisonResult.prototype.isSuccessful = function() {
    return nullOrUndefined(this.penaltyPoints) ||
this.penaltyPoints === 0;
};

/**
 * Compares two simple objects (String|Number) and if they are
equal then returns a ComparisonResult with zero
 * penalty points assigned, otherwise returns a ComparisonResult
with the given number of penalty points assigned.
 *
 * @param currentValue (String|Number) The current value.
 * @param storedValue (String|Number) The stored value.
 * @param config: {
 *            "penaltyPoints": (Number) The number of penalty
points.
 *        }
 * @return ComparisonResult.
 */
ScalarComparator.compare = function (currentValue, storedValue,
config) {
    if (logger.messageEnabled()) {
        logger.message("StringComparator.compare:currentValue: "
+ JSON.stringify(currentValue));
        logger.message("StringComparator.compare:storedValue: "
+ JSON.stringify(storedValue));
        logger.message("StringComparator.compare:config: " +
```

```
JSON.stringify(config));
    }
    if (config.penaltyPoints === 0) {
        return ComparisonResult.ZERO_PENALTY_POINTS;
    }

    if (!nullOrUndefined(storedValue)) {
        if (nullOrUndefined(currentValue) || currentValue !==
storedValue) {
            return new ComparisonResult(config.penaltyPoints);
        }
    } else if (!nullOrUndefined(currentValue)) {
        return new ComparisonResult(true);
    }

    return ComparisonResult.ZERO_PENALTY_POINTS;
};

/**
 * Compares two screens and if they are equal then returns a
ComparisonResult with zero penalty points assigned,
 * otherwise returns a ComparisonResult with the given number of
penalty points assigned.
 *
 * @param currentValue: {
 *            "screenWidth": (Number) The current client screen
width.
 *            "screenHeight": (Number) The current client screen
height.
 *            "screenColourDepth": (Number) The current client
screen colour depth.
 *        }
 * @param storedValue: {
 *            "screenWidth": (Number) The stored client screen
width.
 *            "screenHeight": (Number) The stored client screen
height.
 *            "screenColourDepth": (Number) The stored client
screen colour depth.
 *        }
 * @param config: {
 *            "penaltyPoints": (Number) The number of penalty
points.
 *        }
 * @return ComparisonResult
```

```
 */
ScreenComparator.compare = function (currentValue, storedValue,
config) {
    if (logger.messageEnabled()) {
        logger.message("ScreenComparator.compare:currentValue: "
+ JSON.stringify(currentValue));
        logger.message("ScreenComparator.compare:storedValue: "
+ JSON.stringify(storedValue));
        logger.message("ScreenComparator.compare:config: " +
JSON.stringify(config));
    }

    if (nullOrUndefined(currentValue)) {
        currentValue = {screenWidth: null, screenHeight: null,
screenColourDepth: null};
    }
    if (nullOrUndefined(storedValue)) {
        storedValue = {screenWidth: null, screenHeight: null,
screenColourDepth: null};
    }

    var comparisonResults = [
        ScalarComparator.compare(currentValue.screenWidth,
storedValue.screenWidth, config),
        ScalarComparator.compare(currentValue.screenHeight,
storedValue.screenHeight, config),
        ScalarComparator.compare(currentValue.screenColourDepth,
storedValue.screenColourDepth, config)];

    if (all(comparisonResults, ComparisonResult.isSuccessful)) {
        return new ComparisonResult(any(comparisonResults,
ComparisonResult.additionalInfoInCurrentValue));
    } else {
        return new ComparisonResult(config.penaltyPoints);
    }
};

/**
 * Splits both values using delimiter, trims every value and
compares collections of values.
 * Returns zero-result for same multi-value attributes.
 *
 * If collections are not same checks if number of differences
is less or equal maxDifferences or
 * percentage of difference is less or equal
```

```
maxPercentageDifference.
 *
 * If yes then returns zero-result with additional info, else
returns penaltyPoints-result.
 *
 * @param currentValue: (String) The current value.
 * @param storedValue: (String) The stored value.
 * @param config: {
 *            "maxPercentageDifference": (Number) The max
difference percentage in the values,
 *                                                 before the
penalty is assigned.
 *            "maxDifferences": (Number) The max number of
differences in the values,
 *                                          before the penalty
points are assigned.
 *            "penaltyPoints": (Number) The number of penalty
points.
  *        }
 * @return ComparisonResult
 */
MultiValueComparator.compare = function (currentValue,
storedValue, config) {
    if (logger.messageEnabled()) {

logger.message("MultiValueComparator.compare:currentValue: " +
JSON.stringify(currentValue));

logger.message("MultiValueComparator.compare:storedValue: " +
JSON.stringify(storedValue));
        logger.message("MultiValueComparator.compare:config: " +
JSON.stringify(config));
    }

    var delimiter = ";",
        currentValues = splitAndTrim(currentValue, delimiter),
        storedValues = splitAndTrim(storedValue, delimiter),
        maxNumberOfElements = Math.max(currentValues.length,
storedValues.length),
        numberOfTheSameElements =
calculateIntersection(currentValues, storedValues).length,
        numberOfDifferences = maxNumberOfElements -
numberOfTheSameElements,
        percentageOfDifferences =
calculatePercentage(numberOfDifferences, maxNumberOfElements);
```

```
    if (nullOrUndefined(storedValue) &&
!nullOrUndefined(currentValue)) {
        return new ComparisonResult(true);
    }

    if (logger.messageEnabled()) {
        logger.message(numberOfTheSameElements + " of " +
maxNumberOfElements + " are same");
    }

    if (maxNumberOfElements === 0) {
        logger.message("Ignored because no attributes found in
both profiles");
        return ComparisonResult.ZERO_PENALTY_POINTS;
    }

    if (numberOfTheSameElements === maxNumberOfElements) {
        logger.message("Ignored because all attributes are
same");
        return ComparisonResult.ZERO_PENALTY_POINTS;
    }

    if (numberOfDifferences > config.maxDifferences) {
        if (logger.messageEnabled()) {
            logger.message("Would be ignored if not more than "
+ config.maxDifferences + " differences");
        }
        return new ComparisonResult(config.penaltyPoints);
    }

    if (percentageOfDifferences >
config.maxPercentageDifference) {
        if (logger.messageEnabled()) {
            logger.message(percentageOfDifferences + " percents
are different");
            logger.message("Would be ignored if not more than "
+ config.maxPercentageDifference + " percent");
        }
        return new ComparisonResult(config.penaltyPoints);
    }

    if (logger.messageEnabled()) {
        logger.message("Ignored because number of differences("
+ numberOfDifferences + ") not more than "
```

```
            + config.maxDifferences);
        logger.message(percentageOfDifferences + " percents are
different");
        logger.message("Ignored because not more than " +
config.maxPercentageDifference + " percent");
    }
    return new ComparisonResult(true);
};

/**
 * Compares two User Agent Strings and if they are equal then
returns a ComparisonResult with zero penalty
 * points assigned, otherwise returns a ComparisonResult with
the given number of penalty points assigned.
 *
 * @param currentValue (String) The current value.
 * @param storedValue (String) The stored value.
 * @param config: {
 *            "ignoreVersion": (boolean) If the version numbers
in the User Agent Strings should be ignore
 *                                       in the comparison.
 *            "penaltyPoints": (Number) The number of penalty
points.
 *        }
 * @return A ComparisonResult.
 */
UserAgentComparator.compare = function (currentValue,
storedValue, config) {
    if (logger.messageEnabled()) {

logger.message("UserAgentComparator.compare:currentValue: " +
JSON.stringify(currentValue));
        logger.message("UserAgentComparator.compare:storedValue:
" + JSON.stringify(storedValue));
        logger.message("UserAgentComparator.compare:config: " +
JSON.stringify(config));
    }

    if (config.ignoreVersion) {
        // remove version number
        currentValue = nullOrUndefined(currentValue) ? null :
currentValue.replace(/[\d\.]+/g, "").trim();
        storedValue = nullOrUndefined(storedValue) ? null :
storedValue.replace(/[\d\.]+/g, "").trim();
    }
```

```
    return ScalarComparator.compare(currentValue, storedValue,
config);
};

/**
 * Compares two locations, taking into account a degree of
difference.
 *
 * @param currentValue: {
 *             "latitude": (Number) The current latitude.
 *             "longitude": (Number) The current longitude.
 *         }
 * @param storedValue: {
 *             "latitude": (Number) The stored latitude.
 *             "longitude": (Number) The stored longitude.
 *         }
 * @param config: {
 *             "allowedRange": (Number) The max difference
allowed in the two locations, before the penalty is assigned.
 *             "penaltyPoints": (Number) The number of penalty
points.
*         }
 * @return ComparisonResult
 */
GeolocationComparator.compare = function (currentValue,
storedValue, config) {
    if (logger.messageEnabled()) {

logger.message("GeolocationComparator.compare:currentValue: " +
JSON.stringify(currentValue));

logger.message("GeolocationComparator.compare:storedValue: " +
JSON.stringify(storedValue));
        logger.message("GeolocationComparator.compare:config: "
+ JSON.stringify(config));
    }

    // Check for undefined stored or current locations

    if (undefinedLocation(currentValue) &&
undefinedLocation(storedValue)) {
        return ComparisonResult.ZERO_PENALTY_POINTS;
    }
    if (undefinedLocation(currentValue) &&
```

```
!undefinedLocation(storedValue)) {
        return new ComparisonResult(config.penaltyPoints);
    }
    if (!undefinedLocation(currentValue) &&
undefinedLocation(storedValue)) {
        return new ComparisonResult(true);
    }

    // Both locations defined, therefore perform comparison

    var distance = calculateDistance(currentValue, storedValue);

    if (logger.messageEnabled()) {
        logger.message("Distance between (" +
currentValue.latitude + "," + currentValue.longitude + ") and ("
+
            storedValue.latitude + "," + storedValue.longitude +
") is " + distance + " miles");
    }

    if (parseFloat(distance.toPrecision(5)) === 0) {
        logger.message("Location is the same");
        return ComparisonResult.ZERO_PENALTY_POINTS;
    }

    if (distance <= config.allowedRange) {
        if (logger.messageEnabled()) {
            logger.message("Tolerated because distance not more
then " + config.allowedRange);
        }
        return new ComparisonResult(true);
    } else {
        if (logger.messageEnabled()) {
            logger.message("Would be ignored if distance not
more then " + config.allowedRange);
        }
        return new ComparisonResult(config.penaltyPoints);
    }
};


//-------------------------------------------------------------
-------------//
//                    Device Print Logic - DO NOT MODIFY
//
```

```
//-------------------------------------------------------------
-------------//

// Utility functions

/**
 * Returns true if evaluating function f on each element of the
Array a returns true.
 *
 * @param a: (Array) The array of elements to evaluate
 * @param f: (Function) A single argument function for mapping
elements of the array to boolean.
 * @return boolean.
 */
all = function(a, f) {
    var i;
    for (i = 0; i < a.length; i++) {
        if (f(a[i]) === false) {
            return false;
        }
    }
    return true;
};

/**
 * Returns true if evaluating function f on any element of the
Array a returns true.
 *
 * @param a: (Array) The array of elements to evaluate
 * @param f: (Function) A single argument function for mapping
elements of the array to boolean.
 * @return boolean.
 */
any = function(a, f) {
    var i;
    for (i = 0; i < a.length; i++) {
        if (f(a[i]) === true) {
            return true;
        }
    }
    return false;
};

/**
 * Returns true if the provided location is null or has
```

```
undefined longitude or latitude values.
 *
 * @param location: {
 *            "latitude": (Number) The latitude.
 *            "longitude": (Number) The longitude.
 *        }
 * @return boolean
 */
undefinedLocation = function(location) {
    return nullOrUndefined(location) ||
nullOrUndefined(location.latitude) ||
nullOrUndefined(location.longitude);
};

/**
 * Returns true if the provided value is null or undefined.
 *
 * @param value: a value of any type
 * @return boolean
 */
nullOrUndefined = function(value) {
    return value === null || value === undefined;
};

/**
 * Calculates the distances between the two locations.
 *
 * @param first: {
 *            "latitude": (Number) The first latitude.
 *            "longitude": (Number) The first longitude.
 *        }
 * @param second: {
 *            "latitude": (Number) The second latitude.
 *            "longitude": (Number) The second longitude.
 *        }
 * @return Number The distance between the two locations.
 */
calculateDistance = function(first, second) {
    var factor = (Math.PI / 180),
        theta,
        dist;
    function degreesToRadians(degrees) {
        return degrees * factor;
    }
    function radiansToDegrees(radians) {
```

```
        return radians / factor;
    }
    theta = first.longitude - second.longitude;
    dist = Math.sin(degreesToRadians(first.latitude)) *
Math.sin(degreesToRadians(second.latitude))
        + Math.cos(degreesToRadians(first.latitude)) *
Math.cos(degreesToRadians(second.latitude))
        * Math.cos(degreesToRadians(theta));
    dist = Math.acos(dist);
    dist = radiansToDegrees(dist);
    dist = dist * 60 * 1.1515;
    return dist;
};

/**
 * Converts a String holding a delimited sequence of values into
an array.
 *
 * @param text (String) The String representation of a delimited
sequence of values.
 * @param delimiter (String) The character delimiting values
within the text String.
 * @return (Array) The comma separated values.
 */
splitAndTrim = function(text, delimiter) {

    var results = [],
        i,
        values,
        value;
    if (text === null) {
        return results;
    }

    values = text.split(delimiter);
    for (i = 0; i < values.length; i++) {
        value = values[i].trim();
        if (value !== "") {
            results.push(value);
        }
    }

    return results;
};
```

```
/**
 * Converts value to a percentage of range.
 *
 * @param value (Number) The actual number to be converted to a
percentage.
 * @param range (Number) The total number of values (i.e.
represents 100%).
 * @return (Number) The percentage.
 */
calculatePercentage = function(value, range) {
    if (range === 0) {
        return 0;
    }
    return parseFloat((value / range).toPrecision(2)) * 100;
};

/**
 * Creates a new array containing only those elements found in
both arrays received as arguments.
 *
 * @param first (Array) The first array.
 * @param second (Array) The second array.
 * @return (Array) The elements that found in first and second.
 */
calculateIntersection = function(first, second) {
    return first.filter(function(element) {
        return second.indexOf(element) !== -1;
    });
};

function getValue(obj, attributePath) {
    var value = obj,
        i;
    for (i = 0; i < attributePath.length; i++) {
        if (value === undefined) {
            return null;
        }
        value = value[attributePath[i]];
    }
    return value;
}


function isLeafNode(attributeConfig) {
    return attributeConfig.comparator !== undefined;
```

```
}

function getAttributePaths(attributeConfig, attributePath) {

    var attributePaths = [],
        attributeName,
        attrPaths,
        attrPath,
        i;

    for (attributeName in attributeConfig) {
        if (attributeConfig.hasOwnProperty(attributeName)) {

            if (isLeafNode(attributeConfig[attributeName])) {
                attrPath = attributePath.slice();
                attrPath.push(attributeName);
                attributePaths.push(attrPath);
            } else {
                attrPath = attributePath.slice();
                attrPath.push(attributeName);
                attrPaths =
getAttributePaths(attributeConfig[attributeName], attrPath);
                for (i = 0; i < attrPaths.length; i++) {
                    attributePaths.push(attrPaths[i]);
                }
            }
        }
    }

    return attributePaths;
}

function getDevicePrintAttributePaths(attributeConfig) {
    return getAttributePaths(attributeConfig, []);
}

function hasRequiredAttributes(devicePrint, attributeConfig) {

    var attributePaths =
getDevicePrintAttributePaths(attributeConfig),
        i,
        attrValue,
        attrConfig;

    for (i = 0; i < attributePaths.length; i++) {
```

```
        attrValue = getValue(devicePrint, attributePaths[i]);
        attrConfig = getValue(attributeConfig,
attributePaths[i]);

        if (attrConfig.required && attrValue === undefined) {
            logger.warning("Device Print profile missing
required attribute, " + attributePaths[i]);
            return false;
        }
    }

    logger.message("device print has required attributes");
    return true;
}

function compareDevicePrintProfiles(attributeConfig,
devicePrint, devicePrintProfiles, maxPenaltyPoints) {

    var attributePaths =
getDevicePrintAttributePaths(attributeConfig),
        dao = sharedState.get('_DeviceIdDao'),
        results,
        j,
        aggregatedComparisonResult,
        i,
        currentValue,
        storedValue,
        attrConfig,
        comparisonResult,
        selectedComparisonResult,
        selectedProfile,
        curDevicePrintProfile,
        vals;

    results = [];
    for (j = 0; j < devicePrintProfiles.length; j++) {
        curDevicePrintProfile =
JSON.parse(org.forgerock.json.JsonValue.json(devicePrintProfiles
[j]));
        aggregatedComparisonResult = new ComparisonResult();
        for (i = 0; i < attributePaths.length; i++) {

            currentValue = getValue(devicePrint,
attributePaths[i]);
```

```
            storedValue =
getValue(curDevicePrintProfile.devicePrint, attributePaths[i]);
            attrConfig = getValue(attributeConfig,
attributePaths[i]);

            if (storedValue === null) {
                comparisonResult = new
ComparisonResult(attrConfig.penaltyPoints);
            } else {
                comparisonResult =
attrConfig.comparator.compare(currentValue, storedValue,
attrConfig.args);
            }

            if (logger.messageEnabled()) {
                logger.message("Comparing attribute path: " +
attributePaths[i]
                    + ", Comparison result: successful=" +
comparisonResult.isSuccessful() + ", penaltyPoints="
                    + comparisonResult.penaltyPoints + ",
additionalInfoInCurrentValue="
                    +
comparisonResult.additionalInfoInCurrentValue);
            }

aggregatedComparisonResult.addComparisonResult(comparisonResult)
;
        }
        if (logger.messageEnabled()) {
            logger.message("Aggregated comparison result:
successful="
                + aggregatedComparisonResult.isSuccessful() + ",
penaltyPoints="
                + aggregatedComparisonResult.penaltyPoints + ",
additionalInfoInCurrentValue="
                +
aggregatedComparisonResult.additionalInfoInCurrentValue);
        }

        results.push({
            key: aggregatedComparisonResult,
            value: devicePrintProfiles[j]
        });
    }
```

```
    if (results.length === 0) {
        return null;
    }

    results.sort(function(a, b) {
        return ComparisonResult.compare(a.key, b.key);
    });
    selectedComparisonResult = results[0].key;
    if (logger.messageEnabled()) {
        logger.message("Selected comparison result: successful="
+ selectedComparisonResult.isSuccessful()
            + ", penaltyPoints=" +
selectedComparisonResult.penaltyPoints + ",
additionalInfoInCurrentValue="
            +
selectedComparisonResult.additionalInfoInCurrentValue);
    }

    selectedProfile = null;
    if (selectedComparisonResult.penaltyPoints <=
maxPenaltyPoints) {
        selectedProfile = results[0].value;
        if (logger.messageEnabled()) {
            logger.message("Selected profile: " +
selectedProfile +
                " with " +
selectedComparisonResult.penaltyPoints + " penalty points");
        }
    }

    if (selectedProfile === null) {
        return false;
    }

    /* update profile */
    selectedProfile.put("selectionCounter",

java.lang.Integer.valueOf(parseInt(selectedProfile.get("selectio
nCounter"), 10) + 1));
    selectedProfile.put("lastSelectedDate",
java.lang.Long.valueOf(new Date().getTime()));
    selectedProfile.put("devicePrint", devicePrint);

    vals = [];
    for (i = 0; i < devicePrintProfiles.length; i++) {
```

```
vals.push(org.forgerock.json.JsonValue.json(devicePrintProfiles[
i]));
    }

    dao.saveDeviceProfiles(username, realm, vals);

    return true;
}

function matchDevicePrint() {

    if (!username) {
        logger.error("Username not set. Cannot compare user's
device print profiles.");
        authState = FAILED;
    } else {

        if (logger.messageEnabled()) {
            logger.message("client devicePrint: " +
clientScriptOutputData);
        }

        var getProfiles = function () {

                function isExpiredProfile(devicePrintProfile) {
                    var expirationDate = new Date(),
                        lastSelectedDate;

expirationDate.setDate(expirationDate.getDate() -
config.profileExpiration);

                    lastSelectedDate = new
Date(devicePrintProfile.lastSelectedDate);

                    return lastSelectedDate < expirationDate;
                }

                function getNotExpiredProfiles() {
                    var profile,
                        dao = sharedState.get('_DeviceIdDao'),
                        results = [],
                        profiles,
                        iter;
```

```
                    profiles = dao.getDeviceProfiles(username,
realm);

                    if (profiles) {
                        iter = profiles.iterator();

                        while (iter.hasNext()) {
                            profile = iter.next().getObject();
                            if (!isExpiredProfile(profile)) {
                                results.push(profile);
                            }
                        }
                    }
                    if (logger.messageEnabled()) {
                        logger.message("stored non-expired
profiles: " + results);
                    }
                    return results;
                }

                return getNotExpiredProfiles();
            },
            devicePrint = JSON.parse(clientScriptOutputData),
            devicePrintProfiles = getProfiles();

        if (!hasRequiredAttributes(devicePrint,
config.attributes)) {
            logger.message("devicePrint.hasRequiredAttributes:
false");
            // Will fail this module but fall-through to next
module. Which should be OTP.
            authState = FAILED;
        } else if (compareDevicePrintProfiles(config.attributes,
devicePrint, devicePrintProfiles, config.maxPenaltyPoints)) {
            logger.message("devicePrint.hasValidProfile: true");
            authState = SUCCESS;
        } else {
            logger.message("devicePrint.hasValidProfile:
false");
            sharedState.put('devicePrintProfile',
JSON.stringify(devicePrint));
            // Will fail this module but fall-through to next
module. Which should be OTP.
            authState = FAILED;
        }
```

```
    }
}

matchDevicePrint();
```

Open deviceIdMatch-server-side.js in your browser.

deviceProfileMatch-decision-node.js

▼ View script

```
/*
 * Copyright 2020-2022 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/**
*****************************************************************
**
 *
 * The following script is a simplified template for
understanding
 * the basics of device matching. _This is not functionally
complete._
 * For a functionally complete script as well as a development
toolkit,
 * visit https://github.com/ForgeRock/forgerock-device-match-
script.
 *
 * Global node variables accessible within this scope:
 * 1. `sharedState` provides access to incoming request
 * 2. `deviceProfilesDao` provides access to stored profiles
 * 3. `outcome` variable maps to auth tree node outcomes; values
are
 *    'true', 'false', or 'unknownDevice' (notice _all_ are
strings).
 *
*****************************************************************
**/
```

```
/**
 * Get the incoming request's device profile.
 * Returns serialized JSON (type string); parsing this will
result a
 * native JS object.
 */
var incomingJson =
sharedState.get('forgeRock.device.profile').toString();
var incoming = JSON.parse(incomingJson);

/**
 * Get the incoming user's username and realm.
 * Notice the use of `.asString()`.
 */
var username = sharedState.get("username").asString();
var realm = sharedState.get("realm").asString();

/**
 * Get the user's stored profiles for appropriate realm.
 * Returns a _special_ object with methods for profile data
 */
var storedProfiles =
deviceProfilesDao.getDeviceProfiles(username, realm);

// Default to `outcome` of 'unknownDevice'
outcome = 'unknownDevice';

if (storedProfiles) {
    var i = 0;
    // NOTE: `.size()` method returns the number of stored
profiles
    var len = storedProfiles.size();

    for (i; i < len; i++) {
        /**
         * Get the stored profile.
         * Returns serialized JSON (type string); parsing this
will result
         * a native JS object.
         */
        var storedJson = storedProfiles.get(i);
        var stored = JSON.parse(storedJson);

        /**
         * Find a stored profile with the same identifier.
```

```
         */
        if (incoming.identifier === stored.identifier) {

            /**
             * Now that you've found the appropriate profile,
you will perform
             * the logic here to match the values of the
`incoming` profile
             * with that of the `stored` profile.
             *
             * The result of the matching logic is assigned to
`outcome`. Since
             * we have profiles of the same identifier, the
value (type string)
             * should now be either 'true' or 'false'
(properties matched or not).
             *
             * For more information about this topic, visit this
Github repo:
             * https://github.com/ForgeRock/forgerock-device-
match-script
             */
            outcome = 'false';
        }
    }
}
```

Open deviceProfileMatch-decision-node.js in your browser.

facebook-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
```

```
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
```

```
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
```

```
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData = frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('id'));
    normalizedProfileData.put('displayName', rawProfile.get('name'));
    normalizedProfileData.put('givenName', rawProfile.get('first_name'));
    normalizedProfileData.put('familyName', rawProfile.get('last_name'));
    normalizedProfileData.put('photoUrl', rawProfile.get('picture').get('data').get('url'));
    normalizedProfileData.put('email', rawProfile.get('email'));
    normalizedProfileData.put('username', rawProfile.get('email'));

    return normalizedProfileData;
}());
```

Open facebook-profile-normalization.js in your browser.

fontdetector.js

▼ View script

```
var fontDetector = (function () {
    /**
     * JavaScript code to detect available availability of a
     * particular font in a browser using JavaScript and CSS.
     *
     * Author : Lalit Patel
     * Website: http://www.lalit.org/lab/javascript-css-font-detect/
     * License: Apache Software License 2.0
     *          http://www.apache.org/licenses/LICENSE-2.0
     * Version: 0.15 (21 Sep 2009)
```

```
     *          Changed comparision font to default from sans-
default-default,
     *          as in FF3.0 font of child element didn't
fallback
     *          to parent element if the font is missing.
     * Version: 0.2 (04 Mar 2012)
     *          Comparing font against all the 3 generic font
families ie,
     *          'monospace', 'sans-serif' and 'sans'. If it
doesn't match all 3
     *          then that font is 100% not available in the
system
     * Version: 0.3 (24 Mar 2012)
     *          Replaced sans with serif in the list of
baseFonts
     */
    /*
     * Portions Copyrighted 2013 ForgeRock AS.
     */
    var detector = {}, baseFonts, testString, testSize, h, s,
defaultWidth = {}, defaultHeight = {}, index;

    // a font will be compared against all the three default
fonts.
    // and if it doesn't match all 3 then that font is not
available.
    baseFonts = ['monospace', 'sans-serif', 'serif'];

    //we use m or w because these two characters take up the
maximum width.
    // And we use a LLi so that the same matching fonts can get
separated
    testString = "mmmmmmmmmmlli";

    //we test using 72px font size, we may use any size. I guess
larger the better.
    testSize = '72px';

    h = document.getElementsByTagName("body")[0];

    // create a SPAN in the document to get the width of the
text we use to test
    s = document.createElement("span");
    s.style.fontSize = testSize;
    s.innerHTML = testString;
```

```
    for (index in baseFonts) {
        //get the default width for the three base fonts
        s.style.fontFamily = baseFonts[index];
        h.appendChild(s);
        defaultWidth[baseFonts[index]] = s.offsetWidth; //width
for the default font
        defaultHeight[baseFonts[index]] = s.offsetHeight;
//height for the defualt font
        h.removeChild(s);
    }

    detector.detect = function(font) {
        var detected = false, index, matched;
        for (index in baseFonts) {
            s.style.fontFamily = font + ',' + baseFonts[index];
// name of the font along with the base font for fallback.
            h.appendChild(s);
            matched = (s.offsetWidth !==
defaultWidth[baseFonts[index]] || s.offsetHeight !==
defaultHeight[baseFonts[index]]);
            h.removeChild(s);
            detected = detected || matched;
        }
        return detected;
    };

    return detector;
}());
```

Open fontdetector.js in your browser.

google-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */
```

```
/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - JsonValue (1).
 *              The social identity provider profile information
for the authenticating user.
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
```

```
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
```

```
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('sub'));
    normalizedProfileData.put('displayName',
rawProfile.get('name'));
    normalizedProfileData.put('givenName',
rawProfile.get('given_name'));
    normalizedProfileData.put('familyName',
rawProfile.get('family_name'));
    normalizedProfileData.put('photoUrl',
rawProfile.get('picture'));
    normalizedProfileData.put('email', rawProfile.get('email'));
    normalizedProfileData.put('username',
rawProfile.get('email'));
    normalizedProfileData.put('locale',
rawProfile.get('locale'));

    return normalizedProfileData;
}());
```

Open google-profile-normalization.js in your browser.

instagram-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
```

```
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
```

```
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
```

```
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('id'));
    normalizedProfileData.put('username',
rawProfile.get('username'));

    return normalizedProfileData;
}());
```

Open instagram-profile-normalization.js in your browser.

itsme-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
```

```
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
```

```
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */
```

```
(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('sub'));
    normalizedProfileData.put('displayName',
rawProfile.get('name'));
    normalizedProfileData.put('givenName',
rawProfile.get('given_name'));
    normalizedProfileData.put('familyName',
rawProfile.get('family_name'));
    normalizedProfileData.put('username',
rawProfile.get('email'));
    normalizedProfileData.put('email', rawProfile.get('email'));

    return normalizedProfileData;
}());
```

Open itsme-profile-normalization.js in your browser.

linkedIn-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
```

```
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
```

```
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */
```

```
(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('id'));
    normalizedProfileData.put('givenName',
rawProfile.get('firstName').get('localized').get(0));
    normalizedProfileData.put('familyName',
rawProfile.get('lastName').get('localized').get(0));
    normalizedProfileData.put('photoUrl',
rawProfile.get('profilePicture').get('displayImage'));
    normalizedProfileData.put('email',
rawProfile.get('elements').get(0).get('handle~').get('emailAddre
ss'));
    normalizedProfileData.put('username',
rawProfile.get('elements').get(0).get('handle~').get('emailAddre
ss'));

    return normalizedProfileData;
}());
```

Open linkedIn-profile-normalization.js in your browser.

microsoft-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
```

```
Node.
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
```

```
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
```

```
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('id'));
    normalizedProfileData.put('displayName',
rawProfile.get('displayName'));
    normalizedProfileData.put('givenName',
rawProfile.get('givenName'));
    normalizedProfileData.put('familyName',
rawProfile.get('surname'));
    normalizedProfileData.put('email',
rawProfile.get('userPrincipalName'));
    normalizedProfileData.put('username',
rawProfile.get('userPrincipalName'));

    return normalizedProfileData;
}());
```

Open microsoft-profile-normalization.js in your browser.

normalized-profile-to-identity.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script translates the normalized social identity profile
information for the authenticating user
 * into the identity object key/value pairs.
```

```
 *
 * Defined variables:
 * normalizedProfile - The social identity provider profile
information for the authenticating user
 *                     in a standard format expected by this
node.
 *                     JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
```

```
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var identityData =
frJava.JsonValue.json(frJava.JsonValue.object());

    identityData.put('givenName',
normalizedProfile.get('givenName'));
    identityData.put('sn', normalizedProfile.get('familyName'));
    identityData.put('mail', normalizedProfile.get('email'));
    identityData.put('cn',
normalizedProfile.get('displayName'));
    identityData.put('userName',
normalizedProfile.get('username'));
    identityData.put('iplanet-am-user-alias-list', selectedIdp +
'-' + normalizedProfile.get('id').asString());

    return identityData;
}());
```

Open normalized-profile-to-identity.js in your browser.

normalized-profile-to-managed-user.js

▼ View script

```
/*
 * Copyright 2021-2022 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script translates the normalized social identity profile
information for the authenticating user
 * into the managed user object key/value pairs.
 *
 * Defined variables:
 * normalizedProfile - The social identity provider profile
information for the authenticating user
 *                     in a standard format expected by this
node.
 *                     JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
```

```
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var managedUserData =
frJava.JsonValue.json(frJava.JsonValue.object());

    managedUserData.put('givenName',
normalizedProfile.get('givenName'));
    managedUserData.put('sn',
normalizedProfile.get('familyName'));
    managedUserData.put('mail', normalizedProfile.get('email'));
    managedUserData.put('userName',
normalizedProfile.get('username'));
```

```
    if (normalizedProfile.get('postalAddress').isNotNull()) {
        managedUserData.put('postalAddress',
normalizedProfile.get('postalAddress'));
    }
    if (normalizedProfile.get('addressLocality').isNotNull()) {
        managedUserData.put('city',
normalizedProfile.get('addressLocality'));
    }
    if (normalizedProfile.get('addressRegion').isNotNull()) {
        managedUserData.put('stateProvince',
normalizedProfile.get('addressRegion'));
    }
    if (normalizedProfile.get('postalCode').isNotNull()) {
        managedUserData.put('postalCode',
normalizedProfile.get('postalCode'));
    }
    if (normalizedProfile.get('country').isNotNull()) {
        managedUserData.put('country',
normalizedProfile.get('country'));
    }
    if (normalizedProfile.get('phone').isNotNull()) {
        managedUserData.put('telephoneNumber',
normalizedProfile.get('phone'));
    }

    // if the givenName and familyName is null or empty
    // then add a boolean flag to the shared state to indicate
names are not present
    // this could be used elsewhere
    // for eg. this could be used in a scripted decision node to
by-pass patching
    // the user object with blank values when givenName  and
familyName is not present
     var noGivenName =
normalizedProfile.get('givenName').isNull()
                                        ||
normalizedProfile.get('givenName').asString().trim().length ===
0
     var noFamilyName =
normalizedProfile.get('familyName').isNull()
                                         ||
normalizedProfile.get('familyName').asString().trim().length ===
0
     sharedState.put('nameEmptyOrNull', noGivenName &&
```

```
noFamilyName)

    return managedUserData;
}());
```

Open normalized-profile-to-managed-user.js in your browser.

oauth2-access-token-modification.js

▼ View script

```
/*
 * Copyright 2019-2021 ForgeRock AS. All Rights Reserved.
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script lets you modify information associated with an
OAuth2 access token
 * with methods provided by the AccessToken (1) interface.
 * The changes made to OAuth2 access tokens will directly impact
the size of the CTS tokens,
 * and, similarly, the size of the JWTs if client-based OAuth2
tokens are utilized.
 * When adding/updating fields make sure that the token size
remains within client/user-agent limits.
 *
 * Defined variables:
 * accessToken - AccessToken (1).
 *               The access token to be updated.
 *               Mutable object, all changes to the access token
will be reflected.
 * scopes - Set<String> (6).
 *          Always present, the requested scopes.
 * requestProperties - Unmodifiable Map (5).
 *                     Always present, contains a map of request
properties:
 *                     requestUri - The request URI.
 *                     realm - The realm that the request
relates to.
```

```
 *                     requestParams - A map of the request
params and/or posted data.
 *                                     Each value is a list of
one or more properties.
 *                                     Please note that these
should be handled in accordance with OWASP best practices:
 *                                     https://owasp.org/www-
community/vulnerabilities/Unsafe_use_of_Reflection.
 * clientProperties - Unmodifiable Map (5).
 *                    Present if the client specified in the
request was identified, contains a map of client properties:
 *                    clientId - The client's URI for the
request locale.
 *                    allowedGrantTypes - List of the allowed
grant types (org.forgerock.oauth2.core.GrantType) for the
client.
 *                    allowedResponseTypes - List of the allowed
response types for the client.
 *                    allowedScopes - List of the allowed scopes
for the client.
 *                    customProperties - A map of the custom
properties of the client.
 *                                       Lists or maps will be
included as sub-maps; for example:
 *                                       customMap[Key1]=Value1
will be returned as customMap -> Key1 -> Value1.
 *                                       To add custom
properties to a client, update the Custom Properties field
 *                                       in AM Console > Realm
Name > Applications > OAuth 2.0 > Clients > Client ID >
Advanced.
 * identity - AMIdentity (3).
 *            Always present, the identity of the resource
owner.
 * session - SSOToken (4).
 *           Present if the request contains the session cookie,
the user's session object.
 * scriptName - String (primitive).
 *              Always present, the display name of the script.
 * logger - Always present, the "OAuth2Provider" debug logger
instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 *          Corresponding log files will be prefixed with:
```

```
scripts.OAUTH2_ACCESS_TOKEN_MODIFICATION.
 * httpClient - HTTP Client (8).
 *              Always present, the HTTP Client instance:
 *
https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-http-client.html#scripting-api-
global-http-client.
 *
 * Return - no value is expected, changes shall be made to the
accessToken parameter directly.
 *
 * Class reference:
 * (1) AccessToken -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
oauth2/core/AccessToken.html.
 * (3) AMIdentity -
https://backstage.forgerock.com/docs/am/7/apidocs/com/sun/identi
ty/idm/AMIdentity.html.
 * (4) SSOToken -
https://backstage.forgerock.com/docs/am/7/apidocs/com/iplanet/ss
o/SSOToken.html.
 * (5) Map -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/HashMap.html,
 *           or
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 * (6) Set -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/HashSet.html.
 * (8) Client -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
http/Client.html.
 */

/* EXAMPLE
(function () {
    var frJava = JavaImporter(
        org.forgerock.http.protocol.Request,
        org.forgerock.http.protocol.Response
    );

    // Always includes this field in the token.
    accessToken.setField('key1', 'value1');
```

```
    // Receives and adds to the access token additional values
by performing a REST call to an external service.
    // WARNING: Below, you will find a reference to a third-
party site, which is provided only as an example.
    var uri = 'https://jsonplaceholder.typicode.com/posts';

    try {
        var request = new frJava.Request();

        // You can chain methods that return the request object.
        request.setUri(uri)
            .setMethod('POST')
            .setEntity(JSON.stringify({
                updatedFields: {
                    key2: 'value2',
                    key3: 'value3'
                }
            }));

        // You can call a method when chaining is not possible.
        request.getHeaders().add('Content-Type',
'application/json; charset=UTF-8');

        // Sends the request and receives the response.
        var response = httpClient.send(request).getOrThrow();

        // Checks if the response status is as expected.
        if (response.getStatus() ===
org.forgerock.http.protocol.Status.CREATED) {
            var result =
JSON.parse(response.getEntity().getString());

            // Set multiple token fields at once.
            accessToken.setFields(result.updatedFields);
        } else {
            logger.error('Unable to obtain access token
modifications. Status: ' + response.getStatus() + '. Content: '
+ response.getEntity().getString());
        }
    } catch (e) {
        logger.error('The request processing was interrupted. '
+ e);

        // The access token request fails with the HTTP 500
error in this case.
```

```
        throw ('Unable to obtain response from: ' + uri);
    }

    // Adds new fields containing identity attribute values to
the access token.
    accessToken.setField('mail', identity.getAttribute('mail'));
    accessToken.setField('phone',
identity.getAttribute('telephoneNumber').toArray()[0]);

    // Adds new fields containing the session property values.
    // NOTE: session may not be available for non-interactive
authorization grants.
    if (session) {
        try {
            accessToken.setField('ipAddress',
session.getProperty('Host'));
        } catch (e) {
            logger.error('Unable to retrieve session property
value. ' + e);
        }
    }

    // Removes a native field from the token entry, that was set
by AM.
    // WARNING: removing native fields from the token may result
in loss of functionality.
    // accessToken.removeTokenName()

    // No return value is expected. Let it be undefined.
}());
*/
```

Open oauth2-access-token-modification.js in your browser.

oauth2-authorize-endpoint-data-provider.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
```

```
 */

/*
 * This script lets you return additional data when authorize
request is called.
 *
 * Defined variables:
 *
 * session - SSOToken (1)
 *           Present if the request contains the session cookie,
the user's session object.
 *
 * httpClient - HTTP Client (2).
 *              Always present, the HTTP client that can be used
to make external HTTP requests
 *
 * logger - Debug (3)
 *          Always present, the
"ScriptedAuthorizeEndpointDataProvider" debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 *          Corresponding log files will be prefixed with:
scripts.OAUTH2_AUTHORIZE_ENDPOINT_DATA_PROVIDER.
 *
 * scriptName - String (primitive).
 *              Always present, the display name of the script
 *
 * Return - a Map<String, String> of additional data (4).
 *
 * Class reference:
 * (1) SSOToken -
https://backstage.forgerock.com/docs/am/7/apidocs/com/iplanet/ss
o/SSOToken.html.
 * (2) Client -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
http/Client.html.
 * (3) Debug -
https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * (4) Map -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/HashMap.html.
 */
```

```
/**
 * Default authorize endpoint data provider script to use as a
template for new scripts.
 */

/* EXAMPLE
var map = new java.util.HashMap();

function addAdditionalData() {

    //If constant data needs to be returned
    map.put("hello", "world");

    //If some data needs to be returned from third party service
    addAdditionalDataFromExternalService();

    //If there is a need to return some user session data
    addAdditionalDataFromSessionProperties()

    return map;
};

function addAdditionalDataFromExternalService() {
  var frJava = JavaImporter(
        org.forgerock.oauth2.core.exceptions.ServerException
    );
  try {
        //Obtain additional data by performing a REST call to an
external service
        var request = new org.forgerock.http.protocol.Request();
        request.setUri("https://third.party.app/hello.jsp");
        request.setMethod("POST");
        //request.setEntity("foo=bar&hello=world");
        request.setEntity(json(object(
                    field("foo", "bar"))));
        var response = httpClient.send(request).getOrThrow();
        logResponse(response);
        var result = JSON.parse(response.getEntity());
        map.put("someKey",result.get("someKey"));
  } catch (err) {
     throw new frJava.ServerException(err);
  }
};
```

```
function addAdditionalDataFromSessionProperties() {
  //Add additional data from session property values
   if (session != null) { // session is not available for
resource owner password credentials grant
     map.put("ipAddress", session.getProperty("Host"))
   }
};

function logResponse(response) {
    logger.message("User REST Call. Status: " +
response.getStatus() + ", Body: " + response.getEntity());
};

addAdditionalData();
*/
```

Open oauth2-authorize-endpoint-data-provider.js in your browser.

oauth2-evaluate-scope.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script lets you populate the scopes with profile
attribute values when the tokeninfo endpoint is called.
 * For example, if one of the scopes is mail, AM sets mail to
the resource owner's email address in the token information
returned.
 *
 * Defined variables:
 * accessToken - AccessToken (1).
 *               The access token to be updated.
 *               Mutable object, all changes to the access token
will be reflected.
 * identity - AMIdentity (2).
 *            The client's identity if present or the resource
```

```
owner's identity. Can be null.
 * scriptName - String (primitive).
 *              Always present, the display name of the script.
 * logger - Always present, the debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 *          Corresponding log files will be prefixed with:
scripts.OAUTH2_EVALUATE_SCOPE
 * httpClient - HTTP Client (3).
 *              Always present, the HTTP Client instance:
 *
https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-http-client.html#scripting-api-
global-http-client.
 *
 * Return - a Map<String, Object> of the access token's
information (4).
 *
 * Class reference:
 * (1) AccessToken -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
oauth2/core/AccessToken.html.
 * (2) AMIdentity -
https://backstage.forgerock.com/docs/am/7/apidocs/com/sun/identi
ty/idm/AMIdentity.html.
 * (3) Client -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
http/Client.html.
 * (4) Map -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/HashMap.html.
 */

/**
 * Default evaluate scope script to use as a template for new
scripts.
 */

(function () {
    var map = new java.util.HashMap();
    if (identity !== null) {
        var scopes = accessToken.getScope().toArray();
        scopes.forEach(function (scope) {
            var attributes =
```

```
identity.getAttribute(scope).toArray();
            map.put(scope, attributes.join(","));
        });
    } else {
        logger.error('identity is null');
    }
    return map;
}());
```

Open oauth2-evaluate-scope.js in your browser.

oauth2-may-act.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script lets you add may_act field
 * to an OAuth2 access token
 * or OIDC ID Token
 * object with the setMayAct method.
 *
 * Defined variables:
 * token - AccessToken (1) or
org.forgerock.openidconnect.OpenIdConnectToken.
 *               The token to be updated.
 *               Mutable object, all changes to the token will
be reflected.
 * scopes - Set<String> (6).
 *          Always present, the requested scopes.
 * requestProperties - Unmodifiable Map (5).
 *                     Always present, contains a map of request
properties:
 *                     requestUri - The request URI.
 *                     realm - The realm that the request
relates to.
```

```
 *                     requestParams - A map of the request
params and/or posted data.
 *                                     Each value is a list of
one or more properties.
 *                                     Please note that these
should be handled in accordance with OWASP best practices:
 *                                     https://owasp.org/www-
community/vulnerabilities/Unsafe_use_of_Reflection.
 * clientProperties - Unmodifiable Map (5).
 *                    Present if the client specified in the
request was identified, contains a map of client properties:
 *                    clientId - The client's URI for the
request locale.
 *                    allowedGrantTypes - List of the allowed
grant types (org.forgerock.oauth2.core.GrantType) for the
client.
 *                    allowedResponseTypes - List of the allowed
response types for the client.
 *                    allowedScopes - List of the allowed scopes
for the client.
 *                    customProperties - A map of the custom
properties of the client.
 *                                       Lists or maps will be
included as sub-maps; for example:
 *                                       customMap[Key1]=Value1
will be returned as customMap -> Key1 -> Value1.
 *                                       To add custom
properties to a client, update the Custom Properties field
 *                                       in AM Console > Realm
Name > Applications > OAuth 2.0 > Clients > Client ID >
Advanced.
 * identity - AMIdentity (3).
 *            Always present, the identity of the resource
owner.
 * session - SSOToken (4).
 *           Present if the request contains the session cookie,
the user's session object.
 * scriptName - String (primitive).
 *              Always present, the display name of the script.
 * logger - Always present, the "OAuth2Provider" debug logger
instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 *          Corresponding log files will be prefixed with:
```

```
scripts.OAUTH2_MAY_ACT.
 *
 * Return - no value is expected, changes shall be made to the
token parameter directly.
 *
 * Class reference:
 * (1) AccessToken -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
oauth2/core/AccessToken.html.
 * (3) AMIdentity -
https://backstage.forgerock.com/docs/am/7/apidocs/com/sun/identi
ty/idm/AMIdentity.html.
 * (4) SSOToken -
https://backstage.forgerock.com/docs/am/7/apidocs/com/iplanet/ss
o/SSOToken.html.
 * (5) Map -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/HashMap.html,
 *           or
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 * (6) Set -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/HashSet.html.
 */

/* EXAMPLE
(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var mayAct =
frJava.JsonValue.json(frJava.JsonValue.object());
    mayAct.put('client_id', 'myClient');
    mayAct.put('sub', '(usr!myActor)');

    token.setMayAct(mayAct);

    // No return value is expected. Let it be undefined.
}());
*/
```

Open oauth2-may-act.js in your browser.

oauth2-validate-scope.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script validates the requested scopes against the
allowed scopes.
 * If no scopes are requested, default scopes are assumed.
 * The script has four top level functions that could be
executed during the different OAuth2 flows:
 *      - validateAuthorizationScope
 *      - validateAccessTokenScope
 *      - validateRefreshTokenScope
 *      - validateBackChannelAuthorizationScope
 *
 * Defined variables:
 * requestedScopes - Set<String> (1).
 *          The set of requested scopes.
 * defaultScopes - Set<String> (1).
 *                 The set of default scopes.
 * allowedScopes - Set<String> (1).
 *                 The set of allowed scopes.
 * scriptName - String (primitive).
 *              Always present, the display name of the script.
 * logger - Always present, the debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 *          Corresponding log files will be prefixed with:
scripts.OAUTH2_VALIDATE_SCOPE
 * httpClient - HTTP Client (2).
 *              Always present, the HTTP Client instance:
 *
https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-http-client.html#scripting-api-
global-http-client.
```

```
 *
 * Throws InvalidScopeException:
 *      - if there are no scopes requested and default scopes
are empty
 *      - if a requested scope is not allowed
 *
 * Return - a Set<String> of validated scopes (1).
 *
 * Class reference:
 * (1) Set -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/HashSet.html.
 * (2) Client -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
http/Client.html.
 */

/**
 * Default validate scope script.
 */
function validateScopes () {
    var frJava = JavaImporter(

org.forgerock.oauth2.core.exceptions.InvalidScopeException
    );

    var scopes;
    if (requestedScopes == null || requestedScopes.isEmpty()) {
        scopes = defaultScopes;
    } else {
        scopes = new java.util.HashSet(allowedScopes);
        scopes.retainAll(requestedScopes);
        if (requestedScopes.size() > scopes.size()) {
            var invalidScopes = new
java.util.HashSet(requestedScopes);
            invalidScopes.removeAll(allowedScopes);
            throw new
frJava.InvalidScopeException('Unknown/invalid scope(s)');
        }
    }

    if (scopes == null || scopes.isEmpty()) {
        throw new frJava.InvalidScopeException('No scope
requested and no default scope configured');
    }
```

```
    return scopes;
}

function validateAuthorizationScope () {
    return validateScopes();
}

function validateAccessTokenScope () {
    return validateScopes();
}

function validateRefreshTokenScope () {
    return validateScopes();
}

function validateBackChannelAuthorizationScope () {
    return validateScopes();
}
```

Open oauth2-validate-scope.js in your browser.

oidc-claims-extension.js

▼ View script

```
/*
 * Copyright 2014-2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script computes claim values returned in ID tokens
and/or at the UserInfo Endpoint.
 * The claim values are computed for:
 * the claims derived from the requested scopes,
 * the claims provided by the authorization server,
 * and the claims requested by the client via the claims
parameter.
 *
 * In the CONFIGURATION AND CUSTOMIZATION section, you can
```

```
 * define the scope-to-claims mapping, and
 * assign to each claim a resolver function that will compute
the claim value.
 *
 * Defined variables (class references are provided below):
 * scopes - Set<String> (6).
 *          Always present, the requested scopes.
 * claims - Map<String, Object> (5).
 *          Always present, default server provided claims.
 * claimObjects - List<Claim> (7, 2).
 *                Always present, the default server provided
claims.
 * requestedClaims - Map<String, Set<String>> (5).
 *                   Always present, not empty if the request
contains the claims parameter and the server has enabled
 *                   claims_parameter_supported. A map of the
requested claims to possible values, otherwise empty;
 *                   requested claims with no requested values
will have a key but no value in the map. A key with
 *                   a single value in its Set (6) indicates
that this is the only value that should be returned.
 * requestedTypedClaims - List<Claim> (7, 2).
 *                        Always present, the requested claims.
 *                        Requested claims with no requested
values will have a claim with no values.
 *                        A claim with a single value indicates
this is the only value that should be returned.
 * claimsLocales - List<String> (7).
 *                 The values from the 'claims_locales'
parameter.
 *                 See https://openid.net/specs/openid-connect-
core-1_0.html#ClaimsLanguagesAndScripts for the OIDC
specification details.
 * requestProperties - Unmodifiable Map (5).
 *                     Always present, contains a map of request
properties:
 *                     requestUri - The request URI.
 *                     realm - The realm that the request
relates to.
 *                     requestParams - A map of the request
params and/or posted data.
 *                                     Each value is a list of
one or more properties.
 *                                     Please note that these
should be handled in accordance with OWASP best practices:
```

```
 *                                          https://owasp.org/www-
community/vulnerabilities/Unsafe_use_of_Reflection.
 * clientProperties - Unmodifiable Map (5).
 *                    Present if the client specified in the
request was identified, contains a map of client properties:
 *                    clientId - The client's URI for the
request locale.
 *                    allowedGrantTypes - List of the allowed
grant types (org.forgerock.oauth2.core.GrantType) for the
client.
 *                    allowedResponseTypes - List of the allowed
response types for the client.
 *                    allowedScopes - List of the allowed scopes
for the client.
 *                    customProperties - A map of the custom
properties of the client.
 *                                        Lists or maps will be
included as sub-maps; for example:
 *                                        customMap[Key1]=Value1
will be returned as customMap -> Key1 -> Value1.
 *                                        To add custom
properties to a client, update the Custom Properties field
 *                                        in AM Console > Realm
Name > Applications > OAuth 2.0 > Clients > Client ID >
Advanced.
 * identity - AMIdentity (3).
 *            Always present, the identity of the resource
owner.
 * session - SSOToken (4).
 *           Present if the request contains the session cookie,
the user's session object.
 * scriptName - String (primitive).
 *              Always present, the display name of the script.
 * logger - Always present, the "OAuth2Provider" debug logger
instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 *          Corresponding files will be prefixed with:
scripts.OIDC_CLAIMS.
 * httpClient - HTTP Client (8).
 *              Always present, the HTTP Client instance:
 *
https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-http-client.html#scripting-api-
```

```
global-http-client.
 *              In order to use the client, you may need to add
 *              org.forgerock.http.Client,
 *              org.forgerock.http.protocol.*,
 *              and org.forgerock.util.promise.PromiseImpl
 *              to the allowed Java classes in the scripting
engine configuration, as described in:
 *
https://backstage.forgerock.com/docs/am/7/scripting-
guide/script-engine-security.html
 *
 * Return - a new UserInfoClaims(Map<String, Object> values,
Map<String, List<String>> compositeScopes) (1) object.
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *          See RESULTS section for additional details.
 *
 * Class reference:
 * (1) UserInfoClaims -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
oauth2/core/UserInfoClaims.html.
 * (2) Claim -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
openidconnect/Claim.html).
 *         An instance of org.forgerock.openidconnect.Claim has
methods to access
 *         the claim name, requested values, locale, and whether
the claim is essential.
 * (3) AMIdentity -
https://backstage.forgerock.com/docs/am/7/apidocs/com/sun/identi
ty/idm/AMIdentity.html.
 * (4) SSOToken -
https://backstage.forgerock.com/docs/am/7/apidocs/com/iplanet/ss
o/SSOToken.html.
 * (5) Map -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/HashMap.html,
 *           or
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
```

```
a/util/LinkedHashMap.html.
 * (6) Set -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/HashSet.html.
 * (7) List -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/ArrayList.html.
 * (8) Client -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
http/Client.html.
*/

(function () {
    // SETUP

    /**
     * Claim processing utilities.
     * An object that contains reusable functions for processing
claims.
     * @see CLAIM PROCESSING UTILITIES section for details.
     */
    var utils = getUtils();

    // CONFIGURATION AND CUSTOMIZATION

    /**
     * OAuth 2.0 scope values (scopes) can be used by the Client
to request OIDC claims.
     *
     * Call this configuration method, and pass in as the first
argument
     * an object that maps a scope value to an array of claim
names
     * to specify which claims need to be processed and returned
for the requested scopes.
     * @see {@link https://openid.net/specs/openid-connect-core-
1_0.html#ScopeClaims}
     * for the scope values that could be used to request claims
as defined in the OIDC specification.
     *
     * Below, find a default configuration that is expected to
work in the current environment.
     *
     * CUSTOMIZATION
     * You can choose the claim names returned for a scope.
```

```
     */
    utils.setScopeClaimsMap({
        profile: [
            'name',
            'family_name',
            'given_name',
            'zoneinfo',
            'locale'
        ],
        email: ['email'],
        address: ['address'],
        phone: ['phone_number']
    });

    /**
     * In this script, each claim
     * derived from the requested scopes,
     * provided by the authorization server, and
     * requested by the client via the claims parameter
     * will be processed by a function associated with the claim
name.
     *
     * Call this configuration method, and pass in as the first
argument
     * an object that maps a claim name to a resolver function,
     * which will be automatically executed for each claim
processed by the script.
     *
     * The claim resolver function will receive the requested
claim information
     * in an instance of org.forgerock.openidconnect.Claim as
the first argument.
     * @see {@link
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
openidconnect/Claim.html}
     * for details on the Claim class.
     *
     * If the claim resolver function returns a value,
     * other than undefined or null,
     * the claim will be included in the script's results.
     *
     * The Claim instance provides methods to check
     * what the name of the claim is,
     * which values the claim request contains,
     * whether the claim is essential, and
```

```
     * which locale the claim is associated with.
     * The resolver function can consider this information when
computing and returning the claim value.
     *
     * Below, find a default configuration that is expected to
work in the current environment.
     * A reusable function,
utils.getUserProfileClaimResolver(String attribute-name),
     * is called to return a claim resolver function based on a
user profile attribute.
     * @see CLAIM RESOLVERS section for the implementation
details and examples.
     * For the address claim, an example of a claim resolver
that uses another claim resolver is provided.
     *
     * CUSTOMIZATION
     * You can reuse the predefined utils methods with your
custom arguments.
     * You can also specify a custom resolver function for a
claim name,
     * that will compute and return the claim value—as shown in
the commented out example below.
     */
    utils.setClaimResolvers({
        /*
        // An example of a simple claim resolver function that
is defined for a claim
        // directly in the configuration object:
        custom-claim-name: function (requestedClaim) {
            // In this case, initially, the claim value comes
straight from a user profile attribute value:
            var claimValue = identity.getAttribute('custom-
attribute-name').toArray()[0]

            // Optionally, provide additional logic for
processing (filtering, formatting, etc.) the claim value.
            // You can use:
            // requestedClaim.getName()
            // requestedClaim.getValues()
            // requestedClaim.getLocale()
            // requestedClaim.isEssential()

            return claimValue
        },
        */
```

```
        /**
         * The use of utils.getUserProfileClaimResolver shows
how
         * an argument passed to a function that returns a claim
resolver
         * becomes available to the resolver function (via its
lexical context).
         */
        name: utils.getUserProfileClaimResolver('cn'),
        family_name: utils.getUserProfileClaimResolver('sn'),
        given_name:
utils.getUserProfileClaimResolver('givenname'),
        zoneinfo:
utils.getUserProfileClaimResolver('preferredtimezone'),
        locale:
utils.getUserProfileClaimResolver('preferredlocale'),
        email: utils.getUserProfileClaimResolver('mail'),
        address: utils.getAddressClaimResolver(
            /**
             * The passed in user profile claim resolver
function
             * can be used by the address claim resolver
function
             * to obtain the claim value to be formatted as per
the OIDC specification:
             * @see https://openid.net/specs/openid-connect-
core-1_0.html#AddressClaim.
             */
            utils.getUserProfileClaimResolver('postaladdress')
        ),
        phone_number:
utils.getUserProfileClaimResolver('telephonenumber')
    });

    // CLAIM PROCESSING UTILITIES

    /**
     * @returns {object} An object that contains reusable claim
processing utilities.
     * @see PUBLIC METHODS section and the return statement for
the list of exported functions.
     */
    function getUtils () {
        // IMPORT JAVA
```

```
        /**
         * Provides Java scripting functionality.
         * @see {@link https://developer.mozilla.org/en-
US/docs/Mozilla/Projects/Rhino/Scripting_Java#javaimporter_const
ructor}.
         */
        var frJava = JavaImporter(

org.forgerock.oauth2.core.exceptions.InvalidRequestException,
            org.forgerock.oauth2.core.UserInfoClaims,
            org.forgerock.openidconnect.Claim,

            java.util.LinkedHashMap,
            java.util.ArrayList
        );

        // SET UP CONFIGURATION

        /**
         * Placeholder for a configuration option that contains
         * an object that maps the supported scope values
(scopes)
         * and the corresponding claim names for each scope
value.
         */
        var scopeClaimsMap;

        /**
         * Placeholder for a configuration option that contains
         * an object that maps the supported claim names
         * and the resolver functions returning the claim value.
         */
        var claimResolvers;

        /**
         * A (public) method that accepts an object that maps
the supported scopes and the corresponding claim names,
         * and assigns it to a (private) variable that serves as
a configuration option.
         * @param {object} params - An object that maps each
supported scope value to an array of claim names,
         * in order to specify which claims need to be processed
for the requested scopes.
         * @see {@link https://openid.net/specs/openid-connect-
core-1_0.html#ScopeClaims} for details.
```

```
         * @param {string[]} [params.profile] - An array of
claim names to be returned if the profile scope is requested.
         * @param {string[]} [params.email] - An array of claim
names to be returned if the email scope is requested.
         * @param {string[]} [params.address] - An array of
claim names to be returned if the address scope is requested.
         * @param {string[]} [params.phone] - An array of claim
names to be returned if the phone scope is requested.
         * @returns {undefined}
         */
        function setScopeClaimsMap(params) {
            scopeClaimsMap = params;
        }

        /**
         * A (public) method that accepts an object that maps
the supported claim names
         * and the resolver functions returning the claim value,
         * and assigns it to a (private) variable that serves as
a configuration option.
         * @param {object} params - An object that maps
         * each supported claim name to a function that computes
and returns the claim value.
         */
        function setClaimResolvers(params) {
            claimResolvers = params;
        }

        // CLAIM RESOLVERS

        /**
         * Claim resolvers are functions that return a claim
value.
         * @param {*}
         * @returns {*}
         */

        /**
         * Defines a claim resolver based on a user profile
attribute.
         * @param {string} attributeName - Name of the user
profile attribute.
         * @returns {function} A function that will determine
the claim value
         * based on the user profile attribute and the
```

```
(requested) claim properties.
         */
        function getUserProfileClaimResolver (attributeName) {
            /**
             * Resolves a claim with a user profile attribute
value.
             * Returns undefined if the identity attribute is
not populated,
             * OR if the claim has requested values that do not
contain the identity attribute value.
             * ATTENTION: the aforementioned comparison is case-
sensitive.
             * @param {org.forgerock.openidconnect.Claim} claim
             * An object that provides methods to obtain
information/requirements associated with a claim.
             * @see {@link
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
openidconnect/Claim.html} for details.
             * @returns {string|HashSet|undefined}
             */
            function resolveClaim(claim) {
                var userProfileValue;

                if (identity) {
                    userProfileValue =
getClaimValueFromSet(claim,
identity.getAttribute(attributeName));

                    if (userProfileValue &&
!userProfileValue.isEmpty()) {
                        if (!claim.getValues() ||
claim.getValues().isEmpty() ||
claim.getValues().contains(userProfileValue)) {
                            return userProfileValue;
                        }
                    }
                }
            }

            return resolveClaim;
        }

        /**
         * Returns an address claim resolver based on a claim
value obtained with another claim resolver.
```

```
         * @param {function} resolveClaim - A function that
returns a claim value.
         * @returns {function} A function that will accept a
claim as an argument,
         * run the claim resolver function for the claim and
obtain the claim value,
         * and apply additional formatting to the value before
returning it.
         */
        function getAddressClaimResolver (resolveClaim) {
            /**
             * Creates an address claim object from a value
returned by a claim resolver,
             * and returns the address claim object as the claim
value.
             * @see {@link https://openid.net/specs/openid-
connect-core-1_0.html#AddressClaim}.
             * The claim value is obtained with a claim
resolving function available from the closure.
             * @param {org.forgerock.openidconnect.Claim} claim
             * An object that provides methods to obtain
information/requirements associated with a claim.
             * @see {@link
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
openidconnect/Claim.html} for details.
             * @returns {java.util.LinkedHashMap|undefined} The
address claim object created from a claim value.
             */
            function resolveAddressClaim(claim) {
                var claimValue = resolveClaim(claim);
                var addressObject;

                if (isClaimValueValid(claimValue)) {
                    addressObject = new frJava.LinkedHashMap();

                    addressObject.put('formatted', claimValue);

                    return addressObject;
                }
            }

            return resolveAddressClaim;
        }

        /**
```

```
         * Returns an essential claim resolver based on a claim
value obtained with another claim resolver.
         * @param {function} resolveClaim - A function that
returns a claim value.
         * @returns {function} A function that will accept a
claim as an argument,
         * run the claim resolver function for the claim and
obtain the claim value,
         * and apply additional logic for essential claims.
         */
        function getEssentialClaimResolver (resolveClaim) {
            /**
             * Returns a claim value or throws an error.
             * The claim value is obtained with a claim
resolving function available from the closure.
             * Throws an exception if the claim is essential and
no value is returned for the claim.
             *
             * Use of this resolver is optional.
             * @see {@link https://openid.net/specs/openid-
connect-core-1_0.html#IndividualClaimsRequests} stating:
             * "Note that even if the Claims are not available
because the End-User did not authorize their release or they are
not present,
             * the Authorization Server MUST NOT generate an
error when Claims are not returned, whether they are Essential
or Voluntary,
             * unless otherwise specified in the description of
the specific claim."
             *
             * @param {org.forgerock.openidconnect.Claim} claim
             * An object that provides methods to obtain
information/requirements associated with a claim.
             * @see {@link
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
openidconnect/Claim.html} for details.
             * @returns {*}
             * @throws
{org.forgerock.oauth2.core.exceptions.InvalidRequestException}
             */
            function resolveEssentialClaim(claim) {
                var claimValue = resolveClaim(claim);

                if (claim.isEssential() &&
!isClaimValueValid(claimValue)) {
```

```
                    throw new
frJava.InvalidRequestException('Could not provide value for
essential claim: ' + claim.getName());
                }

                return claimValue;
            }

            return resolveEssentialClaim;
        }

        /**
         * Provides default resolution for a claim.
         * Use it if a claim-specific resolver is not defined in
the configuration.
         * @param {org.forgerock.openidconnect.Claim} claim
         * An object that provides methods to obtain
information/requirements associated with a claim.
         * @see {@link
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
openidconnect/Claim.html} for details.
         * @returns {*} A single value associated with this
claim.
         */
        function resolveAnyClaim (claim) {
            if (claim.getValues().size() === 1) {
                return claim.getValues().toArray()[0];
            }
        }

        // UTILITIES

        /**
         * Returns claim value from a set.
         * If the set contains a single value, returns the
value.
         * If the set contains multiple values, returns the set.
         * Otherwise, returns undefined.
         *
         * @param {org.forgerock.openidconnect.Claim} claim
         * An object that provides methods to obtain
information/requirements associated with a claim.
         * @see {@link
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
openidconnect/Claim.html} for details.
```

```
         * @param {java.util.HashSet} set The set-for example, a
user profile attribute value.
         * @returns {string|java.util.HashSet|undefined}
         */
        function getClaimValueFromSet (claim, set) {
            if (set && set.size()) {
                if (set.size() === 1) {
                    return set.toArray()[0];
                } else {
                    return set;
                }
            } else if (logger.warningEnabled()) {
                logger.warning('OIDC Claims script. Got an empty
set for claim: ' + claim.getName());
            }
        }

        function isClaimValueValid (claimValue) {
            if (typeof claimValue === 'undefined' || claimValue
=== null) {
                return false;
            }

            return true;
        }

        // CLAIM PROCESSING

        /**
         * Constructs and returns an object populated with the
computed claim values
         * and the requested scopes mapped to the claim names.
         * @returns {org.forgerock.oauth2.core.UserInfoClaims}
The object to be returned to the authorization server.
         * @see {@link
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
oauth2/core/UserInfoClaims.html}.
         * @see RESULTS section for the use of this function.
         */
        function getUserInfoClaims () {
            return new
frJava.UserInfoClaims(getComputedClaims(),
getCompositeScopes());
        }
```

```
        /**
         * Creates a map of (requested) claim names populated
with the computed claim values.
         * @returns {java.util.LinkedHashMap}
         * A map of the requested claim names and the
corresponding claim values.
         */
        function getComputedClaims () {
            /**
             * Creates a complete list of claim objects from:
             * the claims derived from the scopes,
             * the claims provided by the authorization server,
             * and the claims requested by the client.
             * @returns {java.util.ArrayList}
             * Returns a complete list of
org.forgerock.openidconnect.Claim objects available to the
script.
             * @see {@link
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
openidconnect/Claim.html} for the claim object details.
             */
            function getClaims() {
                /**
                 * Returns a list of claim objects for the
requested scopes.
                 * Uses the scopeClaimsMap configuration option
to derive the claim names;
                 * no other properties of a claim derived from a
scope are populated.
                 * @returns {java.util.ArrayList}
                 * A list of org.forgerock.openidconnect.Claim
objects derived from the requested scopes.
                 * @see {@link
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
openidconnect/Claim.html} for the claim object details.
                 */
                function convertScopeToClaims() {
                    var claims = new frJava.ArrayList();

                    scopes.toArray().forEach(function (scope) {
                        if (String(scope) !== 'openid' &&
scopeClaimsMap[scope]) {

scopeClaimsMap[scope].forEach(function (claimName) {
                                claims.add(new
```

```
frJava.Claim(claimName));
                            });
                        }
                    });

                    return claims;
                }

                var claims = new frJava.ArrayList();

                claims.addAll(convertScopeToClaims());
                claims.addAll(claimObjects);
                claims.addAll(requestedTypedClaims);

                return claims;
            }

            /**
             * Computes and returns a claim value.
             * To obtain the claim value, uses the resolver
function specified for the claim in the claimResolvers
configuration object.
             * @see claimResolvers
             * If no resolver function is found, uses the
default claim resolver function.
             *
             * @param {org.forgerock.openidconnect.Claim} claim
             * An object that provides methods to obtain
information/requirements associated with a claim.
             * @see {@link
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
openidconnect/Claim.html} for details.
             * @returns {*} Claim value.
             * @throws
{org.forgerock.oauth2.core.exceptions.InvalidRequestException}
             * Rethrows this exception if a claim resolver
throws it.
             * You can throw
org.forgerock.oauth2.core.exceptions.InvalidRequestException
from your custom claim resolver
             * if you want to terminate the claim processing.
             */
            function computeClaim(claim) {
                var resolveClaim;
                var message;
```

```
                try {
                    resolveClaim =
claimResolvers[claim.getName()] || resolveAnyClaim;

                    return resolveClaim(claim);
                } catch (e) {
                    message = 'OIDC Claims script exception.
Unable to resolve OIDC Claim. ' + e;

                    if
(String(e).indexOf('org.forgerock.oauth2.core.exceptions.Invalid
RequestException') !== -1) {
                        throw e;
                    }

                    if (logger.warningEnabled()) {
                        logger.warning(message);
                    }
                }
            }

            var computedClaims = new frJava.LinkedHashMap();

            getClaims().toArray().forEach(function (claim) {
                var claimValue = computeClaim(claim);

                if (isClaimValueValid(claimValue)) {
                    computedClaims.put(claim.getName(),
claimValue);
                } else {
                    /**
                     * If a claim has been processed, but
appears in the list again,
                     * and its value cannot be computed under
the new conditions,
                     * the claim is removed from the final
result.
                     *
                     * For example, a claim could be mapped to a
scope and found in the user profile,
                     * but also requested by the client with
required values that don't match the computed one.
                     * @see {link
https://openid.net/specs/openid-connect-core-
```

```
1_0.html#IndividualClaimsRequests}.
                     * for the relevant OIDC specification
details.
                     */
                    computedClaims.remove(claim.getName());
                }
            });

            return computedClaims;
        }

        /**
         * Creates a map of requested scopes and the
corresponding claim names.
         * @returns {java.util.LinkedHashMap}
         */
        function getCompositeScopes () {
            var compositeScopes = new frJava.LinkedHashMap();

            scopes.toArray().forEach(function (scope) {
                var scopeClaims = new frJava.ArrayList();

                if (scopeClaimsMap[scope]) {
                    scopeClaimsMap[scope].forEach(function
(claimName) {
                        scopeClaims.add(claimName);
                    });
                }

                if (scopeClaims.size()) {
                    compositeScopes.put(scope, scopeClaims);
                }
            });

            return compositeScopes;
        }

        // PUBLIC METHODS

        return {
            setScopeClaimsMap: setScopeClaimsMap,
            setClaimResolvers: setClaimResolvers,
            getUserProfileClaimResolver:
getUserProfileClaimResolver,
            getAddressClaimResolver: getAddressClaimResolver,
```

```
            getEssentialClaimResolver:
getEssentialClaimResolver,
            getUserInfoClaims: getUserInfoClaims
        };
    }

    // RESULTS

    /**
     * This script returns an instance of the
org.forgerock.oauth2.core.UserInfoClaims class
     * populated with the computed claim values and
     * the requested scopes mapped to the claim names.
     * @see {@link
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
oauth2/core/UserInfoClaims.html}.
     *
     * Assigning it to a variable gives you an opportunity
     * to log the content of the returned value during
development.
     */
    var userInfoClaims = utils.getUserInfoClaims();

    /*
    logger.error(scriptName + ' results:')
    logger.error('Values: ' + userInfoClaims.getValues())
    logger.error('Scopes: ' +
userInfoClaims.getCompositeScopes())
    */

    return userInfoClaims;
}());
```

Open oidc-claims-extension.js in your browser.

policy-condition.js

▼ View script

```
/*
 * Copyright 2015-2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
```

```
subject
 * to such license between the licensee and ForgeRock AS.
 */
/**
 * This is a Policy Condition example script. It demonstrates
how to access a user's information,
 * use that information in external HTTP calls and make a policy
decision based on the outcome.
 */

var userAddress, userIP, resourceHost;

if (validateAndInitializeParameters()) {

    var countryFromUserAddress = getCountryFromUserAddress();
    logger.message("Country retrieved from user's address: " +
countryFromUserAddress);
    var countryFromUserIP = getCountryFromUserIP();
    logger.message("Country retrieved from user's IP: " +
countryFromUserIP);
    var countryFromResourceURI = getCountryFromResourceURI();
    logger.message("Country retrieved from resource URI: " +
countryFromResourceURI);

    if (countryFromUserAddress === countryFromUserIP &&
countryFromUserAddress === countryFromResourceURI) {
        logger.message("Authorization Succeeded");
        responseAttributes.put("countryOfOrigin",
[countryFromUserAddress]);
        authorized = true;
    } else {
        logger.message("Authorization Failed");
        authorized = false;
    }

} else {
    logger.message("Required parameters not found. Authorization
Failed.");
    authorized = false;
}

/**
 * Use the user's address to lookup their country of residence.
 *
 * @returns {*} The user's country of residence.
```

```
 */
function getCountryFromUserAddress() {

    var request = new org.forgerock.http.protocol.Request();

request.setUri("http://maps.googleapis.com/maps/api/geocode/json
?address=" + encodeURIComponent(userAddress));
        request.setMethod("GET");

    var response = httpClient.send(request).get();
    logResponse(response);

    var geocode = JSON.parse(response.getEntity());
    var i;
    for (i = 0; i < geocode.results.length; i++) {
        var result = geocode.results[i];
        var j;
        for (j = 0; j < result.address_components.length; i++) {
            if (result.address_components[i].types[0] ==
"country") {
                return result.address_components[i].long_name;
            }
        }
    }
}

/**
 * Use the user's IP to lookup the country from which the
request originated.
 *
 * @returns {*} The country from which the request originated.
 */
function getCountryFromUserIP() {
    var request = new org.forgerock.http.protocol.Request();
    request.setUri("http://ip-api.com/json/" + userIP);
        request.setMethod("GET");

    var response = httpClient.send(request).get();
    logResponse(response);

    var result = JSON.parse(response.getEntity());
    if (result) {
        return result.country;
    }
}
```

```
/**
 * Use the requested resource's host name to lookup the country
where the resource is hosted.
 *
 * @returns {*} The country in which the resource is hosted.
 */
function getCountryFromResourceURI() {
    var request = new org.forgerock.http.protocol.Request();
    request.setUri("http://ip-api.com/json/" +
encodeURIComponent(resourceHost));
        request.setMethod("GET");

    var response = httpClient.send(request).get();
    logResponse(response);

    var result = JSON.parse(response.getEntity());
    if (result) {
        return result.country;
    }
}

/**
 * Retrieve and validate the variables required to make the
external HTTP calls.
 *
 * @returns {boolean} Will be true if validation was successful.
 */
function validateAndInitializeParameters() {
    var userAddressSet = identity.getAttribute("postalAddress");
    if (userAddressSet == null || userAddressSet.isEmpty()) {
        logger.warning("No address specified for user: " +
username);
        return false;
    }
    userAddress = userAddressSet.iterator().next();
    logger.message("User address: " + userAddress);

    if (!environment) {
        logger.warning("No environment parameters specified in
the evaluation request.");
        return false;
    }

    var ipSet = environment.get("IP");
```

```
    if (ipSet == null || ipSet.isEmpty()) {
        logger.warning("No IP specified in the evaluation
request environment parameters.");
        return false;
    }
    userIP = ipSet.iterator().next();
    logger.message("User IP: " + userIP);

    if (!resourceURI) {
        logger.warning("No resource URI specified.");
        return false;
    }
    resourceHost = resourceURI.match(/^(.*:\/\/)(www\.)?([A-Za-
z0-9\-\.]+)(:[0-9]+)?(.*)$/)[3];
    logger.message("Resource host: " + resourceHost);

    return true;
}

function logResponse(response) {
    logger.message("User REST Call. Status: " +
response.getStatus() + ", Body: " + response.getEntity());
}
```

Open policy-condition.js in your browser.

saml2-idp-adapter.js

▼ View script

```
/*
 * Copyright 2021-2022 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * The script has these top level functions that could be
executed during a SAML2 flow.
 *      - preSingleSignOn
 *      - preAuthentication
```

```
 *      - preSendResponse
 *      - preSignResponse
 *      - preSendFailureResponse
 *
 * Please see the javadoc for the interface definition and more
information about these methods.
 *
https://backstage.forgerock.com/docs/am/7.2/apidocs/com/sun/iden
tity/saml2/plugins/SAML2IdentityProviderAdapter.html
 * Note that the initialize method is not supported in the
scripts.
 *
 * Defined variables. Check the documentation on the respective
functions for the variables available to it.
 *
 * hostedEntityId - String
 *     Entity ID for the hosted IDP
 * realm - String
 *     Realm of the hosted IDP
 * idpAdapterScriptHelper - IdpAdapterScriptHelper (1)
 *     An instance of IdpAdapterScriptHelper containing helper
methods. See Javadoc for more details.
 * request - HttpServletRequest (2)
 *     Servlet request object
 * response - HttpServletResponse (3)
 *     Servlet response object
 * authnRequest - AuthnRequest (4)
 *     The original authentication request sent from SP
 * reqId - String
 *     The id to use for continuation of processing if the
adapter redirects
 * res - Response (5)
 *     The SAML Response
 * session - SSOToken (6)
 *     The single sign-on session. The reference type of this is
Object and would need to be casted to SSOToken.
 * relayState - String
 *     The relayState that will be used in the redirect
 * faultCode - String
 *     the fault code that will be returned in the SAML response
 * faultDetail - String
 *     the fault detail that will be returned in the SAML
response
 * logger - Logger instance
 *     https://backstage.forgerock.com/docs/am/7/scripting-
```

```
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 *     Corresponding log files will be prefixed with: scripts.
<script name>
 *
 * Throws SAML2Exception (7):
 *     for any exceptions occurring in the adapter. The
federation process will continue
 *
 * Class reference:
 * (1) idpAdapterScriptHelper -
https://backstage.forgerock.com/docs/am/7.2/apidocs/com/sun/iden
tity/saml2/plugins/scripted/IdpAdapterScriptHelper.html.
 * (2) HttpServletRequest - https://tomcat.apache.org/tomcat-
7.0-doc/servletapi/javax/servlet/http/HttpServletRequest.html.
 * (3) HttpServletResponse - https://tomcat.apache.org/tomcat-
7.0-doc/servletapi/javax/servlet/http/HttpServletResponse.html.
 * (4) AuthnRequest -
https://backstage.forgerock.com/docs/am/7.2/apidocs/com/sun/iden
tity/saml2/protocol/AuthnRequest.html.
 * (5) Response -
https://backstage.forgerock.com/docs/am/7.2/apidocs/com/sun/iden
tity/saml2/protocol/Response.html.
 * (6) SSOToken -
https://backstage.forgerock.com/docs/am/7.2/apidocs/com/iplanet/
sso/SSOToken.html.
 * (7) SAML2Exception -
https://backstage.forgerock.com/docs/am/7.2/apidocs/com/sun/iden
tity/saml2/common/SAML2Exception.html.
 */

/*
 * Template/default script for SAML2 IDP Adapter scripted
plugin.
 */

/*
 * Available variables for preSingleSignOn:
 *     hostedEntityId
 *     realm
 *     idpAdapterScriptHelper
 *     request
 *     authnRequest
 *     response
 *     reqId
```

```
 *     logger
 *
 * Return - true if browser redirection is happening after
processing, false otherwise. Default to false.
 */
function preSingleSignOn () {
    return false;
}

/*
 * Available variables for preAuthentication:
 *     hostedEntityId
 *     realm
 *     idpAdapterScriptHelper
 *     request
 *     authnRequest
 *     response
 *     reqId
 *     session
 *     relayState
 *     logger
 *
 * Return - true if browser redirection is happening after
processing, false otherwise. Default to false.
 */
function preAuthentication () {
    return false;
}

/*
 * Available variables for preSendResponse:
 *     hostedEntityId
 *     realm
 *     idpAdapterScriptHelper
 *     request
 *     authnRequest
 *     response
 *     reqId
 *     session
 *     relayState
 *     logger
 *
 * Return - true if browser redirection happened after
processing, false otherwise. Default to false.
 */
```

```
function preSendResponse () {
    return false;
}

/*
 * Available variables for preSignResponse:
 *     hostedEntityId
 *     realm
 *     idpAdapterScriptHelper
 *     request
 *     authnRequest
 *     session
 *     relayState
 *     res
 *     logger
 */
function preSignResponse () {
}

/*
 * Available variables for preSendFailureResponse:
 *     hostedEntityId
 *     realm
 *     idpAdapterScriptHelper
 *     request
 *     response
 *     faultCode
 *     faultDetail
 *     logger
 */
function preSendFailureResponse () {
}
```

Open saml2-idp-adapter.js in your browser.

saml2-idp-attribute-mapper.js

▼ View script

```
/*
 * Copyright 2021-2022 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively
```

```
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns a list of SAML Attribute objects for the
IDP framework to insert into the generated Assertion.
 *
 * Defined variables:
 * session - SSOToken (1)
 *           The single sign-on session.
 * hostedEntityId - String (primitive).
 *                  The hosted entity ID.
 * remoteEntityId - String (primitive).
 *                  The remote entity ID.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * logger - Always present, the debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 *          Corresponding log files will be prefixed with:
scripts.SAML2_IDP_ATTRIBUTE_MAPPER
 * idpAttributeMapperScriptHelper -
IdpAttributeMapperScriptHelper (2)
 *                                - An
IdpAttributeMapperScriptHelper instance containing methods used
for IDP attribute mapping.
 *
 * Throws SAML2Exception:
 *      - on failing to map the IDP attributes.
 *
 * Return - a list of SAML Attribute (3) objects.
 *
 * Class reference:
 * (1) SSOToken -
https://backstage.forgerock.com/docs/am/7/apidocs/com/iplanet/ss
o/SSOToken.html.
 * (2) IdpAttributeMapperScriptHelper -
https://backstage.forgerock.com/docs/am/7.2/apidocs/com/sun/iden
tity/saml2/plugins/scripted/IdpAttributeMapperScriptHelper.html.
 * (3) Attribute -
https://backstage.forgerock.com/docs/am/7/apidocs/com/sun/identi
ty/saml2/assertion/Attribute.html.
 */
```

```
/**
 * Default SAML2 IDP Attribute Mapper.
 */
function getAttributes() {
    var frJava = JavaImporter(
        com.sun.identity.saml2.common.SAML2Exception
    );

    const debugMethod =
"ScriptedIDPAttributeMapper.getAttributes:: ";

    try {

        if
(!idpAttributeMapperScriptHelper.isSessionValid(session)) {
            logger.error(debugMethod + "Invalid session.");
            return null;
        }

        var configMap =
idpAttributeMapperScriptHelper.getRemoteSPConfigAttributeMap(rea
lm, remoteEntityId);
        logger.message(debugMethod + "Remote SP attribute map =
{}", configMap);
        if (configMap == null || configMap.isEmpty()) {
            configMap =
idpAttributeMapperScriptHelper.getHostedIDPConfigAttributeMap(re
alm, hostedEntityId);
            if (configMap == null || configMap.isEmpty()) {
                logger.message(debugMethod + "Configuration map
is not defined.");
                return null;
            }
            logger.message(debugMethod + "Hosted IDP attribute
map = {}", configMap);
        }

        var attributes = new java.util.ArrayList();
        var stringValueMap = new java.util.HashSet();
        var binaryValueMap;
        var localAttribute;

        // Don't try to read the attributes from the datastore
if the ignored profile is enabled in this realm.
```

```
        if
(!idpAttributeMapperScriptHelper.isIgnoredProfile(session,
realm)) {
            try {
                // Resolve attributes to be read from the
datastore.
                var stringAttributes = new java.util.HashSet();
                var binaryAttributes = new java.util.HashSet();
                var keyIter = configMap.keySet().iterator();
                while (keyIter.hasNext()) {
                    var key = keyIter.next();
                    localAttribute = configMap.get(key);
                    if
(!idpAttributeMapperScriptHelper.isStaticAttribute(localAttribut
e)) {
                        if
(idpAttributeMapperScriptHelper.isBinaryAttribute(localAttribute
)) {
                            // add it to the list of attributes
to treat as being binary

binaryAttributes.add(idpAttributeMapperScriptHelper.removeBinary
AttributeFlag(localAttribute));
                        } else {

stringAttributes.add(localAttribute);
                        }
                    }
                }

                if (!stringAttributes.isEmpty()) {
                    stringValueMap =
idpAttributeMapperScriptHelper.getAttributes(session,
stringAttributes);
                }
                if (!binaryAttributes.isEmpty()) {
                    binaryValueMap =
idpAttributeMapperScriptHelper.getBinaryAttributes(session,
binaryAttributes);
                }
            } catch (error) {
                logger.error(debugMethod + "Error accessing the
datastore. " + error);
                //continue to check in ssotoken.
            }
```

```
        }

        var keyIter = configMap.keySet().iterator();
        while (keyIter.hasNext()) {
            var key = keyIter.next()
            var nameFormat = null;
            var samlAttribute = key;
            localAttribute = configMap.get(key);
            // check if samlAttribute has format
nameFormat|samlAttribute
            var samlAttributes = String(new
java.lang.String(samlAttribute));
            var tokens = samlAttributes.split('|');

            if (tokens.length > 1) {
                nameFormat = tokens[0];
                samlAttribute = tokens[1];
            }

            var attributeValues = new java.util.HashSet();
            if
(idpAttributeMapperScriptHelper.isStaticAttribute(localAttribute
)) {
                // Remove the static flag before using it as the
static value
                localAttribute =
idpAttributeMapperScriptHelper.removeStaticAttributeFlag(localAt
tribute);
                attributeValues = new
java.util.HashSet([localAttribute]);
                logger.message(debugMethod + "Adding static
value {} for attribute named {}", localAttribute,
samlAttribute);
            } else {
                if
(idpAttributeMapperScriptHelper.isBinaryAttribute(localAttribute
)) {
                    // Remove the flag as not used for lookup
                    localAttribute =
idpAttributeMapperScriptHelper.removeBinaryAttributeFlag(localAt
tribute);
                    attributeValues =
idpAttributeMapperScriptHelper.getBinaryAttributeValues(samlAttr
ibute, localAttribute,
                        binaryValueMap);
```

```
                } else {
                    if (stringValueMap != null &&
!stringValueMap.isEmpty()) {
                        attributeValues =
stringValueMap.get(localAttribute);
                    } else {
                        logger.message(debugMethod + "{} string
value map was empty or null.", localAttribute);
                    }
                }

                // If all else fails, try to get the value from
the users ssoToken
                if (attributeValues == null ||
attributeValues.isEmpty()) {
                    logger.message(debugMethod + "User profile
does not have value for {}, checking SSOToken.",
localAttribute);
                    attributeValues = new
java.util.HashSet(idpAttributeMapperScriptHelper.getPropertySet(
session, localAttribute));
                }
            }

            if (attributeValues == null ||
attributeValues.isEmpty()) {
                logger.message(debugMethod + "{} not found in
user profile or SSOToken.", localAttribute);
            } else {

attributes.add(idpAttributeMapperScriptHelper.createSAMLAttribut
e(samlAttribute, nameFormat, attributeValues));
            }
        }

        return attributes;

    } catch (error) {
        logger.error(debugMethod + "Error mapping IDP
attributes. " + error);
        throw new frJava.SAML2Exception(error);
    }
}

getAttributes();
```

Open saml2-idp-attribute-mapper.js in your browser.

salesforce-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
```

```
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
```

```
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('user_id'));
    normalizedProfileData.put('displayName',
rawProfile.get('name'));
    normalizedProfileData.put('givenName',
rawProfile.get('given_name'));
    normalizedProfileData.put('familyName',
rawProfile.get('family_name'));
    normalizedProfileData.put('photoUrl',
rawProfile.get('picture'));
    normalizedProfileData.put('email', rawProfile.get('email'));
    normalizedProfileData.put('username',
rawProfile.get('email'));
    normalizedProfileData.put('locale',
rawProfile.get('zoneInfo'));
```

```
    return normalizedProfileData;
}());
```

Open salesforce-profile-normalization.js in your browser.

social-idp-profile-transformation.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - JsonValue (1).
 *              The social identity provider profile information
for the authenticating user.
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
```

```
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
```

```
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

/**
 * Default Social Identity Provider Profile Transformation
script to use as a template for new scripts.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    /**
     * Add profile data.
     * @example
     * normalizedProfileData.put('id', rawProfile.get('sub'));
     */
```

```
    return normalizedProfileData;
}());
```

Open social-idp-profile-transformation.js in your browser.

twitter-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
```

```
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
```

```
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('id_str'));
    normalizedProfileData.put('displayName',
rawProfile.get('name'));
    normalizedProfileData.put('photoUrl',
rawProfile.get('profile_image_url'));
    normalizedProfileData.put('email', rawProfile.get('email'));
    normalizedProfileData.put('username',
rawProfile.get('screen_name'));

    return normalizedProfileData;
}());
```

Open twitter-profile-normalization.js in your browser.

vkontakte-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
```

```
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
```

```
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('id'));
    normalizedProfileData.put('displayName',
rawProfile.get('first_name'));
    normalizedProfileData.put('givenName',
rawProfile.get('first_name'));
    normalizedProfileData.put('familyName',
rawProfile.get('last_name'));
    normalizedProfileData.put('photoUrl',
rawProfile.get('photo_50'));
    normalizedProfileData.put('email', rawProfile.get('email'));
    normalizedProfileData.put('username',
rawProfile.get('email'));

    return normalizedProfileData;
}());
```

Open vkontakte-profile-normalization.js in your browser.

wechat-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
```

```
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
```

```
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('openid'));
    normalizedProfileData.put('displayName',
rawProfile.get('nickname'));
    normalizedProfileData.put('photoUrl',
rawProfile.get('headimgurl'));
    normalizedProfileData.put('username',
rawProfile.get('nickname'));

    return normalizedProfileData;
}());
```

Open wechat-profile-normalization.js in your browser.

wordpress-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
```

```
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
```

```
 *          ultimately, the managed object created/updated with
this data
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('username'));
    normalizedProfileData.put('displayName',
rawProfile.get('display_name'));
    normalizedProfileData.put('photoUrl',
rawProfile.get('avatar_URL'));
    normalizedProfileData.put('email', rawProfile.get('email'));
    normalizedProfileData.put('username',
rawProfile.get('username'));

    return normalizedProfileData;
}());
```

Open wordpress-profile-normalization.js in your browser.

yahoo-profile-normalization.js

▼ View script

```
/*
 * Copyright 2021 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with
ForgeRock AS
 * or with one of its affiliates. All use shall be exclusively
subject
 * to such license between the licensee and ForgeRock AS.
 */

/*
 * This script returns the social identity profile information
for the authenticating user
 * in a standard form expected by the Social Provider Handler
Node.
 *
 * Defined variables:
 * rawProfile - The social identity provider profile information
for the authenticating user.
 *              JsonValue (1).
 * logger - The debug logger instance:
 *          https://backstage.forgerock.com/docs/am/7/scripting-
guide/scripting-api-global-logger.html#scripting-api-global-
logger.
 * realm - String (primitive).
 *         The name of the realm the user is authenticating to.
 * requestHeaders - TreeMap (2).
 *                  The object that provides methods for
accessing headers in the login request:
 *
https://backstage.forgerock.com/docs/am/7/authentication-
guide/scripting-api-node.html#scripting-api-node-requestHeaders.
 * requestParameters - TreeMap (2).
 *                     The object that contains the
authentication request parameters.
 * selectedIdp - String (primitive).
 *               The social identity provider name. For example:
google.
 * sharedState - LinkedHashMap (3).
 *               The object that holds the state of the
authentication tree and allows data exchange between the
stateless nodes:
 *               https://backstage.forgerock.com/docs/am/7/auth-
nodes/core-action.html#accessing-tree-state.
 * transientState - LinkedHashMap (3).
```

```
 *                  The object for storing sensitive information
that must not leave the server unencrypted,
 *                  and that may not need to persist between
authentication requests during the authentication session:
 *
https://backstage.forgerock.com/docs/am/7/auth-nodes/core-
action.html#accessing-tree-state.
 *
 * Return - a JsonValue (1).
 *          The result of the last statement in the script is
returned to the server.
 *          Currently, the Immediately Invoked Function
Expression (also known as Self-Executing Anonymous Function)
 *          is the last (and only) statement in this script, and
its return value will become the script result.
 *          Do not use "return variable" statement outside of a
function definition.
 *
 *          This script's last statement should result in a
JsonValue (1) with the following keys:
 *          {
 *              {"displayName": "corresponding-social-identity-
provider-value"},
 *              {"email": "corresponding-social-identity-
provider-value"},
 *              {"familyName": "corresponding-social-identity-
provider-value"},
 *              {"givenName": "corresponding-social-identity-
provider-value"},
 *              {"id": "corresponding-social-identity-provider-
value"},
 *              {"locale": "corresponding-social-identity-
provider-value"},
 *              {"photoUrl": "corresponding-social-identity-
provider-value"},
 *              {"username": "corresponding-social-identity-
provider-value"}
 *          }
 *
 *          The consumer of this data defines which keys are
required and which are optional.
 *          For example, the script associated with the Social
Provider Handler Node and,
 *          ultimately, the managed object created/updated with
this data
```

```
 *          will expect certain keys to be populated.
 *          In some common default configurations, the following
keys are required:
 *          username, givenName, familyName, email.
 *
 * (1) JsonValue -
https://backstage.forgerock.com/docs/am/7/apidocs/org/forgerock/
json/JsonValue.html.
 * (2) TreeMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/TreeMap.html.
 * (3) LinkedHashMap -
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/jav
a/util/LinkedHashMap.html.
 */

(function () {
    var frJava = JavaImporter(
        org.forgerock.json.JsonValue
    );

    var normalizedProfileData =
frJava.JsonValue.json(frJava.JsonValue.object());

    normalizedProfileData.put('id', rawProfile.get('sub'));
    normalizedProfileData.put('displayName',
rawProfile.get('name'));
    normalizedProfileData.put('givenName',
rawProfile.get('given_name'));
    normalizedProfileData.put('familyName',
rawProfile.get('family_name'));
    normalizedProfileData.put('photoUrl',
rawProfile.get('picture'));
    normalizedProfileData.put('email', rawProfile.get('email'));
    normalizedProfileData.put('username',
rawProfile.get('email'));
    normalizedProfileData.put('locale',
rawProfile.get('locale'));

    return normalizedProfileData;
}());
```

Open yahoo-profile-normalization.js in your browser.

Was this helpful?