Claim source examples

This blurb has several examples for creating claim sources. Since these are apt to be wordy. it is better to have them organized in a specific place.

Examples.

Adding to the system claims

If you have a configuration names **`my_source.`** then you can simply add it to the default claims for the system:

```
   claim_sources. := claim_sources. ~ [create_source(cfg.)];
```

(Assuming you did not already invoke **`create_source.`**)

Using a script: NCSA claims

If there are existing QDL scripts use them. Here is a common example for the NCSA. It assumes that the user has come in through the NCSA IDP (it will simply do nothing for any other IDP) and gives the option of returning group claims as a JSON structure or a flat list. Generally you want to use this unless there is something very special you need. This is the block you would paste into your client's cfg attribute:

```
tokens{
 identity{
 type=identity
 qdl{
 load="ncsa/ncsa-default.qdl"
 xmd={exec_phase=["pre_auth","post_token", "post_exchange","post_refresh"]}
 args=["true"] // true if the member of claim is just a list. Default is false.
 }// end qdl
 } //end identity token
} //end tokens
```

NCSA claims directly

This is a specific type of LDAP handler that only talks to the internal NCSA LDAP.

```
cfg. := new_template('ncsa');
// any overrides you may need.
cfg. := create_source(cfg.);
```

You may then invoke this directly or add it to the system **`claim_sources.`** to be run for you. This is far more work than using the default script, but it is good to know this is just a very specific LDAP configuration that can be tweaked.

Generic LDAP claims

You may also create custom claims for an LDAP server. Here is a complete example for LIGO.

```
script_load('utils/init.qdl');
// return if the IDP is not right.
if[claims.idp != idp.ligo][return();];

// The ldap.search_name below is 'uid', so it must be set as a claim
           claims.uid := head(claims.eppn, '@');
                 cfg. := new_template('ldap');
        cfg.auth_type := 'none';
          cfg.address := 'ldap.ligo.org';
             cfg.port := 636;
       cfg.claim_name := 'uid';
      cfg.search_base := 'ou=people,dc=ligo,dc=org';
        cfg.ldap_name := 'uid';
          cfg.groups. := ['isMemberOf'];
cfg.search_attributes. := ['email', 'uin', 'uid', 'isMemberOf'];

 claim_sources. := claim_sources. ~ [create_source(cfg.)];
```

This will now be run with the standard claims for the server. The utility script at **`utils/init.qdl`** loaded at the beginning has, among other things, a list of common IDPs. If you have a login through the LIGO IDP, you can just create the **`cfg.`** and then issue a **`get_claims`** if you prefer. Again, adding it to the system claims means it always gets invoked in every phase, so you only have to set it in the id token handler.

Another LDAP example: Getting claims directly from NCSA LDAP

Just as an example, here is how to construct an LDAP query to NCSA. You can cut and paste this, but it does require either being onsite at the NCSA or a VPN connection thereto

```
cfg2. := new_template('ldap');
cfg2.address := 'ldap1.ncsa.illinois.edu,ldap2.ncsa.illinois.edu';
cfg2.auth_type := 'none';
cfg2.search_base := 'ou=People,dc=ncsa,dc=illinois,dc=edu';

// So the next two lines are key:

// This is the name of the attribute in LDAP to search on
cfg2.ldap_name := 'uid';

// There is a claim with this name, pass the value to the search engine
cfg2.claim_name := 'uid';

// The search in LDAP then is ldap_name == claim_name
// The next commands specify what attributes to get back from
// the server (rather than all of them)
cfg2.search_attributes. := ['email','uid','uin','memberOf'] ;

// Since memberOf is a group, we want it parsed rather than being a long messy
string
```

```
cfg2.groups. := ['memberOf'];

// Finally, the name in LDAP is 'memberOf' and the expected claim name is
'isMemberOf'
// So we set the rename of
cfg2.rename.memberOf := 'isMemberOf';

claims. := get_claims(create_source(cfg2.), 'YOUR_USER_NAME');
claims.
```

This gets the record for **`YOUR_USER_NAME`** so do supply your own.

HTTP Header claims

A typical example would be gettign claims in the HTTP Headers from SATOSA. These arrive prefixed with OIDC_ so a sub claim from the IDP would be

```
h. := new_template('http');
h.prefix := 'OIDC__'; // required to filter only headers we want
headers.. := get_claims(create_source(h2.), 'jgaynor');
```

A typical claim would be

```
   headers.OIDC__sub
http://cilogon.org/serverT/users/12345
```

To rename all of these is easy in QDL with the **`rename_keys`** function:

```
   headers. := rename_keys(headers., keys(headers.)-'OIDC_');
   headers.sub
http://cilogon.org/serverT/users/12345
```

You can add them all to the claims.

```
claims. := claims. ~ headers.
```

or just some subset of them:

```
claims. := claims. ~ headers.given_name ~ headers.family_name ~ headers.eptid;
```

File Claims

An example test file giving the claims structure can be found at test-claims.json.

```
f. := new_template('file');
f.file_path :=
'/home/ncsa/dev/ncsa-git/oa4mp/oa4mp-server-test-oauth2/src/main/resources/test-
claims.json';
file_claims. := get_claims(create_source(f.) , 'jeff');
```

Note that in the sample file, there is also a configured default user, so if you wanted to enable that you would have

```
f. := new_template('file');
f.file_path :=
'/home/ncsa/dev/ncsa-git/oa4mp/oa4mp-server-test-oauth2/src/main/resources/test-
claims.json';
     f.use_default := true;
   f.default_claim := 'default_claim'
file_claims. := get_claims(create_source(f.) , 'arglebargle');
```

And you would get whatever is the default record for the user.

Creating claims from Java code.

There is a test class for this `edu.uiuc.ncsa.myproxy.oa4mp.oauth2.claims.TestClaimSource` which will simply return a claims object that echos the parameters. Here's how to invoke it. Note that all you need to do is change the java class to your class (make sure its in the classpath!) and pass in your arguments. See the documentation for BasicClaimSourceImpl. In this example, we invoke this test claim source with two custom parameters, foo and bar:

```
  cfg. := new_template('code')
  cfg.java_class := 'edu.uiuc.ncsa.myproxy.oa4mp.oauth2.claims.TestClaimSource'
  cfg.foo := 'foo-test';
  cfg.baz := 'baz-test';

  my_claims. := get_claims(create_source(cfg.), 'jeff');
```

The **`create_source`** function adds required properties. If you display **`my_claims.`** A name must be supplied to the **`get_claims`** function, but the handler actually just ignores it. You get

```
   my_claims.
{
 config param "type":code,
 config param "foo":foo-test,
 config param "baz":baz-test,
 config param "enabled":true,
 config param "fail_on_error":false,
 config param
"java_class":edu.uiuc.ncsa.myproxy.oa4mp.oauth2.claims.TestClaimSource,
 info:Echoing configuration parameters.,
 config param "id":qdl_claim_source,
 config param "notify_on_fail":true
}
```

Note that the default parameters like **`notify_on_fail`** were added by the **`create_source`** function. This is wordy, so yes, the keys are things like **`'config param "type"'`**