QDL ini file for clients

If you want to run the command line client for OA4MP in QDL, then it can accept ini files rather than the standard XML file as a configuration. This blurb documents that.

Usage

You can use this with the command line client module, so a typical invocation would be

```
   j_use('ini');
true
   init('/path/to/ini/files/clients.ini','name_of_client');
true
```

The command line client (which is now local to the current workspace) has been initialized and is ready for use with the client named **`name_of_client`** (this is the name of the section in the ini file with the client configuration – see below).

The basics

The basic format for an ini file is

```
[name0]
id0:= identifier
extends:=name0, name1, name2, …
// other entries
[name1]
id1 := identifier
//etc.
```

What this means is that the name of the client must be a standard ini identifier, not just an id.

Inheritence

The *extends* keyword is a list of client names and the given client will inherit from them in order. (This allows for a very simple multiple inheritance mechanism, by the way). In the list

```
  extends := id_0,id_1,...id_n
```

The configuration for id_0 is overlaid with that of id_1, … id_n in turn and finally the current configuration is overlaid. Note that this will overlay each entry in the stem. All references are resolved before overlaying them. If this is not possible (e.g. circular dependence, **`extends := A,B,C,D,B`** then an error is raised.)

Example

```
[meta_root]
```

```
[meta_root.endpoints]
 well_known := 'https://localhost:9443/oauth2/.well-known/openid-configuration/oa4mp_test'
 // More!!

[root]
extends := 'meta_root'
   jwks := '/home/ncsa/dev/csd/config/auto-test/keys.json'

[ccf.oidc.basic]
        id := 'auto-test:/oidc/ccf'
   scopes  := 'openid'
   extends := 'meta_root'

[oauth.conf.basic]
     id:='auto-test:/oauth/conf'
    kid:='DAE4FADC4B9B8373'
extends:='root'
```

In this case **`ccf.oidc.basic`** would inherit from **`meta_root`** and **`oauth.conf.basic`** would inherit from **`root`** and **`meta_root`**.

Sections

The ini file divides naturally into sections. These are

Section	Entries	Description
(top)		The top level for the configuration.
	asset_lifetime	If asset store cleanup is enabled, this determines how long an asset that is unused is permitted to remain.
	callback	The callback aka redirect uri for this client, if there is one. Not all clients have these.
	debug_level	A string given the debug level. Allowed values are off, trace, info, warn, severe
	enable_asset_cleanup	If true will enable cleaning up old assets in the store. Default is ***false***.
	enable_oidc	Enable OIDC for this client. Default is ***true***.
	jwks	Path to the JSON web Keys on this system.
	scopes	A list of scopes that this client will send
	id	The identifier for this client
	kid	The ***k***ey ***id*** for the JSON Web key that this client will use if it is using private key authorization
	extends	list of configuration names, in order, from which this client inherits.
	secret	The secret,if there is one, for this client to use if using client credentials
	skin	If the service supports skins (i.e. custom look and feel), then you may pass this along

endpoints		The endpoints this client will use. See note below.
	authorization	The authorization endpoint
	client_management	The client management endpoint
	device	The device flow endpoint
	introspection	he introspection endpoint
	revocation	The revocation endpoint
	token	The token endpoint
	user_info	The user information endpoint
	well_known	The well-known endpoint
logging		Used for logging. If this is omitted, logging will be to standard out so it may get messy.
	count	An integer. The number of log files in the rotation. Once a file reaches it maximum size, another one is opened. This determines how many there are. default is 2.
	disable_log4j	Disable Log 4 Java. Since this is a dependency in many projects, this tell the system to aggressively track down instance and kill them. default is ***true***
	append_on	If appending entries to log files is enabled. Default is ***false***. This is used chiefly if a log file is written to by other applications and prevents the system from deleting old ones on start up.
	file	The base name of the file to write to. Note that this will get suffices depending on the count, max size etc.
	max_size	The largest (in bytes) a log file may be before rotation. Default is 10k.
	name	The internal name for entries. It is possible to have several applications write to the same log file. This allows you to separate out the entries
assets		The asset store
	type	fileStore or memoryStore.
(for file stores)	path	The base path for storage. Directories under this path are managed by the system
	remove_empty_files	If there are empty files found, remove them. Default is ***false***
	remove_failed_files	If an attempt is made to load a file and it fails, an attempt to remove the file is made. Default is ***false***
ssl		
ssl.trust_store		The trust store for the system. This is needed if you are, for instance, contacting a service with a self-signed certificate.

	cert_dn	The certificate distinguished name as found in the server's certificate.
	path	The full path to the certificate
	password	The password for the certificate file
	use_java	If true will also check the standard certs sent with Java. Default is **true**. Note that you may have this false and the client can then only speak to exactly on server.
	type	The trust store type. Supported types are JKS or PKCS12
ssl.keystore		The keystore. Generally this is not used for clients. Trust stores are used to contact other server, key stores are used for storing cert needed when receiving connections. The client itself never acts as a server.
	path	The full path to the keystore
	password	The password for the key store
	type	The type of keystore. Supported values are JKS or PKCS12.
extended_attributes		Parameters sent to the service. Normally these start with **org.oa4mp:** or **org.cilogon:** and have a path

Notes

The client may either specify the well-known endpoint for this service and all other values will be read from that. Or, it may specify the service URI and the system will create the default endpoints. Each endpoint, however, may be overridden explicit if you choose to do so.

Example

```
[root]
        jwks:='/home/ncsa/dev/csd/config/keys.jwk'
      scopes:='email','openid','profile','org.cilogon.userinfo'
max_lifetime := 1000000

  [root.endpoints]
  well_known := 'https://localhost:9443/oauth2/.well-known/openid-configuration'

 [root.logging]
  file:='/tmp/auto-test.log'
  size:=100000
 count:=2
  name:='auto-test'

 [root.assets]
 type:='file'
 path:='/tmp/oa4mp2/command-line'
```

```
   [root.ssl.trust_store]
       path:='/path/to/certs/localhost.jks'
   password:='mairzy doates'
       type:='JKS'
     certDN:='CN=localhost'

 [root.extended_attributes]
       oa4mp:/roles :='admin'
   oa4mp:/test/path :='a','b','c',42

/* Now for another client that inherits from root */
[commandline2]
id:='ashigaru:command.line2'
kid:='EC9FC3716AC4C22742EC98CF'
extends:='root'
```

So in this case there is a root configuration which is incomplete. The other configuration, commandline2 inherits from it and specifies the id and key to use.

Note: You may load clients using their stem coordinates, e.g. in the CLI

```
cli>load oauth.basic.override /path/to/file.ini
```