ConnId

ConnId

ConnId is a framework that does most of the provisioning operations in midPoint. It is a descendant of Sun Identity Connector Framework (Sun ICF) which was probably the only usable open-source part of Sun Identity Manager. ConnId is now an open project with contributors from several independent companies. Although there was a significant number of improvements over the original Sun version the basic approach and architecture remains mostly the same.

Project home page	http://connid.tirasa.net/
Project source code	https://github.com/Tirasa/ConnId

Connectors

TODO: Picture

The basic concept of ConnId is a *connector*. The connector is a piece of (usually) Java code that is governed by a uniform Java interface on one side (ConnId API/SPI) and that is using a variety of protocols and interfaces to connect to the resource on the other side. Connectors are distributed in *bundles* which is an ConnId packaging mechanism. However most connector bundles contain only a single connector. Connector bundles are versioned and the ConnId framework allows to run a connector in several versions at the same time.

Connector is identified by a triple consisting of:

- **Bundle name** which is usually in a form of maven group or java package name, such as `com.evolveum.polygon.connector-ldap`.
- **Bundle version** which is usually a "dotted decimal" notation. ConnId is using four version numbers (e.g. `1.4.2.0`) which is somehow inherited from Sun. The last number in version number indicates a "build" of a connector. This is derived from the source code control system. Official releases have the last number set to zero.
- **Connector type** which is usually in a form of fully qualified Java class name, e.g. `com.evolveum.polygon.connector.ldap.LdapConnector`.

Connector Configuration Schema

TODO .. in the meantime see Resource and Connector Schema Explanation.

Connectors and Resource Schema

TODO .. in the meantime see Resource and Connector Schema Explanation.

NAME and UID

There are two peculiarities when it comes to ICF resource schema design. ICF mandates that all resource objects **must** have two special attributes: `__NAME__` and `__UID__`. We do not like the underscores and therefore we refer to them as ICF NAME and ICF UID. Their meaning is as follows:

	Description	**Required for**	**Typical usage**	**ICF representation**	**XML representation**
ICF NAME	Represents user-friendly identifier of an object on a target resource. It should be human-readable and it is used to create an resource object (e.g. account username). It is the name of the object that humans usually care about.	create	username, LDAP DN	`__NAME__`	`icfs:name`
ICF UID	Represents the unique identifier of an object within the name-space of the target resource. Should be immutable if possible. It is returned from the *create* operation and must be used for other operations to identify the object. This may not be human readable and is often (pseudo-randomly) generated. This is the identifier of an object that machines usually care about.	get, modify, delete	Entry UUID, auto-generated ID column, record number	`__UID__`	`icfs:uid`

For some simple resources the UID and NAME may be the same. Actually that is a usual case that they are the same, which adds to the confusion. Also do not confuse the ICF UID with other attributes that may be named "uid" such as LDAP `uid` which is a part of `inetOrgPerson` schema. ICF UID is a separate and quite powerful concept which needs some time to get used to. In the early phases of the learning curve it helps to think about it simply as an *identifier* of a resource object.

Confusingly enough ICF NAME and UID usually take place of another account attributes. E.g. in "smart" LDAP deployment the ICF NAME usually contains distinguished name (`dn`) and ICF UID usually contains the data from `entryUuid` LDAP attribute. This is excellent setup to support object renames and moves which are surprisingly common in IDM deployments. But it also means that there will be no `dn` and `entryUuid` attributes seen in the resource schema. There will be ICF NAME and ICF UID instead. This "hiding" of such very important attributes is very difficult to get used to. We are working with the OpenICF team to somehow make it more user friendly to use by leveraging ICF schema meta-data. But for now we have to live with ICF NAME and ICF UID.

Note: XML representation of the objects is using URIs instead of arbitrary prefixes (such as underscores) to denote namespaces. Therefore the XML representation of ICF NAME and UID is `icfs:name` and `icfs:uid` (`icfs` is a namespace prefix that refers to actual URL). Please take care not to confuse this with an actual *attribute* named "uid" such as LDAP uid. The attribute is using a different namespace and is usually denoted `ri:uid`.

Tip: ICF NAME and ICF UID attributes may be "renamed" to a more user-friendly names using `displayName` clause in Resource Schema Handling. This does not change the true nature of the attributes nor does it change the XML representation. But it may improve user experience.

ConnId Issues

Although there are several known and painful issues in ICF design there are not critical. There are many exiting ICF-comaptible connectors developed by several independent companies in the scope of several projects. The benefits of ICF greatly outweighs the drawbacks. Therefore we have chosen to stick with ConnId as the primary provisioning framework in midPoint. We are actively contributing to ConnId development. Our goal is to evolve the framework and resolve most of the original ICF issues.

ConnId Framework Logging

Useful information may also be provided by the logging the operations of the ConnId connector framework. This is very useful in cases that a suspected problem is in the interpretation of the values (e.g. data type conversions). This logs all the communication between connector, connector framework and midPoint. It can be enabled by setting the logging to:

```
org.identityconnectors.framework.api: TRACE
```

See Also

- OpenICF
- ICF Issues
- Connector Development Guide