# Creating your own Schema in M-Vault for Test and Evaluation Purposes

# 1 Introduction

The purpose of this document is to outline the process for creating your own schema in the Isode M-Vault Directory server in a test/evaluation environment. More detailed information can also be found in Section 5.1.6 of the M-Vault Administration Guide and also within the headings of files referenced in this document. This document is useful for those wanting to modify schema, without detailed review of the manual.

# 2 Background

One of the key benefits of using LDAP or X.500 directories is that the core schema can be readily extended, using standard and proprietary schema. This means that an organization can extend information in the directory to meet its own purposes. The schema is important, as it will be used throughout the organization, and may be relied on by applications that are built to use the directory. It can be seen that schema changes are important, and should be made with some care. A typical process is:

- Design the schema for the organization. It is important to get this right.

- Build and test the schema, separately from the production service.

- Distribute the schema to all servers and applications.

The Isode schema is represented as text files, which makes distribution and management straightforward for this sort of process.

This document describes the mechanics of changing schema in the Isode directory. It can be used immediately by those who wish to try out Isode's schema management. If you are building a production system, we recommend you read the last section of this note before starting.

# 3 Where to Start

Existing schema is defined in the oidtable.oc, oidtable.at & oidtable.gen files. These files are stored in the following locations:-

| | |
|---|---|
| Red Hat Linux | /opt/isode/share |
| Solaris | /opt/isode/share |
| Windows | C:\Program Files\Isode\share |

If the schema is already defined then all that is required is to use the appropriate Object Class in the definition of your entries. If your schema does not match any of these then you will need to design you own. The simplest way to do this is define

your schema in additional files rather than editing the supplied oidtable files. This has two advantages in that it is easier to manage and the files will not be overwritten when upgrades are installed. You will need to create the following three files:-

| | |
|---|---|
| myschema.oc | Object Class Definition |
| myschema.at | Attribute Definition |
| myschema.gen | Glue |

The myschema part of the filename can be replaced by your company name or something else that is indicative of the Object Class(es)..

# 4 Object Class Definition

Below is a sample myschema.oc file showing how to define your Object Class(es).

*# myschema object class(es)*

*myschemaUser: myschema-oc.1: top: \*
    *commonName, footballTeam, playingPosition: \*
    *rfc822Mailbox, passNumber, insideLeg, chestSize: \*
    *kind=structural*

In this file "commonName", "footballTeam", "playingPosition", "rfc822Mailbox", "passNumber", "insideLeg" & "chestSize" are the attributes for the Object Class "myschemaUser". The "myschema-oc.1" indicates that this is the first object class definition for myschema, the next one would be "myschema-oc.2" and so on. The "kind=structural" indicates that this is a core object class, alternatively it could be "kind=auxiliary" to define additional attributes for certain objects as a subclass to the core object class . The "top" indicates that this is a top level Object Class.

# 5 Attribute Definition

Once you have detailed your attributes for the object class in the previous section you need to define the type of those attributes not already defined in the oidtable.at file. All this is done in the myshecma.at file as shown below:-

*# mychema attribute types*

| | | |
|---|---|---|
| *footballTeam:* | *myschema-at.1:* | *CaseExactString: single-value* |
| *playingPosition:* | *myschema-at.2:* | *CaseExactString: single-value* |
| *passNumber:* | *myschema-at.3:* | *Integer: single-value* |
| *insideLeg:* | *myschema-at.4:* | *Integer: single-value* |
| *chestSize:* | *myschema-at.5:* | *Integer: single-value* |

Attributes that are already defined in the *oidtable.at* file such as *commonName* and *rfc822Mailbox* should take their attribute identifier from that file. The remaining attributes should be assigned numbers sequentially as shown above. Available Schema Types are detailed in Appendix A of the M-Vault Administration Manual.

# 6 The myschema.gen File

This is the glue that brings the attributes and object class(es) together and an example is shown below:-

*# REPLACE 1.2.3.4.5 with enterprises.<something> if you register with IANA.*

| | |
|---|---|
| *myschema:* | *1.2.3.4.5* |
| *myschema-at:* | *myschema.1* |
| *myschema-oc:* | *myschema.2* |

As this file explains to use this Schema in the wider Internet Community you should obtain an oid number from IANA, which would take the form enterprises.<some set of numbers separated by dots e.g 21.5785.34.28>.

# 7 Incorporating the New Schema

You have now defined your schema and need to add it to the existing schema, this is achieved by use of a file called dsaptailor. In the following directories you will find a file called dsaptailor.sample.

| | |
|---|---|
| *Solaris* | */etc/isode* |
| *Red Hat Linux* | */etc/isode* |
| *Windows* | *C:\Isode\etc* |

Take a Copy of this file and name it *dsaptailor* then after the initial comments are the lines:-

*# this line must occur first, if changed*
*# oidtable      oidtable*

Change these lines as shown below:-

*# this line must occur first, if changed*
*oidtable      oidtable myschema*

The schema is now ready for use in the directory and the directory server can be started using EDM, if EDM fails to start is likely that there is an error in the schema files and they should be checked for accuracy. If you would like to add entries to your directory serer with the new schema using DDM you will need to add a template for this into DDM otherwise your directory server and new schema are ready for testing.

# 8 Creating a Template for DDM

DDM (Directory Data Manager) is Isode's data management tool.  It is "aware" of schema, and uses templates to conveniently display information.  This section describes how to modify these templates.

To create a template for your schema in DDM you will need to edit the following
file:-

| | | |
|---|---|---|
| *Solaris* | *.ddm.ini* | *In the users home directory* |
| *Red Hat Linux* | *.ddm.ini* | *In the users home directory* |
| *Windows* | *ddm.ini* | *In C:\Documents and Settings\* |
| | | *<userid>\Application Data\Isode* |

Below is detailed the modifications required for the sample schema myschemaUser.

In the section titled *[Templates]* add the following line:-

*MySchema User = top, myschemaUser:myschemauserTemplate*

Then after this section add the following lines:-

*[markschemaUserTemplate]*
*; Attribute details for the markschemaUser template*
*Common Name=commonName:mand,single,edit, rdn*
*Football Team=footballTeam:optional,single,edit*
*Playing Position=playingPosition:optional,single,edit*
*Email Address=rfc822Mailbox:optional,multi,edit*
*Pass Number=passNumber:optional,single,edit*
*Inside Leg=insideLeg:optional,single,edit*
*Chest Size=chestSize:optional,single,edit*

Your schema will now appear in the list identified by a grey "?" icon. If you want to
change this to the standard person icon then at the end of the file is the section
*[ObjectClassToPicture]* add the following line;-

*markschemaUser=8*

Once this file is working for you should take a backup of it as upgrades will
overwrite this file.

# 9  Testing the Schema

The simple test for the Schema is to start the directory server and if it does not start
then there is a problem with your schema definition. Further details on the nature of
the problem will be found by examining the "dsap.log" file. This file can be found in
the following locations:-

| | |
|---|---|
| *Solaris* | */var/isode/log* |
| *Red Hat Linux* | */var/isode/log* |
| *Windows:* | *C:\Isode\log* |

# 10  Production Systems

Schema is used by many components, and it is worth taking some care in schema design. In order to use this schema in a production environment some or all of the following steps should be followed:-

- See if there is standard schema to do what you need.  If you use standard schema, applications may be able to use this directly, without modification to use private schema.

- Review existing defined object classes and see if you can create an auxiliary object class to one already defined.  It is better to build on standard objects (rather than create new ones), as applications may have built in support for standard schema.

- Ensure that none of your attributes names have already been defined checking at the following URL:-

     http://www.alvestrand.no/objectid/top.html

   Conflicts could cause interoperability problems.

- Register your OID with IANA and make the necessary modification to the myschema.gen file.  This registration is quite straightforward, and ensures that your schema will never conflict with anyone else's.