DS Design Summary 2

From Idmwiki

Contents

Policies

The policies consist of the three different parts.

- Configuration management
- Role management

- Action execution

Previously we thought that actions should be handled in a special way however the proposed data structure allows handling actions the same way as the configuration policies. The actions become just a configuration policy for an "Shell" application. See details in the action section late on this page.

Containers

To express, manage and use the policies we need to build a specific tree structure described below. In the nodes of the tree we need to put container objects. Originally we thought that we will use nsContainer class for this purpose, however, the nsContainer class does not provide a "description" attribute. To remediate this we decided to create a base IPA class that we will use for node entries in the tree:

```
objectclass ( 2.16.840.1.113730.3.8.4.TBD
   NAME 'ipaContainer'
   SUP nsContainer
   STRUCTURAL
   MAY description
   X-ORIGIN 'IPA v2' )
```

Tree Top

All the policy management related objects described below will be located under: cn=policies, dc=... For the configuration management we will create a container named "configs" and for role related data we will have "roles" subtree:

```
cn=configs,cn=policies, dc=...
cn=roles,cn=policies, dc=...
```

Entry:

```
dn: cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: policies
description: Root of the policy related sub tree
```

Entry:

```
dn: cn=configs,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: configs
description: Root of the sub tree that holds configuration policies for different applications
```

Entry:

```
dn: cn=roles,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: roles
description: Root of the sub tree that holds role management data
```

Configuration Management

Concepts

The configuration management consists of the following concepts:

- **Application**. This is some sort of application that runs on the client machine (SUDO, SELinux, IPA Client, IPTables, etc.) and needs to be centrally configured.
- **Configuration Template**. Applications might have different sub components that need to be managed

differently; applications evolve and one might have significantly different versions of the same application in the environment; an application may have different classes of the configurable properties access to which should be split between different administrators. All these examples show that under umbrella of one application there might be several different configuration profiles not only in terms of values but in terms of structure too. While they are somewhat different they are related to one application and should be treated as a family. This led us to the concept of the "Configuration template". The template constitutes a pair between the description of the structure policy and the description of how to interpret and apply the policy.

- **Policy** - a collection of the configuration settings applicable to a specific application template
- **Policy Group** - a set of policies for different applications and and templates distributed together
- **Host Applicability** - not all policies are applicable to the same set of hosts; the Policy Groups are targeted for set of hosts and delivered to those hosts by the IPA system.

Attributes and Objects

Application

The applications will be represented by the container objects (ipaContainer) located in the following part of the tree: cn=applications,cn=configs,cn=policies, dc=... The name of the application will be stored in CN and will be used as part of DN. The names of the applications tend to not change much. And if they are they can be treated as a different application. This assumption allows us to ignore the DS limitation related to sub tree renames.

Entry:

```
dn: cn=applications,cn=configs,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: applications
description: Root of the tree that hold all definitions of the supported applications
```

Entry:

```
dn: cn=SUDO,cn=applications,cn=configs,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: SUDO
description: SUDO gives root privileges for certain applications
```

Entry:

```
dn: cn=SELinux,cn=applications,cn=configs,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: SELinux
description: SELinux - advance Linux security environment
```

Policy Template

Definition

The templates will be the objects that will live inside the application container and will describe how to handle the policies for this version of the application. Handling really means: describing what data constitutes the policy and defining the rules how to process the values that constitute the policy. The policies will be represented by the XML files. We are planning to use Relax NG schema definition language to describe the structure of the policy. The Relax NG will describe not only data but the UI too. The Relax NG schema will be stored in a special XML schema file. As we mentioned above in the concepts section one and the same application can hold multiple types of configurations. The processing rules are defined by the XSL template. The XSLT and RNG file go hand in hand. A pair of these constitute the configuration template for the application. In IPA v2 we will store these files on the server's file system, not in the DS itself. To describe a template the following entry is defined.

Entry:

```
dn: ipaUniqueID=5c8a3750-cc68-11dd-ad8b-0800200c9a66,cn=SUDO,cn=applications,cn=configs,cn=policies,dc=...
objectclass: ipaPolicyTemplate
ipaUniqueID: 5c8a3750-cc68-11dd-ad8b-0800200c9a66
cn: sudo_config_1
description: Basic SUDO policy
ipaPolicyType: config
ipaSchemaFile: /var/lib/ipa/policy/sudo_config_1.rng
ipaTrasformFile: /var/lib/ipa/policy/sudo_config_1.xslt
```

As one can see we will use special attributes to point to the files in the file system. We considered the use of the URIs but came to conclusion it would create a lot of unneeded complexity. If once we decide to put the files into the DS we will add a pair of attributes that will be of the DN type.

Attributes

Policy type is the type of the policy. For configuration policy the value will be "config". Though currently we anticipate that the values of this attribute are mutually exclusive and there can be only one type of the policy per policy we will leave it multi value for the potential future extension.

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaPolicyType'
  DESC 'Type of the policy'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-ORIGIN 'IPA v2')
```

The attributes that point to RNG and XSLT files are multi valued on purpose. This would potentially allow having multiple RNGs per policy allowing system to try until it finds the right one. In IPA v2 we will support only one RNG and one XSLT file per template.

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaSchemaFile'
  DESC 'Name of the file with schema definition'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-ORIGIN 'IPA v2')
```

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaTrasformFile'
  DESC 'Name of the policy transformation file'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-ORIGIN 'IPA v2')
```

Object Class

Most of the attributes are required since their absence makes the entry unusable.

```
objectclass ( 2.16.840.1.113730.3.8.4.TBD
  NAME 'ipaPolicyTemplate'
  SUP top
  STRUCTURAL
  MUST ( cn $ ipaUniqueID $ ipaPolicyType $ ipaSchemaFile )
  MAY ( ipaTrasformFile $ description )
  X-ORIGIN 'IPA v2' )
```

The ipaTrasformFile attribute can be omitted in case of the action policies. See example later on this page.

Policy Group

Definition

Policy groups will be objects located in the following part of the tree:
cn=policygroups,cn=configs,cn=policies,dc=...
The "policygroups" will be a normal new "ordered" container object with an ordering attribute added to it. The ordering attribute will be named ipaOrderedUUIDList and will hold the list of the UUIDs of the child entries in the proper order separated by the "$" symbol. Also container might have the attributes that describe who and when changed the order.

Entry:

```
dn: cn=policygroups,cn=configs,cn=policies,dc=...
objectclass: ipaContainer
objectclass: ipaOrderedContainer
cn: policygroups
description: Sub tree to hold policy groups
ipaOrderedUUIDList: 537e5f90-cc6f-11dd-ad8b-0800200c9a66 $ b03e0f50-ce22-11dd-ad8b-0800200c9a66
ipaLastChangeBy: uid=sbose,cn=users,cn=accounts,dc=...
ipaLastChanged: 20090101000000
```

The entries under policygroups will look like this.

Entry:

```
dn: ipaUniqueID=537e5f90-cc6f-11dd-ad8b-0800200c9a66,cn=policygroups,cn=configs,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
objectclass: ipaOrderedContainer
objectclass: ipaPolicyGroup
ipaUniqueID=537e5f90-cc6f-11dd-ad8b-0800200c9a66
cn: Westford Policy Group
description: Policy Group applicable to all hosts in Westford
ipaOrderedUUIDList: 2ef2a400-cc70-11dd-ad8b-0800200c9a66 $ 2342a400-cc70-11dd-ad8b-0800200c9a66 $ 3df2a400-cc70-11dd-ad8b-0800200c9a66
ipaLastChangeBy: uid=sbose,cn=users,cn=accounts,dc=...
ipaLastChanged: 20090101000000
ipaEnabledFlag: true
ipaAllowedTemplateRef: ipaUniqueID=5c8a3750-cc68-11dd-ad8b-0800200c9a66,cn=SUDO,cn=applications,cn=configs,cn=policies,dc=...
ipaAllowedTemplateRef: ipaUniqueID=5829d430-cc70-11dd-ad8b-0800200c9a66,cn=SELinux,cn=applications,cn=configs,cn=policies,dc=...
ipaAllowedTemplateRef: ipaUniqueID=5829d430-cc70-11dd-ad8b-0800200c9a66,cn=MRG,cn=applications,cn=configs,cn=policies,dc=...
```

Attributes

For the container object we add ipaOrderedUUIDList, ipaLastChangeBy and ipaLastChanged attributes. The following attribute holds the list of the UUIDs:

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaOrderedUUIDList'
  DESC 'Defines order of the entities within some sort of ordered group'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-ORIGIN 'IPA v2')
```

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaLastChangeBy'
  DESC 'DN of the user who caused the configuration change'
  SUP owner
  EQUALITY distinguishedNameMatch
  ORDERING distinguishedNameMatch
  SUBSTR distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  X-ORIGIN 'IPA v2' )

attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaLastChanged'
  DESC 'Last time there was some change to the data'
  EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
  SINGLE-VALUE
  X-ORIGIN 'IPA v2' )
```

For the policy group we also need ipaEnabledFlag which also reused by other objects including ipaAssociation and ipaAllowedTemplateRef.

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
   NAME 'ipaEnabledFlag'
   DESC 'The flag to show if the association is active or should be ignored'
   EQUALITY booleanMatch
   ORDERING booleanMatch
   SUBSTR booleanMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
   SINGLE-VALUE
   X-ORIGIN 'IPA v2')
```

The ipaAllowedTemplateRef attribute points to allowed policy templates. Only policies created using these templates are allowed to be included into the policy group. If the attribute is not defined then any policy can be added to the group.

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
   NAME 'ipaAllowedTemplateRef'
   DESC 'DN of the allowed policy template'
   SUP distinguishedName
   EQUALITY distinguishedNameMatch
   ORDERING distinguishedNameMatch
   SUBSTR distinguishedNameMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
   X-ORIGIN 'IPA v2' )
```

Object Class

The nsContainer class makes cn a required attribute. The ipaOrderedContainer class adds the ipaOrderedUUIDList attribute to the ipaContainer class.

```
objectclass ( 2.16.840.1.113730.3.8.4.TBD
   NAME 'ipaOrderedContainer'
   SUP ipaContainer
   STRUCTURAL
   MAY ( ipaOrderedUUIDList $ ipaLastChangeBy $ ipaLastChanged )
   X-ORIGIN 'IPA v2' )
```

If ordered attribute is omitted this means that there are yet no groups to order. As soon as any changes are made or anthing is added to the container the ipaLastChanged and ipaLastChangeBy should be populated.

The policy group object class is derived from the ipaOrderedContainer since it is both a container and requires its members to be ordered.

```
objectclass ( 2.16.840.1.113730.3.8.4.TBD
   NAME 'ipaPolicyGroup'
   SUP ipaOrderedContainer
   STRUCTURAL
   MUST ( ipaUniqueID $ ipaEnabledFlag )
   MAY ipaAllowedTemplateRef
   X-ORIGIN 'IPA v2' )
```

Policy

Definition

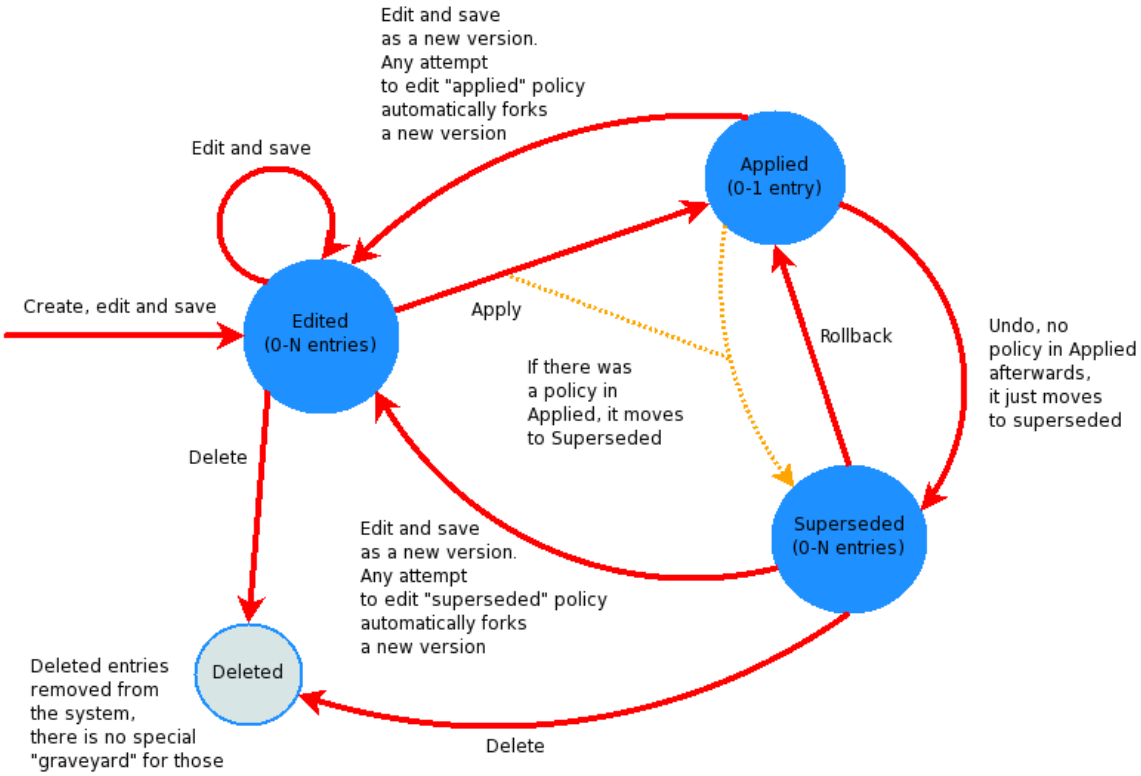
The "policy" turned out to be a pretty complex concept so we decided to split it into couple related objects. The first one is the policy itself. The policy has to express that it belongs to a specific template and that it is enabled. It also needs to express the history of the policy. The policy can be in tree different states. It can be an "edited" policy, which means that the policy is in works and not active yet. It can be "applied", which means that administrator has applied the policy and it is now active. It can be "superseded" which means it was applied but then a new version was pushed over. The policy would change the state depending upon the administrator's actions. The following table shows the states and operations that define the transition of the policy between states.

Initial state	Operation	Final state
no policy	Start editing a policy and then save it.	Edited
no policy	Start editing and did not save it.	no policy
Edited	Continue editing and save over. One can't edit policies that are "applied" or "superseded" and save them in place, only save it with a new name.	Edited
Edited Applied Superseded	Edit and save as a new version.	Edited
No active policy. Prepared a policy to apply which is in "edited" state.	Apply policy	Applied
There is an active policy. Prepared a policy to apply over. The policy is in "edited" state.	Apply policy	New policy moves to "applied"state. Old, previously active policy moves to "superseded" state
Applied	Undo applied policy	Active policy moves to "superseded" state. No policy is in the "applied" state
One policy is in applied state and there are superseded policies	Rollback	Currently "applied" policy moves to "superseded" state. Last "superseded" policy becomes "applied".
One policy is in applied state and there are no superseded policies	Rollback (result is same as Undo in this case)	Active policy moves to "superseded" state. No policy is in the "applied" state
Edited	Delete	Policy deleted
Superseded	Delete (privileged operation)	Policy deleted
Applied	Delete - not allowed	N/A

The full list of actions consists of:

- create - the policy data was created but the policy itself is not defined yet
- edit - the policy data was edited overwriting previous data
- save as - the policy data was saved as a different data blob
- apply - administrator applied policy
- undo - administrator undid current policy and no policy is currently active
- rollback - restore one of the previous policies
- delete - policy is deleted so it this action is never recorded in the policy data

The following diagram shows states and transitions between them:



It is clear that over time the history of the policies will grow. In v3 we will create a special utility to clear the history of the policies.

The policy created based on the SUDO template will look like this.

Entry:

```
dn: ipaUniqueID=a5120010-ccff-11dd-ad8b-0800200c9a66,ipaUniqueID=537e5f90-cc6f-11dd-ad8b-0800200c9a66,cn=policygroups,cn=configs,cn=po
objectclass: nsContainer
objectclass: ipaContainer
objectclass: ipaPolicy
ipaUniqueID: a5120010-ccff-11dd-ad8b-0800200c9a66
cn: sudoPolicy_1
description: A Sudo Policy
ipaTemplateRef: ipaUniqueID=5c8a3750-cc68-11dd-ad8b-0800200c9a66,cn=SUDO,cn=applications,cn=configs,cn=policies,dc=...
ipaEnabledFlag: true
ipaLastChangeBy: uid=sbose,cn=users,cn=accounts,dc=...
ipaLastChanged: 20090101000000
```

The second object that we decided to create is the object that would actually contain the policy data. The policy data will be stored under the policy entry as a child entry. It is done this way for better access control granularity.

Entry:

```
dn: ipaUniquesID=091702e0-cd00-11dd-ad8b-0800200c9a66,ipaUniqueID=a5120010-ccff-11dd-ad8b-0800200c9a66,ipaUniqueID=537e5f90-cc6f-11dd-
objectlass: ipaPolicyData
ipaUniquesID: 091702e0-cd00-11dd-ad8b-0800200c9a66
cn: sudoPolicyData_1
ipaPolicyBlob: <compressed policy data>
ipaLastChangeBy: uid=sbose,cn=users,cn=accounts,dc=...
ipaLastChanged: 20090101000000
ipaPolicyState: applied
```

Entry:

```
dn: ipaUniquesID=101b22c0-cd04-11dd-ad8b-0800200c9a66,ipaUniqueID=a5120010-ccff-11dd-ad8b-0800200c9a66,ipaUniqueID=537e5f90-cc6f-11dd-
objectlass: ipaPolicyData
ipaUniquesID: 101b22c0-cd04-11dd-ad8b-0800200c9a66
cn: sudoPolicyData_2
ipaPolicyBlob: <compressed policy data>
ipaLastChangeBy: uid=sbose,cn=users,cn=accounts,dc=...
ipaLastChanged: 20090101000000
ipaPolicyState: superseded
```

The two policy data objects above contain the ipaPolicyBlob attribute that is blob of the compressed XML policy. We considered base64 encoding it first but the encoding expansion would significantly reduce the benefits of compression. As a first iteration we will use a binary attribute. If we see any problems we will reconsider.
There potentially can be another attribute in the policy data class named "ipaPolicyURL". We might decide to store policy data outside the DS on the files system. If we do so we will use ipaPolicyURL to point to that object. Since it is not needed at the moment and we do not plan to use it in v2 we might consider omitting it completely. We would be able to add it to the ipaPolicyData object class later or create and auxiliary class for it.

Attributes

The new attributes for the policy class is ipaTemplateRef. It is important that it is a single value attribute since policy can by nature be created only using one template:

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaTemplateRef'
  DESC 'DN of the allowed policy template'
  SUP distinguishedName
  EQUALITY distinguishedNameMatch
  ORDERING distinguishedNameMatch
  SUBSTR distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  SINGLE-VALUE
  X-ORIGIN 'IPA v2' )
```

Other attributes are reused.

For the policy data the new attributes are:

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaPolicyBlob'
  DESC 'Compressed XML policy data in binary format'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
  X-ORIGIN 'IPA v2' )
```

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaPolicyState'
  DESC 'State of the policy data'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE
  X-ORIGIN 'IPA v2')
```

Other attributes are reused from other classes.

Object Classes

The ipaLastChangeBy and ipaLastChanged attributes will be used to track that the policy changed. Since the DS does not support transactions but when we rollback policy there might be a situation for a split second when there is not active policy it might lead to the situation when client will undo the policy instead of rollback. This might be a very bad precedent so to avoid this the client should always look at the time stamp of the policy itself. The time stamp in the policy will be updated after the switch of the state in both child entries is complete. The ipaLastChangeBy and ipaLastChanged attributes might be omitted when there are no policy data child entries yet. The object class for the policy is defined below.

```
objectclass ( 2.16.840.1.113730.3.8.4.TBD
   NAME 'ipaPolicy'
   SUP ipaContainer
   STRUCTURAL
   MUST ( ipaUniqueID $ ipaEnabledFlag $ ipaTemplateRef )
   MAY ( ipaLastChangeBy $ ipaLastChanged )
   X-ORIGIN 'IPA v2' )
```

When we were discussing the ipaPolicy object we debated the need to have a special attribute that would point to the currently applied policy out of its children. We see this as a potential optimization if we see that the performance of the searches shall be improved. For now the ipaPolicy object will not include such attribute.

```
objectclass ( 2.16.840.1.113730.3.8.4.TBD
   NAME 'ipaPolicyData'
   SUP top
   STRUCTURAL
   MUST ( ipaUniqueID $ cn $ ipaPolicyState $ ipaLastChangeBy $ ipaLastChanged )
   MAY ( ipaPolicyBlob $ description )
   X-ORIGIN 'IPA v2' )
```

The ipaPolicyBlob should be filled as soon as administrator saves the policy data. It is made optional because in future we might have a different attribute that will point to policy data blob stored externally.

Policy Link

Definition

Policy link represents the association between the policy group and a collection of hosts. It is similar to ipaAssociation and thus derived from it. However it would be ugly to use the user reference to point to policy group. For this purpose we create a new attribute that would point to policy groups rather than users. We also do not need to have a category for policy groups. The notion of "all policy groups" does not make much sense. It is very unlikely that there ever be an environment where one would want to apply all defined policy groups to one and the same host or set of hosts. Creating and applying policies can significantly affect the host they apply to so these operations should be performed with extreme caution. Other than that the policy link is just an association.
Policy links will be located under special container entry outside of the policy group sub tree. This is done on purpose. With such organization we can accomplish the separation of duties between administrators who manage policy groups, administrators who manage policies within policy groups and administrators that manage hosts and thus need to be able to create policy links. The only problem is that the policy links would be accessible by all administrators that are granted access to policy link hive. This means that policy link management should be done by a high level admin. This approach, however, might not be acceptable in all environments. This can be solved by creating custom ACIs to manage the individual links. This might be viewed as a too complex way. The alternative will be to add an multi value attribute to the entry that will contain the dn of the owner or owners. The "owner" attribute will serve well in this case. We will add it to the object class. If the customer is not satisfied with the model that only high level admin can manage the links he would be able to change one ACI to look at this attribute and populate this attribute with the DNs of the right users.
Policy link object will look like this:

Entry:

```
dn: ipaUniquesID=221b22c0-cd04-11dd-ad8b-0800200c9a66,cn=policylinks,cn=configs,cn=policies,dc=...
objectclass: ipaPolicyLink
objectclass: ipaAssociation
ipaUniquesID: 221b22c0-cd04-11dd-ad8b-0800200c9a66
cn: link_sudo_1
ipaPolicyGroupRef: ipaUniqueID=537e5f90-cc6f-11dd-ad8b-0800200c9a66,cn=policygroups,cn=configs,cn=policies,dc=...
memberHost: dn: cn=Westford,cn=computergroups,cn=accounts,dc=...
description: Links a SUDO related policy group to the Westford hosts
```

or like this in rare cases when the policy group should be applied to all hosts in the enterprise

Entry:

```
dn: ipaUniquesID=221b22c0-cd04-11dd-ad8b-0800200c9a66,cn=policylinks,cn=configs,cn=policies,dc=...
objectclass: ipaPolicyLink
objectclass: ipaAssociation
ipaUniquesID: 221b22c0-cd04-11dd-ad8b-0800200c9a66
cn: link_sudo_1
ipaPolicyGroupRef: ipaUniqueID=537e5f90-cc6f-11dd-ad8b-0800200c9a66,cn=policygroups,cn=configs,cn=policies,dc=...
hostCategory: all
description: Links SUDO policy to all hosts
```

The policy link entries will be created as child entries under the policylink standard container.

Attributes

New attribute:

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaPolicyGroupRef'
  DESC 'DN of the member policy group reference'
  SUP distinguishedName
  EQUALITY distinguishedNameMatch
  ORDERING distinguishedNameMatch
  SUBSTR distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  X-ORIGIN 'IPA v2' )
```

Object Class

The object class for IPA policy link will look like this:

```
objectclass ( 2.16.840.1.113730.3.8.4.TBD
  NAME 'ipaPolicyLink'
  SUP ipaAssociation
  STRUCTURAL
  MAY ( ipaPolicyGroupRef $ owner )
  X-ORIGIN 'IPA v2' )
```

All attributes other than ipaPolicyGroupRef are already defined by the association.

Role Management

High Level Structure

To start managing roles, first of all, one has to have a list of the applications for which IPA will manage roles. The applications for the role management purpose can be different than for the configuration purpose. For example we plan to provide configuration for SUDO but there is no role management for SUDO. For Virt we will provide role management but not configuration management. We will create application hives under the roles to keep the list of the applications and the template that will describe how to process the role data for the application.
In case of the roles, there is only one role policy per application. The role policies should not be stored in one hive for better separation of duties. If they stored in one place there would be hard to define an access rule that would allow an administrator to manage roles for only selected subset of applications. To achieve better access control granularity we will create the roledata hive on per application basis. Since we plan to do it under application we want to separate the templates (that will be accessible only to the Directory Manager) and the roledata. To accomplish this the tree will look like this:

```
cn=templates,cn=<application>,cn=roles,cn=policies,dc=...
cn=roledata,cn=<application>,cn=roles,cn=policies,dc=...
```

Also the "relations" (the associations of the users and hosts to a role) should be accessible on per application basis. The administrator who defines relations for SELinux might not have the right to manage the "relations" for other application and vice verse. To accomplish that the relations should also be created under a separate hive on per application basis.

```
cn=relations,cn=<application>,cn=roles,cn=policies,dc=...
```

The objects that will be stored inside the relation container will reference the policy names defined in the role definition policy. By the roles are not equal and should be prioritized between each other. For example if a user can be mapped to multiple roles then which role he should assume? There need to be a way to describe this. For this purpose the relations would be stored in the container that would offer two additional attributes than the ipaContainer. One attribute will store the type of the role model that should be used - exclusive or inclusive, the other will store the priority list of the roles.

Sample Data

Containers

Let us say that we are now looking at the "SELinux" application. First there will be a normal container object created with 3 sub containers

Entry:

```
dn: cn=SELinux,cn=roles,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: SELinux
description: SELinux role related data
```

Entry:

```
dn: cn=templates,cn=SELinux,cn=roles,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: templates
description: Template for the SELinux role policy data
```

Entry:

```
dn: cn=roledata,cn=SELinux,cn=roles,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: roledata
description: SELinux role policy definition
```

Entry:

```
dn: cn=relations,cn=SELinux,cn=roles,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaRelationsContainer
cn: relations
ipaRoleType: exclusive
ipaRoleOrder: administrator,user,guest
description: Mappings of the users to the SELinux roles
```

Let us say that we also have the "MRG" application.

Entry:

```
dn: cn=MRG,cn=roles,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: MRG
description: MRG role related data
```

Entry:

```
dn: cn=templates,cn=MRG,cn=roles,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: templates
description: Template for the MRG role policy data
```

Entry:

```
dn: cn=roledata,cn=MRG,cn=roles,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: roledata
description: MRG role policy definition
```

Entry:

```
dn: cn=relations,cn=MRG,cn=roles,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaRelationsContainer
ipaRoleType: inclusive
ipaRoleOrder: administrator,user,guest
cn: relations
description: Mappings of the users to the MRG roles
```

Template

The template object then will go into the template container:

Entry:

```
dn: ipaUniqueID=0e84fb20-cd40-11dd-ad8b-0800200c9a66,cn=templates,cn=SELinux,cn=roles,cn=policies,dc=...
objectclass: ipaPolicyTemplate
ipaUniqueID: 0e84fb20-cd40-11dd-ad8b-0800200c9a66
cn: SELinux role template
description: Role definition policy for SELinux
ipaPolicyType: role
ipaSchemaFile: /var/lib/ipa/policy/SELinux_role.rng
ipaTrasformFile: /var/lib/ipa/policy/SELinux_role.xslt
```

In IPA v2 we will support only one role template per application but later we might need to add more. The suggested tree structure will be able to accomodate this without changes.

Policy and Policy Data

The roledata container will store the role policy built based on the template. Again we plan to support only one actual policy per application in IPA v2 but if in future we need to allow different role definition policies on per application basis the tree would be able to accommodate this without any changes.

Entry:

```
dn: ipaUniqueID=8ad16550-cd41-11dd-ad8b-0800200c9a66,cn=roledata,cn=SELinux,cn=roles,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
objectclass: ipaPolicy
ipaUniqueID: 8ad16550-cd41-11dd-ad8b-0800200c9a66
cn: SELinux role policy
description: SELinux role definition policy
ipaTemplateRef: 0e84fb20-cd40-11dd-ad8b-0800200c9a66,cn=templates,cn=SELinux,cn=roles,cn=policies,dc=...
ipaEnabledFlag: true
ipaLastChangeBy: uid=ssorce,cn=users,cn=accounts,dc=...
ipaLastChanged: 20090101000000
```

The policy data object will look like this. The mehanism that we once defined for the life cycle mangement of the configuration policies is applicable here too.

Entry:

```
dn: ipaUniquesID=cf915740-cd41-11dd-ad8b-0800200c9a66,ipaUniqueID=a5120010-ccff-11dd-ad8b-0800200c9a66,cn=roledata,cn=SELinux,cn=roles
objectclass: ipaPolicyData
ipaUniquesID: cf915740-cd41-11dd-ad8b-0800200c9a66
cn: SELinux role policy data
ipaPolicyBlob: <compressed policy data>
ipaLastChangeBy: uid=ssorce,cn=users,cn=accounts,dc=...
ipaLastChanged: 20090303000000
ipaPolicyState: applied
```

Entry:

```
dn: ipaUniquesID=db4a2260-cd41-11dd-ad8b-0800200c9a66,ipaUniqueID=a5120010-ccff-11dd-ad8b-0800200c9a66,cn=roledata,cn=SELinux,cn=roles
objectclass: ipaPolicyData
ipaUniquesID: db4a2260-cd41-11dd-ad8b-0800200c9a66
cn: SELinux role policy data
ipaPolicyBlob: <compressed policy data>
ipaLastChangeBy: uid=ssorce,cn=users,cn=accounts,dc=...
ipaLastChanged: 20090303000000
ipaPolicyState: superseded
```

Relations Container

As it was mentioned above the role relations will be stored in a special sort of container. This container will have two extra attributes:

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
   NAME 'ipaRoleType'
   DESC 'Type of the role'
   EQUALITY caseIgnoreMatch
   ORDERING caseIgnoreMatch
   SUBSTR caseIgnoreSubstringsMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   X-ORIGIN 'IPA v2')
```

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
   NAME 'ipaRoleOrder'
   DESC 'List of possible roles in priority order'
   EQUALITY caseIgnoreMatch
   ORDERING caseIgnoreMatch
   SUBSTR caseIgnoreSubstringsMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   X-ORIGIN 'IPA v2')
```

```
objectclass ( 2.16.840.1.113730.3.8.4.TBD
   NAME 'ipaRelationsContainer'
   SUP ipaContainer
   STRUCTURAL
   MUST ( ipaRoleType $ ipaRoleOrder )
   X-ORIGIN 'IPA v2' )
```

Relations

Definition

The last piece is the "relations". The relations require some additional design work. Each relation is a derivative from the ipaAssociation object with minor additions.

Entry:

```
dn: ipaUniquesID=cad284c0-cd43-11dd-ad8b-0800200c9a66,cn=relations,cn=SELinux,cn=roles,cn=policies,dc=...
objectclass: ipaRelation
objectclass: ipaAssociation
ipaUniquesID: cad284c0-cd43-11dd-ad8b-0800200c9a66
cn: Role mapping for NA engineers
memberUser: cn=Westford,cn=groups,cn=accounts,dc=...
memberUser: cn=Raleigh,cn=groups,cn=accounts,dc=...
memberHost: cn=Dev-machines,cn=computergroups,cn=accounts,dc=...
memberHost: cn=Dev-lab-machines,cn=computergroups,cn=accounts,dc=...
ipaRoleRef: 8ad16550-cd41-11dd-ad8b-0800200c9a66,cn=roledata,cn=SELinux,cn=roles,cn=policies,dc=...
ipaRoleName: user
ipaEnabledFlag: true
description: This role mapping defines that users in Westford and Raleigh user groups should be
             assigned "user" role when they access Dev and Dev-lab machines.
```

The description attribute above speaks for itself.

Attributes

The ipaRoleRef attribute is a DN of the role definition policy for the relation.

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaRoleRef'
  DESC 'DN of the role definition policy'
  SUP distinguishedName
  EQUALITY distinguishedNameMatch
  ORDERING distinguishedNameMatch
  SUBSTR distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  X-ORIGIN 'IPA v2' )
```

The ipaRoleName is an attribute that will hold specific role name the relation is pointing to. The attribute will be case insensitive to reduce amount of typos. It will be up to the policy engine to enforce uniqueness of the role names within the policy.

```
attributeType ( 2.16.840.1.113730.3.8.3.TBD
  NAME 'ipaRoleName'
  DESC 'Name of the role'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-ORIGIN 'IPA v2')
```

Both attributes are required. Without them the entry does not make sense.

Object Class

As it was mentioned the object class is derived from the ipaAssociation.

```
objectclass ( 2.16.840.1.113730.3.8.4.TBD
  NAME 'ipaRelation'
  SUP ipaAssociation
  STRUCTURAL
  MUST ( ipaRoleRef $ ipaRoleName )
  X-ORIGIN 'IPA v2' )
```

Action Execution

Application Container

We decided that the action will be a flavor of the configuration policy. There will be a predefined application named "Shell Actions" that will have a template. This template will have the XSLT and RNG files that will describe a predefined action specific format and how to execute actions.

Entry:

```
dn: cn=Shell Actions,cn=applications,cn=configs,cn=policies,dc=...
objectclass: nsContainer
objectclass: ipaContainer
cn: Shell Actions
description: Shell Actions - special application that holds templates for actions
```

Template

The template will look like this.

Entry:

```
dn: ipaUniqueID=8f0dab10-cd23-11dd-ad8b-0800200c9a66,cn=Shell Actions,cn=applications,cn=configs,cn=policies,dc=...
objectclass: ipaPolicyTemplate
ipaUniqueID: 8f0dab10-cd23-11dd-ad8b-0800200c9a66
cn: actions template
description: Standard actions template
ipaPolicyType: action
ipaSchemaFile: /var/lib/ipa/policy/ipaaction.rng
ipaTrasformFile: /var/lib/ipa/policy/empty.xslt
```

If we decide that we do not need the XSL template for actions at all the xsltFile can be omitted. It is allowed by the class definition. See above.

Policy and Policy Data

The actual action policies will be created under inside the policy groups in the same way as other configuration policies. They will look pretty similar to the configuration policies.

Entry:

```
dn: ipaUniqueID=a5120010-ccff-11dd-ad8b-0800200c9a66,ipaUniqueID=537e5f90-cc6f-11dd-ad8b-0800200c9a66,cn=policygroups,cn=configs,cn=po
objectclass: nsContainer
objectclass: ipaContainer
objectclass: ipaPolicy
ipaUniqueID: a5120010-ccff-11dd-ad8b-0800200c9a66
cn: SELinux check script
description: Script that needs to be run once at the boot time and check if the machine is compliant to the current SELinux policies.
ipaTemplateRef: ipaUniqueID=8f0dab10-cd23-11dd-ad8b-0800200c9a66,cn=SUDO,cn=applications,cn=configs,cn=policies,dc=...
ipaEnabledFlag: true
ipaLastChangeBy: uid=dpal,cn=users,cn=accounts,dc=...
ipaLastChanged: 20090102000000
```

It is assumed that the same policy management states and operations (and thus very similar UI) will be applicable to the management of the actions. The policy data objects will be created under the corresponding ipaPolicy containers. Here is the example of a pair of the data entries.

Entry:

```
dn: ipaUniquesID=881702e0-cd00-11dd-ad8b-0800200c9a66,ipaUniqueID=a5120010-ccff-11dd-ad8b-0800200c9a66,ipaUniqueID=537e5f90-cc6f-11dd-
objectlass: ipaPolicyData
ipaUniquesID: 881702e0-cd00-11dd-ad8b-0800200c9a66
cn: SELinux_script
ipaPolicyBlob: <compressed policy data>
ipaLastChangeBy: uid=dpal,cn=users,cn=accounts,dc=...
ipaLastChanged: 20090102000000
ipaPolicyState: applied
```

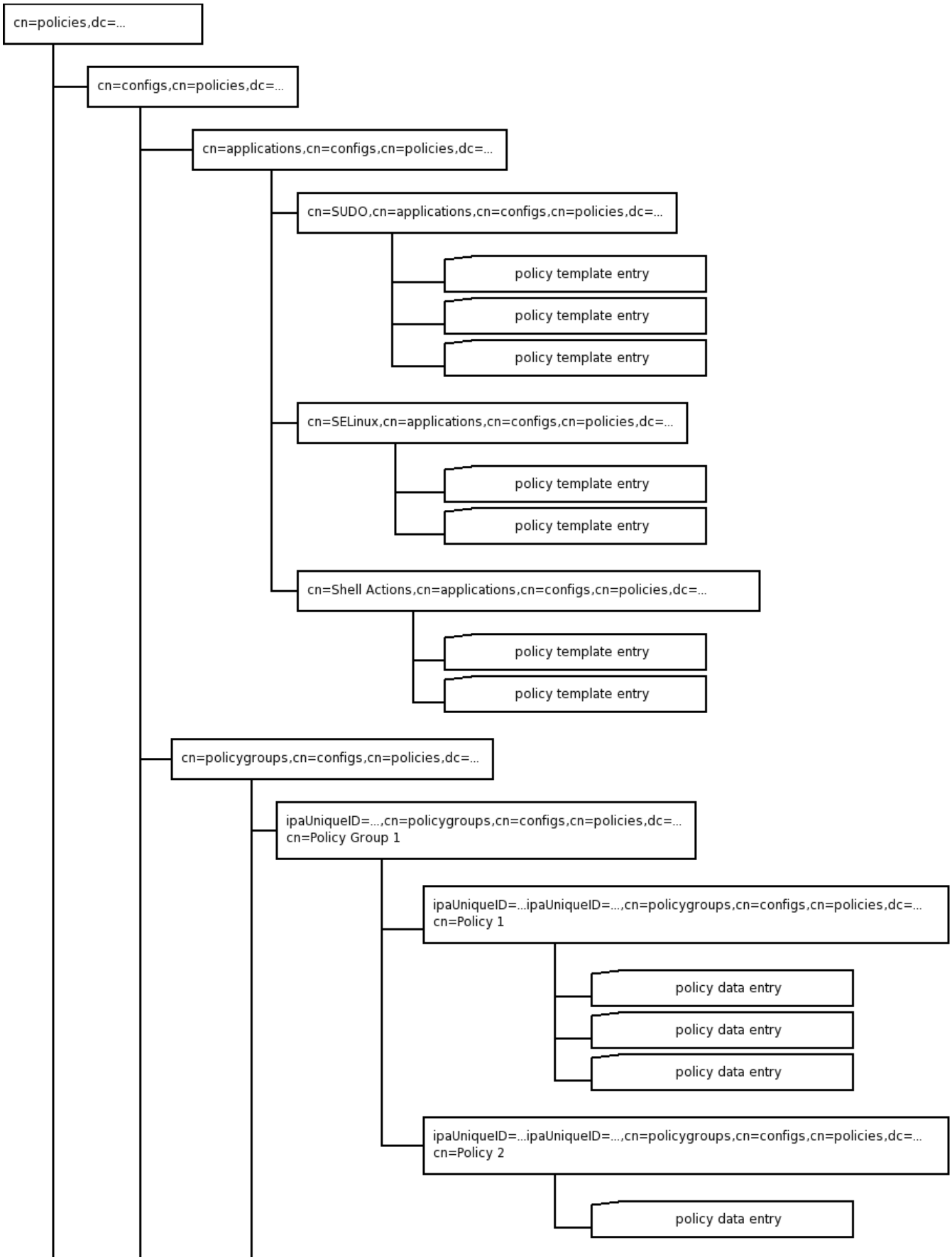
Entry:

```
dn: ipaUniquesID=771b22c0-cd04-11dd-ad8b-0800200c9a66,ipaUniqueID=a5120010-ccff-11dd-ad8b-0800200c9a66,ipaUniqueID=537e5f90-cc6f-11dd-
objectlass: ipaPolicyData
ipaUniquesID: 771b22c0-cd04-11dd-ad8b-0800200c9a66
cn: SELinux_script
ipaPolicyBlob: <compressed policy data>
ipaLastChangeBy: uid=dpal,cn=users,cn=accounts,dc=...
ipaLastChanged: 20090102000000
ipaPolicyState: superseded
```

Policy Link

There are no special links for the actions since they will be put into the policy groups. The policy groups are collections that are linked to the hosts.

Tree for Policy Objects


cn=policies,dc=...
cn=configs,cn=policies,dc=...
cn=applications,cn=configs,cn=policies,dc=...
cn=SUDO,cn=applications,cn=configs,cn=policies,dc=...
policy template entry
policy template entry
policy template entry
cn=SELinux,cn=applications,cn=configs,cn=policies,dc=...
policy template entry
policy template entry
cn=Shell Actions,cn=applications,cn=configs,cn=policies,dc=...
policy template entry
policy template entry
cn=policygroups,cn=configs,cn=policies,dc=...
ipaUniqueID=...,cn=policygroups,cn=configs,cn=policies,dc=...
cn=Policy Group 1
ipaUniqueID=...ipaUniqueID=...,cn=policygroups,cn=configs,cn=policies,dc=...
cn=Policy 1
policy data entry
policy data entry
policy data entry
ipaUniqueID=...ipaUniqueID=...,cn=policygroups,cn=configs,cn=policies,dc=...
cn=Policy 2
policy data entry

- This page was last modified on 10 January 2009, at 01:24.