AUTHORIZED DOCUMENTATION

Action Object Reference Guide

Novell®
Novell Storage Manager for eDirectory™

3.0.1

March 15, 2011

www.novell.com

Legal Notices

Condrey Corporation makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Condrey Corporation reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Condrey Corporation makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Condrey Corporation reserves the right to make changes to any and all parts of the software at any time, without obligation to notify any person or entity of such revisions or changes. See the Software EULA for full license and warranty information with regard to the Software.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export, or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. Condrey Corporation assumes no responsibility for your failure to obtain any necessary export approvals.

Condrey Corporation
125 The Parkway, Suite 500
Greenville, SC 29615
U.S.A.
www.condreycorp.com

Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list (http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

All third-party trademarks are the property of their respective owners.

Contents

About This Guide

This reference guide is written to provide network administrators comprehensive information for understanding and using the new Action Objects that were introduced in Novell Storage Manager 3.0.1 for eDirectory.

Audience

This manual is intended for network administrators who manage user and collaborative storage through Novell Storage Manager 3.0.1 for eDirectory.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comment feature at the bottom of each page of the online documentation, or go to www.novell.com/documentation/feedback.html and enter your comments there.

Documentation Updates

For the most recent version of the *Novell Storage Manager 3.0.1 for eDirectory Action Object Reference Guide*, visit the Novell Storage Manager Documentation Web site (http://www.novell.com/documentation/storagemanager3/).

Additional Documentation

For additional Novell Storage Manager documentation, see the following guides at the Novell Storage Manager Documentation Web site (http://www.novell.com/documentation/storagemanager3/):

- *Novell Storage Manager 3.0.1 for eDirectory Installation Guide*
- *Novell Storage Manager 3.0.1 for eDirectory Administration Guide*

Documentation Conventions

In this documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as Linux*, should use forward slashes as required by your software.

When a startup switch can be written with a forward slash for some platforms or a double hyphen for other platforms, the startup switch is presented with a forward slash. Users of platforms that require a double hyphen, such as Linux, should use double hyphens as required by your software.

1 Overview

Novell Storage Manager 3.0.1 for eDirectory includes new Action Objects to enable the automation of very distinct storage actions outside of what is practical with Novell Storage Manager policies.

1.1 What Are Action Objects?

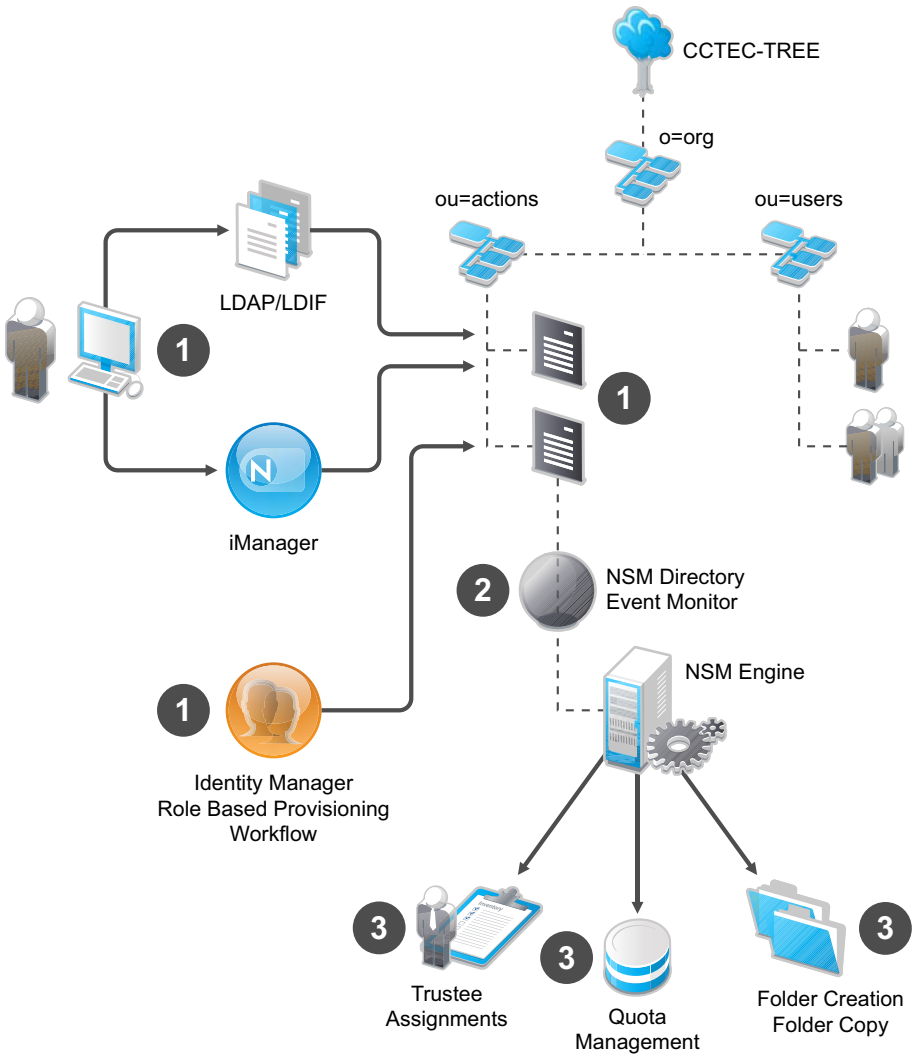
Action Objects provide a type of application programming interface (API) for use with Novell Storage Manager. Instead of writing custom code with vendor-supplied software libraries, the Action Object construct uses eDirectory objects and attributes as the communication method through which the API calls are made.

An Action Object is an object in eDirectory that represents a single file system event, such as creating a directory, or setting directory quota on a specified target path.

1.2 How do Action Objects work?

Action Objects are processed by the NSM Engine, and they provide many of the same benefits of policy-based actions in Novell Storage Manager, such as the state-machine architecture provided by Novell Storage Manager for action processing.

Figure 1-1 *How Action Objects Work.*



1. Action Objects can be created and managed by any application or process that can modify objects in eDirectory, such as ConsoleOne, iManager, Novell Identity Manager, and even LDAP / LDIF.
2. After the objects have been created and the trigger attribute has been set, the Event Monitor sees the trigger event and notifies the NSM Engine, placing an entry in the Engine's process queue.
3. The Engine processes the Action Object according to the rules written in the object's attributes. Depending on the action and rules, the Engine might set trustee rights on a folder, change the directory quota for a folder, or even create new folders and copy data. See Chapter 6, “Actions Reference,” on page 35.

2 Setting Up

Prior to creating Action Objects, a container must be created for hosting the Action Objects and then Novell Storage Manager must be enabled to process the Action Objects.

2.1 Creating an Action Object Container

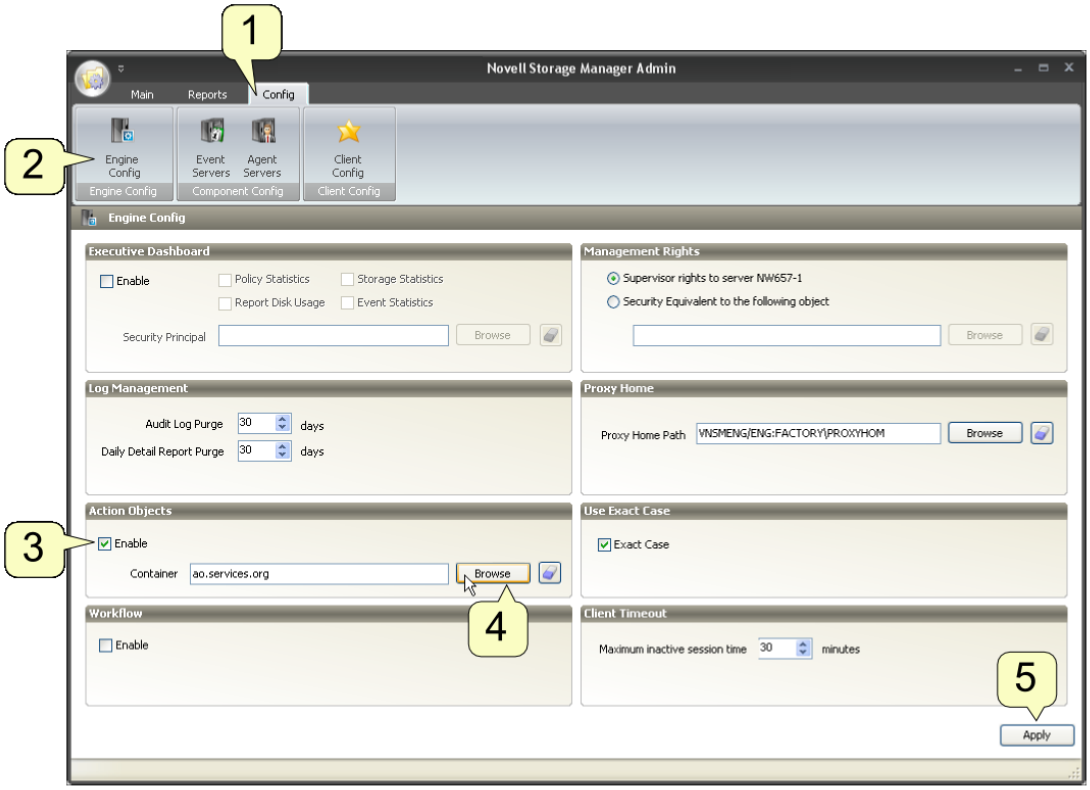
A single eDirectory container must be designated as the parent container in which Action Objects can be created. By applying appropriate access controls, an administrator can control who is allowed to create Action Objects in the specified container. The NSM Engine only processes Action Objects that reside in the designated container or any of its subcontainers.

A local replica of the partition containing the Action Object container is recommended for the server hosting the NSM Engine if large quantities of Action Objects will be created and processed.

2.2 Enabling Action Object Processing

To enable processing of Action Objects by Novell Storage Manager, log in to NSMAdmin, then perform the following steps:

Figure 2-1 *Enabling Action Object Processing Through NSMAdmin.*



1. Click the *Configure* tab.
2. Click the *Engine Config* button in the toolbar.
3. Click the *Action Objects* tab.
4. Select the *Enable* check box.
5. Click *Browse* and select the Action Objects container.
6. Click *Apply* to save the changes.

NOTE: The only Action Objects eligible for processing are those located in the specified container or any of its subcontainers.

---

3 Using Action Objects

3.1 General Guidelines

Action Objects can be created by any process, application, or API that can manage extended object classes and attributes in eDirectory. Examples of such applications include iManager, ConsoleOne, and Novell Identity Manager.

Follow these guidelines when creating Action Objects:

- Create Action Objects in the specified Action Object Container or one of its subcontainers. (See Section 2.1, “Creating an Action Object Container,” on page 11.)
- Set the trigger attribute last. For example, with a SetQuota Action Object, be sure to assign values to the cccFSFactoryActionPath1, cccFSFactoryActionOption, and cccFSFactoryActionOperation attributes before setting the cccFSFactoryActionTrigger attribute.
- It is acceptable to create the Action Object with all values at one time as an atomic operation. This is typically the case when using LDAP with a single LDIF import file. However, if timing or replication issues produce unreliable results, create the Action Object with all necessary attributes first, then add the cccFSFactoryTrigger attribute last in a separate modify action.

3.2 Creating Action Objects

1. Create the Action Object itself in the designated Action Object Container.
2. Add the appropriate attributes needed for the operation.

   At a minimum, you need to specify the cccFSFactoryActionOperation attribute, which determines the type of Action Object. In most cases, you also need to specify the cccFSFactoryActionPath1 attribute. For details on the attributes and values needed for each action type, see Chapter 6, “Actions Reference,” on page 35.
3. Add any additional attributes needed for Action Object linking, scheduled execution time, or system cleanup.
4. Add the cccFSFactoryActionTrigger attribute with a value of Ready to notify the Novell Storage Manager Event Monitors and Engine that the Action Object is ready for processing.

For examples of how to create an Action Object using iManager, LDAP, and Novell Identity Manager, see Chapter 5, “Usage Examples,” on page 21.

3.3 Scheduling Action Objects

Action Objects can be configured for processing at a specific date and time. To schedule a specific Action Object, simply fill in the appropriate value for the Action Object's cccFSFactoryActionExecuteTime attribute prior to setting the trigger.

See Section 6.3.1, “Execute Time,” on page 53 for details on how to do this.

3.4 Linking Action Objects

Action Objects can be linked in series to allow dependent actions to occur first. When linking Action Objects, either the NSM Engine or an external process can be specified for triggering the subsequent action.

For details on how to perform linking, see Section 6.3.2, “Link Next,” on page 53.

3.5 Reusing Action Objects

An Action Object can be reused as needed.

1. Verify that the Action Object to be reused is not currently processing. Verify this by examining the cccFSFactoryActionResult attribute for a Success or error message, or by examining the cccFSFactoryActionStatus attribute.
2. If the cccFSFactoryActionTrigger attribute has a value, clear or delete the cccFSFactoryActionTrigger attribute.

   If you are using iManager or ConsoleOne, be sure to apply the delete or modification of the attribute before continuing. Failure to do so prevents the event trigger from occurring.

   You might also want to clear other values such as the cccFSFactoryActionResult and the cccFSFactoryActionStatus.
3. Set the cccFSFactoryActionTrigger attribute to the value Ready to reissue the Action Object.

3.6 Cleaning Up Action Objects

After the Engine has completed processing an Action Object, that object can be cleaned up (deleted from Directory Services) or left behind for processing by an external application.

For details on how to set the cleanup attribute, see Section 6.3.3, “Cleanup,” on page 55.

3.7 Action Object Security

The NSM Engine processes the directives in Action Objects by using the Novell Storage Manager proxy account that performs file system and directory service operations for Novell Storage Manager. The account was created during installation and configuration of Novell Storage Manager 3.0.1 for eDirectory.

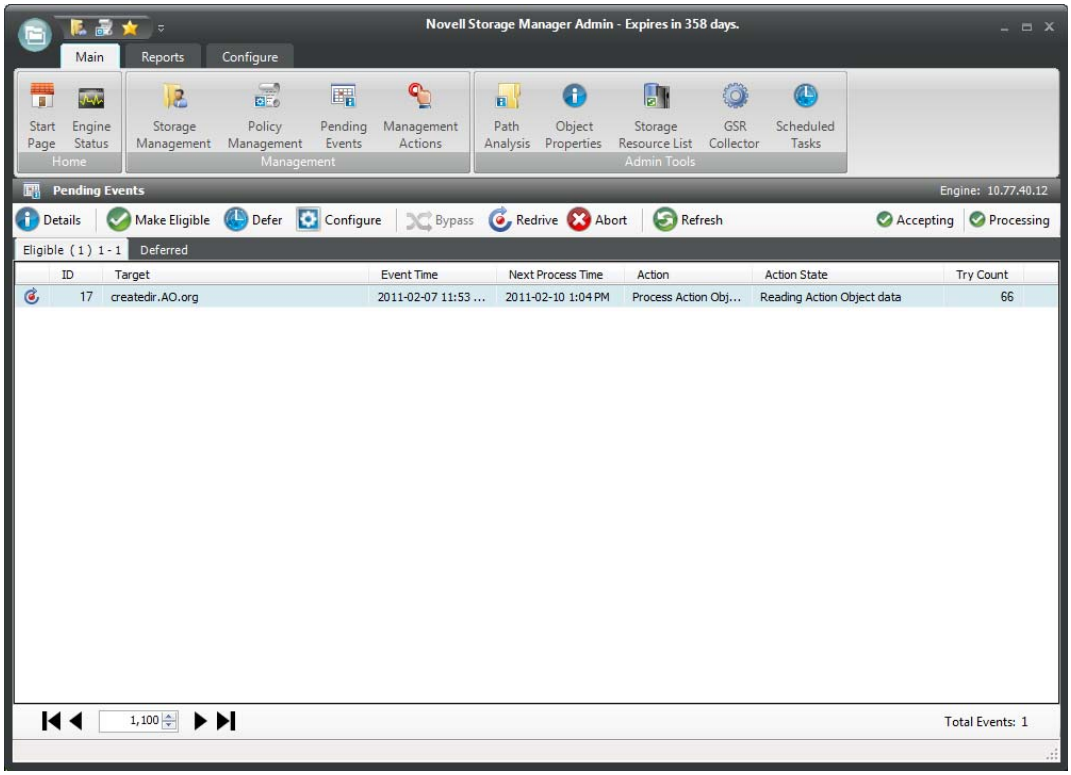
In order to restrict who can create valid Action Objects, Novell Storage Manager 3.0.1 for eDirectory requires that a container in eDirectory be specified for Action Object processing. The designated Action Object container and its subcontainers are the only containers in eDirectory from which the NSM Engine processes Action Objects. The Action Object Container can be secured by using regular access controls for eDirectory, limiting who is allowed to create and modify Action Objects.

3.8 Managing Action Object Events

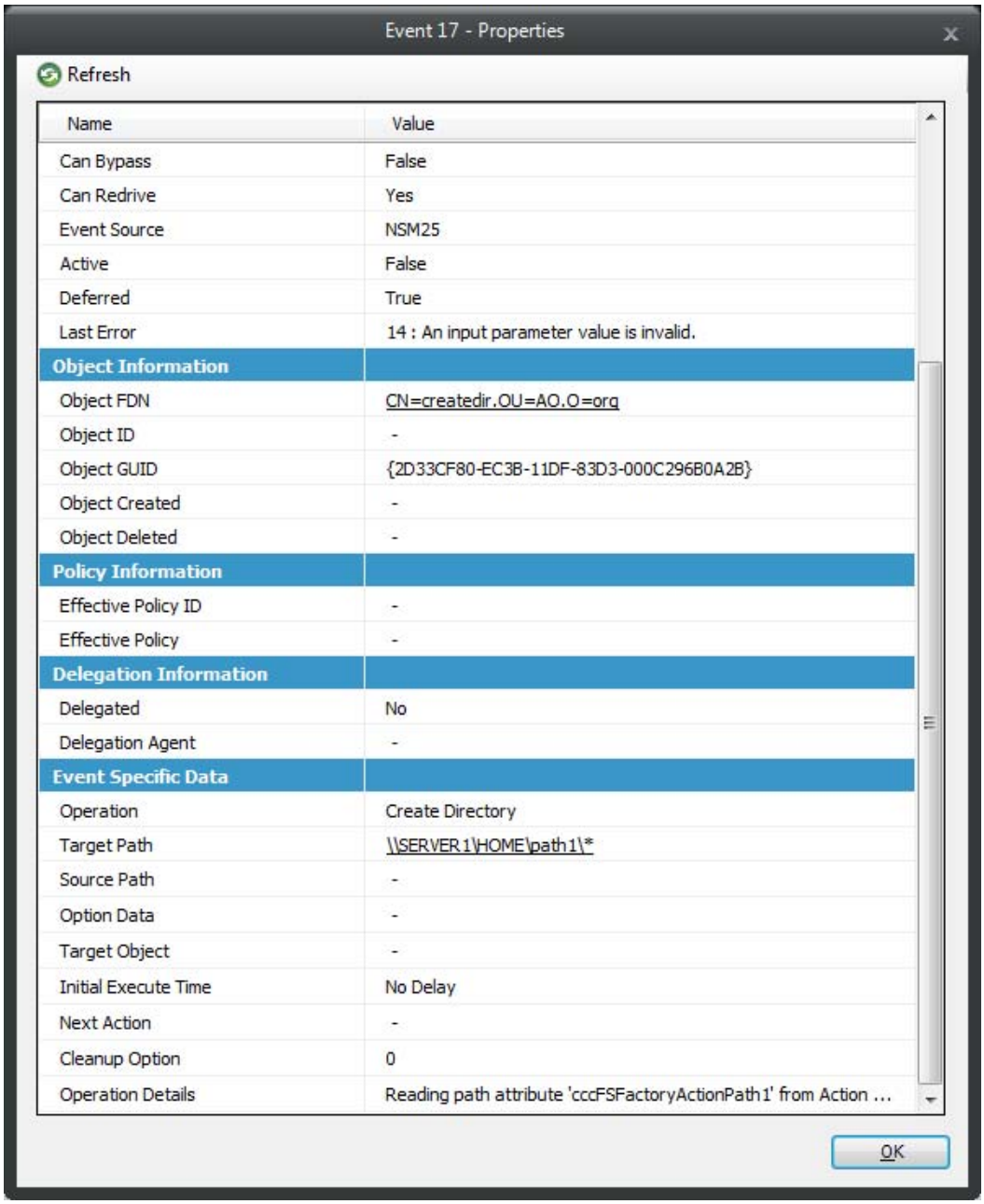
Once the NSM Engine starts processing an Action Object, the events generated by the Action Object can be viewed and managed in the NSMAdmin Pending Events panel.

3.8.1 Viewing Action Object Event Details

1. In NSMAdmin, click the *Main* tab.
2. From the toolbar, click *Pending Events*.



3. Double-click the event corresponding to the Action Object.



Details that are Action Object-specific are shown under the Event Specific Data header. General information, such as the Last Error and Can Redrive are available under the General header.

3.8.2 Aborting an Action Object Event

1. In NSMAdmin, click the *Main* tab.
2. From the toolbar, click *Pending Events*.
3. Select the event to abort.

4 Click the *Abort* button.

5 Click *Yes* in the confirmation dialog box to abort the event.

4 Processing States

As an Action Object is processed by the Novell Storage Manager Engine, it might be in one of several states. As an object transitions to each state, its cccFSFactoryActionStatus attribute is updated with the corresponding state label.

4.1 State Order

The states are processed in the following order:

1. Start
2. Wait for Directory Synchronization
3. Verify Trigger
4. Read Data
5. Wait on Ready
6. Process
7. In Progress
8. Set Result
9. Link
10. Cleanup
11. Complete Pending
12. Complete

4.2 State Descriptions

The various states for Action Object processing are described as follows:

- **START:** This is the initial state before any processing takes place.
- **WAIT_FOR_DIRECTORY_SYNCHRONIZATION:** In this state, the Engine is waiting to see the Action Object itself in eDirectory. Because an Action Object node is placed in the process queue only after that object’s trigger has been set and an Event Monitor has reported it, an assumption is made that the Action Object persists in eDirectory until the Engine can see it. If an Action Object is deleted from eDirectory after the event notification and subsequent addition to the process queue, but before processing this state, the action remains in a pending state indefinitely.
- **VERIFY_TRIGGER:** In this state, the Engine verifies that the trigger attribute has the proper value of Ready. If the attribute is not yet available, the Engine re-reads the Action Object periodically to see if the attribute is available. Once the attribute is available, if it does not contain the proper trigger value, the event is placed in the COMPLETE state. Otherwise, processing continues to the READ_DATA state.

- **READ_DATA:** In this state, the Engine reads in the values of all the Action Object attributes that apply for the specific operation. Basic validation is performed on attribute values, and any error conditions in this level cause the event to be redrivable.
- **WAIT_ON_READY:** This state is a placeholder for Action Object events that have an Execute Time set that defers initial processing of the event.
- **PROCESS:** In this state, the Action Object itself is processed by calling the appropriate action function as determined by the Operation set in the Action Object.
- **IN_PROGRESS:** Depending on the operation, some Action Objects might run asynchronous processes, such as a delegated operation like CopyDir. In these cases, the Engine must check the state of the delegated operation for this action on each process queue iteration until the operation is complete.

  For all other operations that were synchronously called during the PROCESS state, processing simply continues on to the next state.
- **SET_RESULT:** In this state, the Engine sets the cccFSFactoryActionResult attribute of the Action Object in eDirectory. Because later states require the presence of this attribute, and because external systems such as Novell Identity Manager might also be interacting with the Action Object, processing does not continue past this state until the attribute can be set.
- **LINK:** In this state, the Engine attempts to process the cccFSFactoryActionLinkNext attribute of the Action Object in eDirectory. If an action has completed successfully, and the LinkNext attribute was set, the Engine sets the trigger for the specified action to the value “ready”.

  If the cccFSFactoryLinkNext attribute is not set, or if the action itself did not complete successfully, processing continues with the next state.
- **CLEANUP:** In this state, the Engine processes that value read from the cccFSFactoryActionCleanup attribute of the Action Object to determine whether it should remove the Action Object from eDirectory at the end of processing. If the cleanup attribute is present and set to the value System, or if it is set to the value OnSuccess and the operation is successful, it attempts to delete the associated Action Object from eDirectory. In all cases of failure, processing simply continues to the next state.
- **COMPLETE_PENDING:** In this state, the Action Object trigger is cleared for Action Objects that have not been cleaned up.
- **COMPLETE:** After this state is reached, the Action Object is eligible for removal from the process queue.

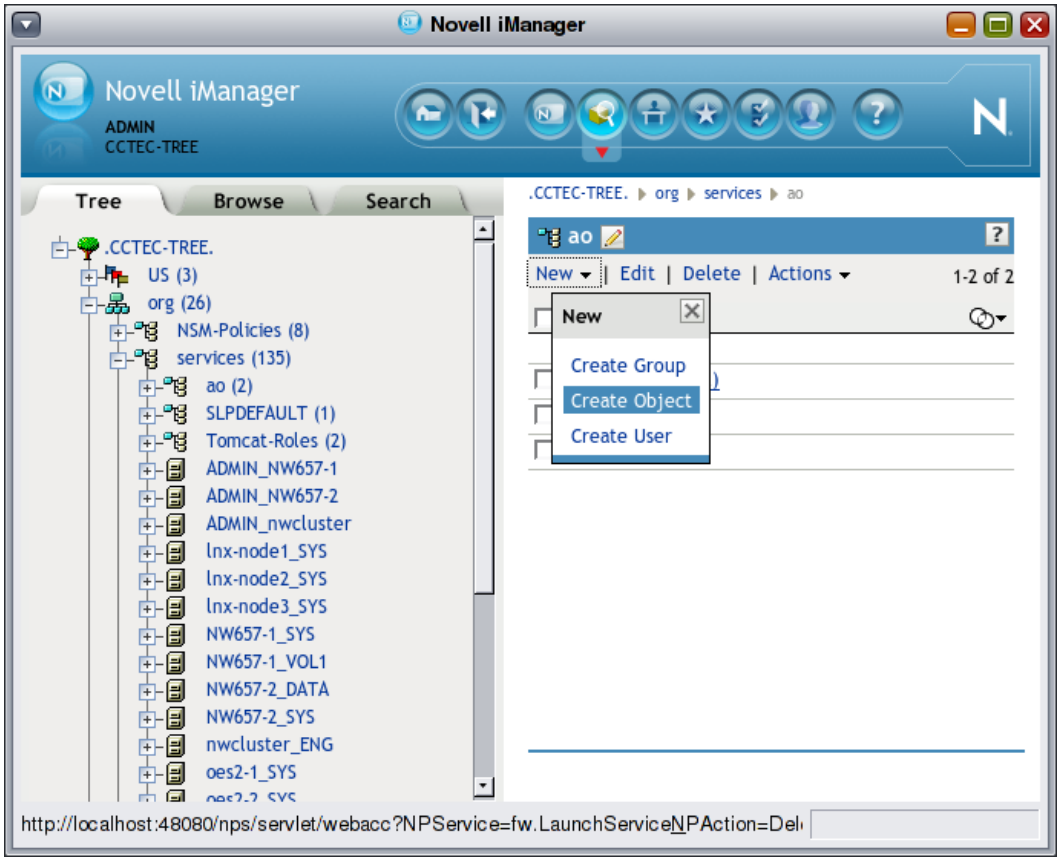
Usage Examples

5

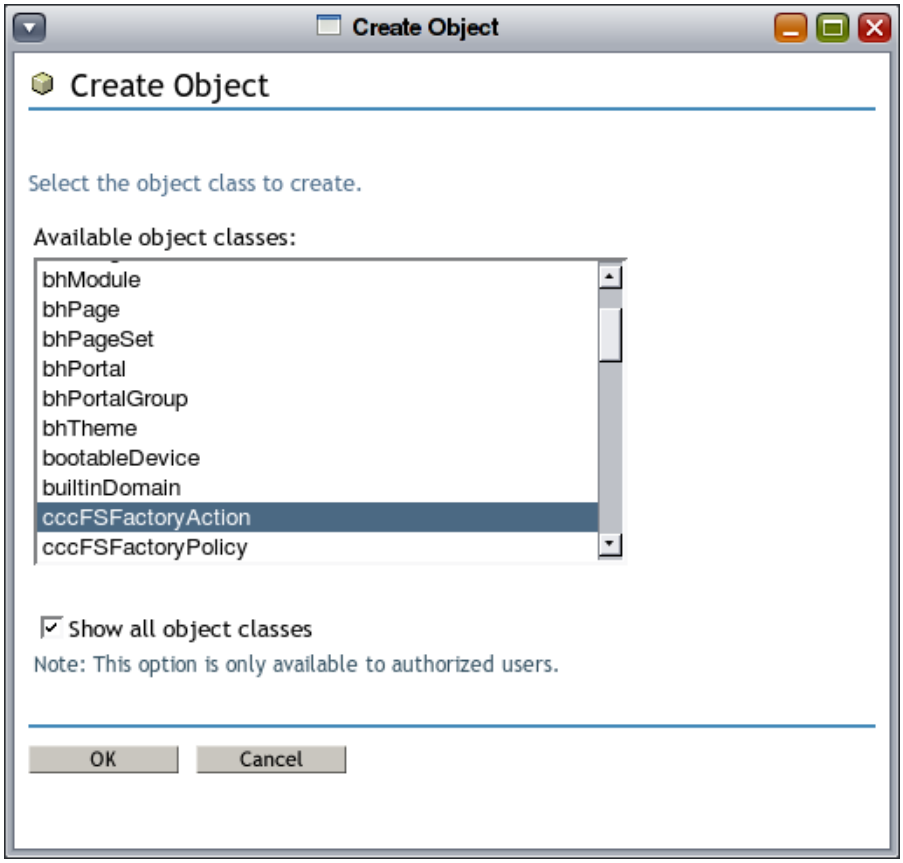
5.1 iManager

The following example illustrates how to create a CreateDir Action Object by using iManager.

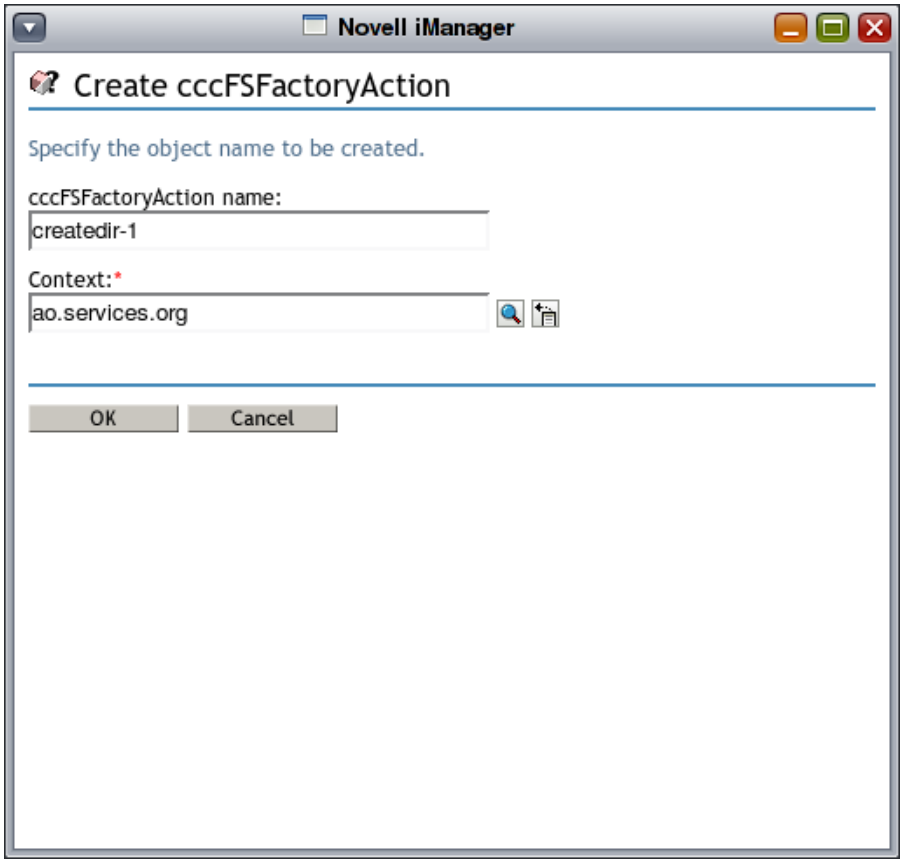
1. Start iManager and select the *View Object* button in the menu at the top.
2. Browse to and select the configured Action Object container in the tree view on the left.



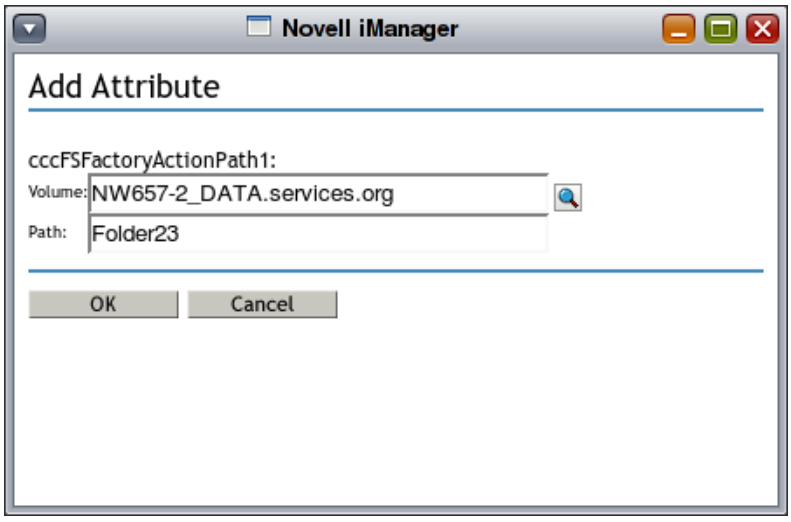
3. Select *New* > *Create Object*.



4 Select *Show all object classes*, browse for and select cccFSFactoryAction, then click *OK*.

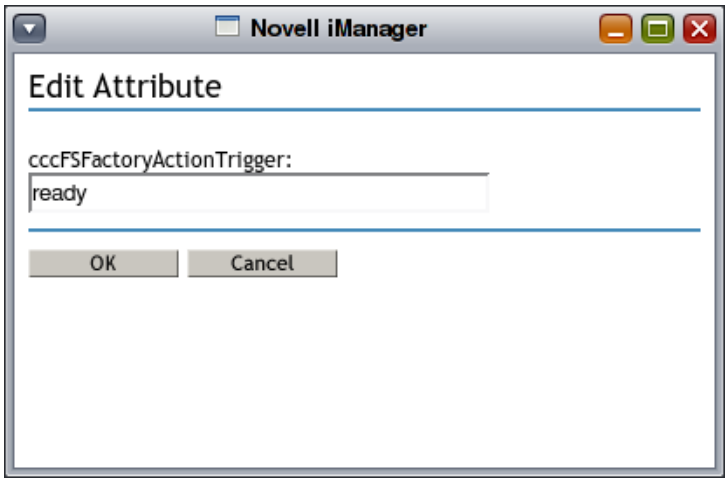


5 Specify a name for the new Action Object, then click *OK*.

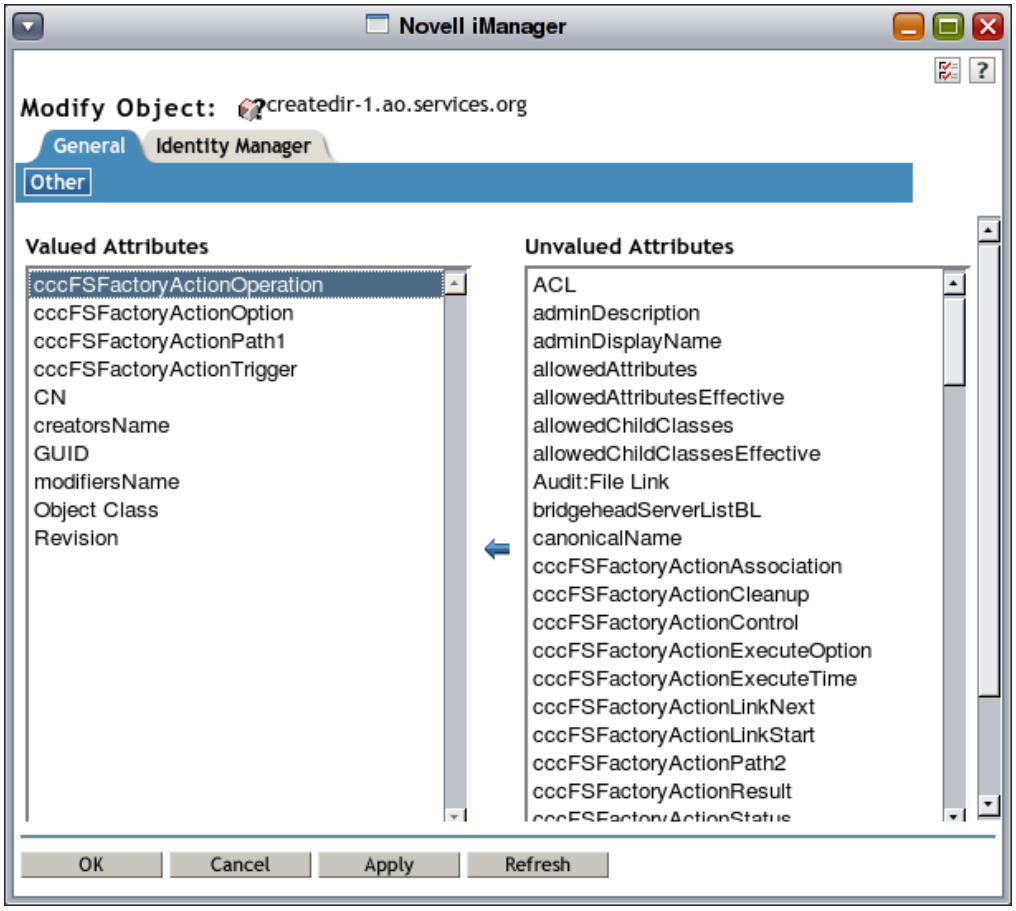


6 For a CreateDir operation, provide values for the following attributes:

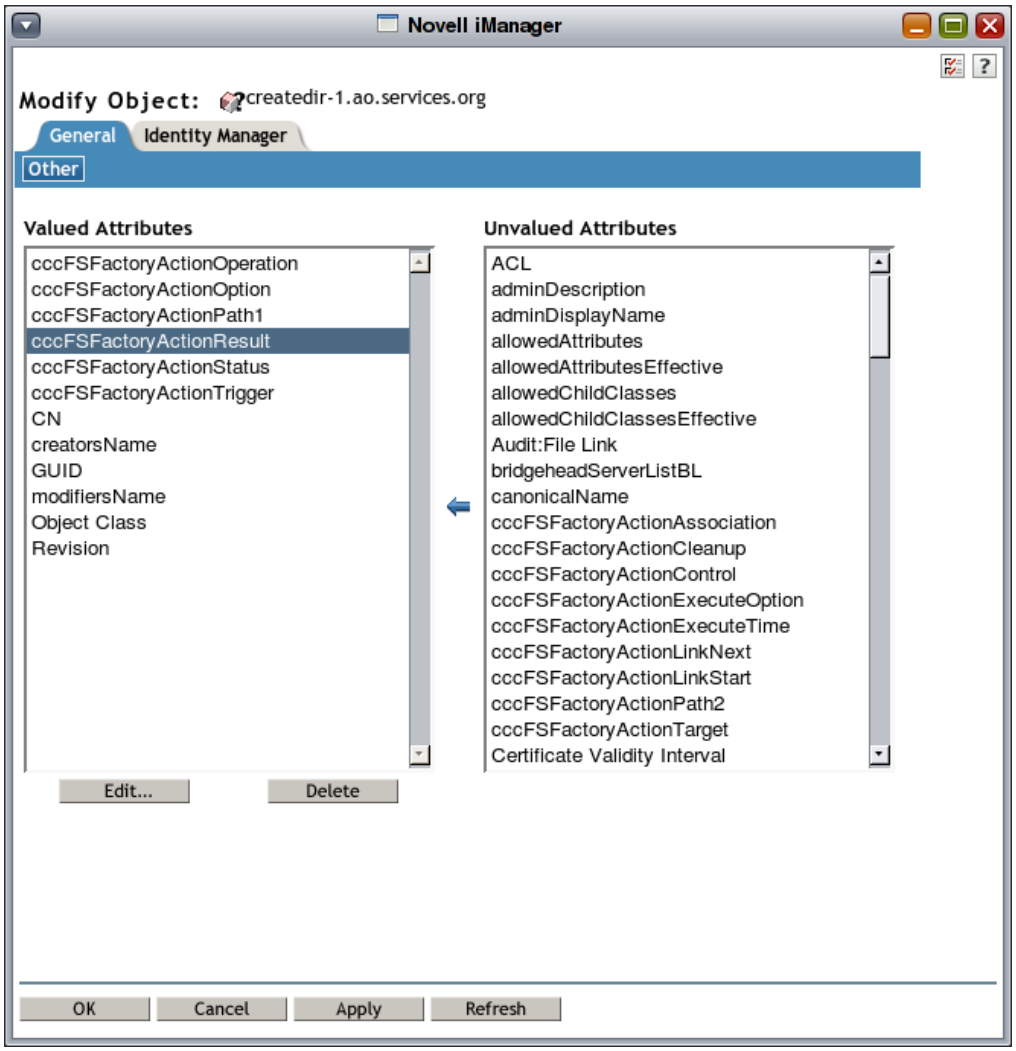
- cccFSFactoryActionOperation
- cccFSFactoryActionOption
- cccFSFactoryActionPath1



7 After all of the appropriate values for the action have been filled in, add the cccFSFactoryActionTrigger attribute, set it to a value of Ready, then click *OK*.



8 The object should now be complete. Watch the appropriate Event Monitor and Engine console screens for indication that the Action Object has been processed.



9 After the Action Object has been processed, the cccFSFactoryActionResult and cccFSFactoryActionStatus attributes are filled in with values indicating success or failure for the processed operation.

5.2 LDAP

This example shows how to create a SetQuota Action Object using LDAP commands with an LDIF (LDAP Directory Interchange Format) input file. When it is processed, the Novell Storage Manager Engine sets a 7 MB quota on the target path SERVER-A\DATA:users\bsmith.

Example: Setting the Quota

```
#Example1 LDIF for SetQuota Action Object
version: 1
dn: cn=action-1,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVERA_DATA,ou=resources,
 o=org#0#\users\bsmith
cccFSFactoryActionOption: <Option><SubCmd>1</SubCmd>
 <Quantity>7<Quantity></Option>
cccFSFactoryActionOperation: SetQuota
cccFSFactoryActionTrigger: Ready
```

Example: Adding 2 MB to the Existing Quota

```
#Example2 LDIF for SetQuota Action Object
version: 1
dn: cn=action-2,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVERA_DATA,ou=resources,
 o=org#0#\users\bsmith
cccFSFactoryActionOption: <Option><SubCmd>2</SubCmd><Quanti
 ty>2</Quantity></Option>
cccFSFactoryActionOperation: SetQuota
cccFSFactoryActionTrigger: Ready
```

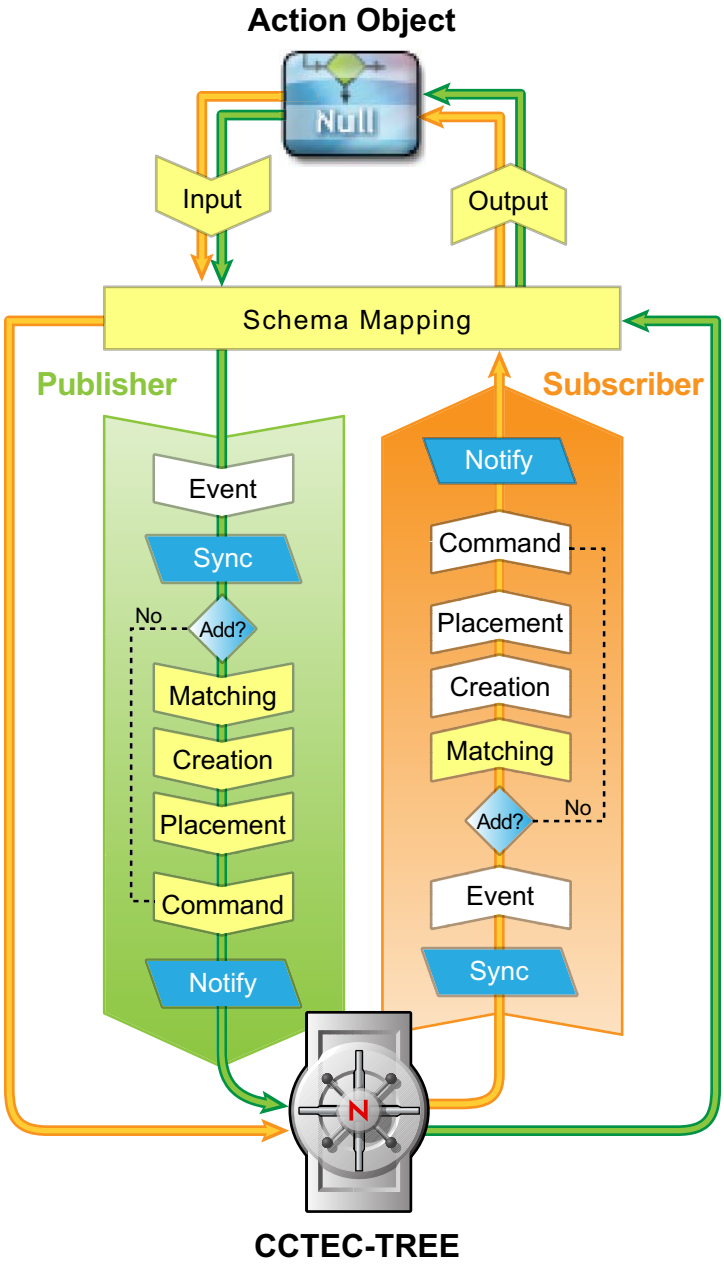
5.3 Novell Identity Manager

The following example illustrates how to use Novell Identity Manager to create a SetQuota Action Object when a user's title is changed to some value containing the text “Admin” or “admin.” In addition, it shows how to clean up Action Objects that have processed successfully.

5.3.1 Driver Overview

The example driver uses the Generic Null driver from Novell Identity Manager 3.6.

Figure 5-1 *Driver Overview*

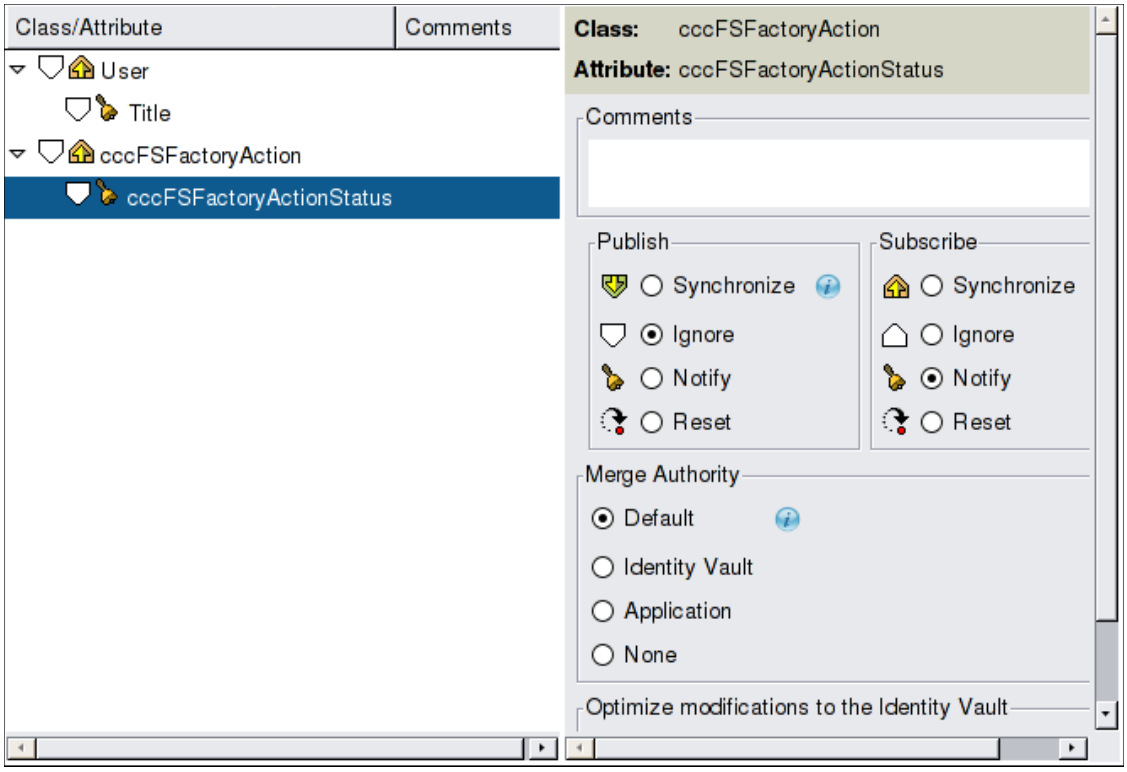


Because you are using the Null driver, only the Filter and Event Policy Sets are used.

5.3.2 Driver Filter

For the Filter in this example, you need the following:

Figure 5-2 *Driver Filter.*



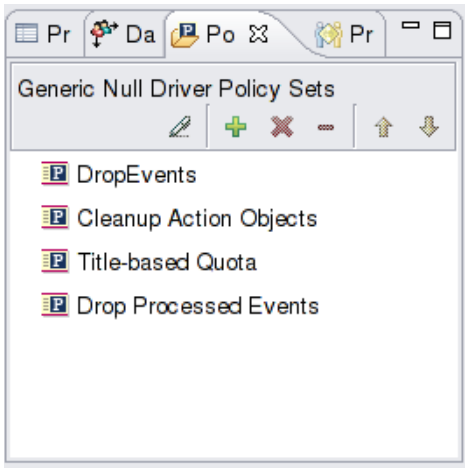
The User class and its associated Title attribute are needed to notify the driver of the user's change in title.

The cccFSFactoryAction class and its associated cccFSFactoryActionStatus attribute are used when determining whether to clean up (delete) the Action Object after processing is complete.

5.3.3 Event Policy Set

The following Event Transformations are set up:

Figure 5-3 *Event Transform Policy Set.*



The policies shown perform the following actions:

The DropEvents policy simply limits the scope of the driver to Modify events.

The Cleanup Action Objects policy is used to delete Action Objects that have successfully completed their assigned actions.

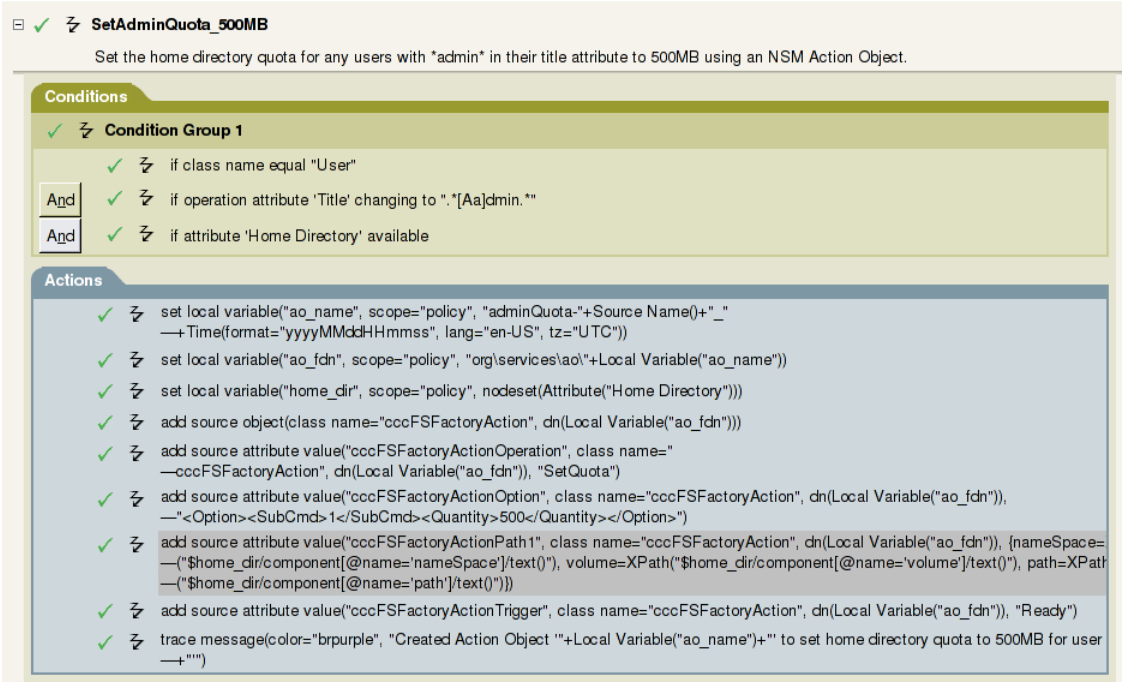
The Title-based Quota policy creates SetQuota Action Objects for users whose Title attribute is changing to include “Admin” or “admin” as part of the title.

The Drop Processed Events policy simply drops the current operation, because no other processing is needed in the driver.

Title-based Quota Policy

The primary policy is the Title-based Quota policy. This policy has the following conditions and actions:

Figure 5-4 *Set Admin Quota Rule.*



The policy performs the following actions:

1. Limits the scope of the processing to user objects whose title is changing to a value that includes “Admin” or “admin” as part of the string.
2. Only processes user objects that have a current Home Directory attribute set.
3. Sets up local variables to hold the new Action Object name and FDN.
4. Creates a new Action Object based on the FDN specified in the local variables.
5. Sets the attributes of the new Action Object based on what is required for a SetQuota action:
    a. Sets cccFSFactoryActionOperation to the value SetQuota.

b. Sets cccFSFactoryActionOption to the XML string:`<Option><SubCmd>1</SubCmd><Quantity>500</Quantity></Option>`

NOTE: The value of 1 for `<SubCmd>` is used to set the quota to the value specified by `<Quantity>`.

c. Sets cccFSFactoryActionPath1 to the Home Directory attribute of the current user in the operation. This requires manipulation of a structured value as opposed to a simple string.

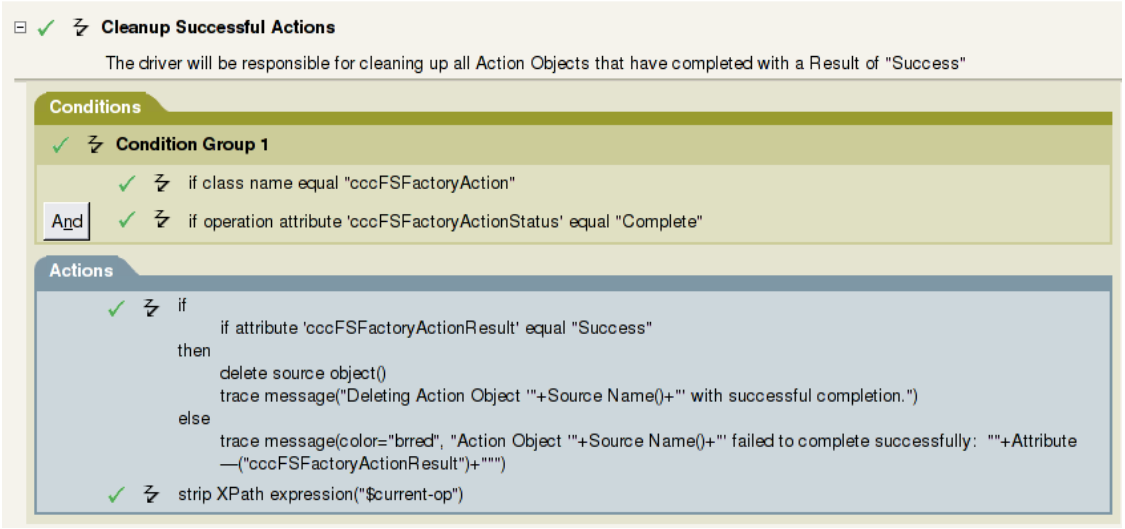
d. Sets cccFSFactoryActionTrigger to the value Ready so that the Action Object can be processed immediately.

6. Because the creation of the Action Object is a direct write back to eDirectory, a trace message indicating the creation of the Action Object was added for clarity in driver traces.

Cleanup Action Objects Policy

The Cleanup Action Objects policy is a helper policy that allows Novell Identity Manager to delete any Action Objects that have successfully completed. In addition, it adds trace messages to indicate success of the Action Object cleanup, or displays the error message from the Action Object's operation if the Set Quota operation failed.

Figure 5-5 *Cleanup Action Objects Rule.*



NOTE: The Action Object should be evaluated for deletion only after a status of Complete has been posted to the Action Object. Deleting the Action Object prior to this might cause the associated pending event in the Novell Storage Manager Engine event queue to stay pending.

5.3.4 Policy Scripts

Title-based Quota Policy Script

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE policy PUBLIC "policy-builder-dtd">
<policy>
  <rule>
    <description>SetAdminQuota_500MB</description>
    <comment xml:space="preserve">Set the home directory quota for any users
with *admin* in their title attribute to 500MB using an NSM Action Object.</
comment>
    <conditions>
      <and>
        <if-class-name mode="nocase" op="equal">User</if-class-name>
        <if-op-attr mode="regex" name="Title" op="changing-to">.*[Aa]dmin.*</
if-op-attr>
        <if-attr name="Home Directory" op="available"/>
      </and>
    </conditions>
    <actions>
      <do-set-local-variable name="ao_name" scope="policy">
        <arg-string>
          <token-text xml:space="preserve">adminQuota-</token-text>
          <token-src-name/>
          <token-text xml:space="preserve">_</token-text>
          <token-time format="yyyyMMddHHmmss" lang="en-US" tz="UTC"/>
        </arg-string>
      </do-set-local-variable>
      <do-set-local-variable name="ao_fdn" scope="policy">
        <arg-string>
          <token-text xml:space="preserve">org\services\ao\</token-text>
          <token-local-variable name="ao_name"/>
        </arg-string>
      </do-set-local-variable>
      <do-set-local-variable name="home_dir" scope="policy">
        <arg-node-set>
          <token-attr name="Home Directory"/>
        </arg-node-set>
      </do-set-local-variable>
      <do-add-src-object class-name="cccFSFactoryAction">
        <arg-dn>
          <token-local-variable name="ao_fdn"/>
        </arg-dn>
      </do-add-src-object>
      <do-add-src-attr-value class-name="cccFSFactoryAction"
name="cccFSFactoryActionOperation">
        <arg-dn>
          <token-local-variable name="ao_fdn"/>
        </arg-dn>
        <arg-value>
          <token-text xml:space="preserve">SetQuota</token-text>
        </arg-value>
      </do-add-src-attr-value>
      <do-add-src-attr-value class-name="cccFSFactoryAction"
name="cccFSFactoryActionOption">
```

```
        <arg-dn>
          <token-local-variable name="ao_fdn"/>
        </arg-dn>
        <arg-value>
          <token-text xml:space="preserve">&lt;Option>&lt;SubCmd>1&lt;/
SubCmd>&lt;Quantity>500&lt;/Quantity>&lt;/Option></token-text>
        </arg-value>
      </do-add-src-attr-value>
      <do-add-src-attr-value class-name="cccFSFactoryAction"
name="cccFSFactoryActionPath1">
        <arg-dn>
          <token-local-variable name="ao_fdn"/>
        </arg-dn>
        <arg-value type="structured">
          <arg-component name="nameSpace">
            <token-xpath expression="$home_dir/component[@name='nameSpace']/
text()"/>
          </arg-component>
          <arg-component name="volume">
            <token-xpath expression="$home_dir/component[@name='volume']/
text()"/>
          </arg-component>
          <arg-component name="path">
           <token-xpath expression="$home_dir/component[@name='path']/text()"/
>
          </arg-component>
        </arg-value>
      </do-add-src-attr-value>
      <do-add-src-attr-value class-name="cccFSFactoryAction"
name="cccFSFactoryActionTrigger">
        <arg-dn>
          <token-local-variable name="ao_fdn"/>
        </arg-dn>
        <arg-value>
          <token-text xml:space="preserve">Ready</token-text>
        </arg-value>
      </do-add-src-attr-value>
      <do-trace-message color="brpurple">
        <arg-string>
         <token-text xml:space="preserve">Created Action Object '</token-text>
          <token-local-variable name="ao_name"/>
          <token-text xml:space="preserve">' to set home directory quota to
500MB for user '</token-text>
          <token-src-name/>
          <token-text xml:space="preserve">'</token-text>
        </arg-string>
      </do-trace-message>
    </actions>
  </rule>
</policy>
```

Cleanup Action Objects Policy Script

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE policy PUBLIC "policy-builder-dtd">
<policy>
  <rule>
    <description>Cleanup Successful Actions</description>
    <comment xml:space="preserve">The driver will be responsible for cleaning
up all Action Objects that have completed with a Result of "Success"</comment>
    <conditions>
      <and>
        <if-class-name mode="nocase" op="equal">cccFSFactoryAction</if-class-
name>
        <if-op-attr mode="nocase" name="cccFSFactoryActionStatus"
op="equal">Complete</if-op-attr>
      </and>
    </conditions>
    <actions>
      <do-if>
        <arg-conditions>
          <and>
            <if-attr mode="nocase" name="cccFSFactoryActionResult"
op="equal">Success</if-attr>
          </and>
        </arg-conditions>
        <arg-actions>
          <do-delete-src-object/>
          <do-trace-message>
            <arg-string>
             <token-text xml:space="preserve">Deleting Action Object '</token-
text>
              <token-src-name/>
              <token-text xml:space="preserve">' with successful completion.</
token-text>
            </arg-string>
          </do-trace-message>
        </arg-actions>
        <arg-actions>
          <do-trace-message color="brred">
            <arg-string>
              <token-text xml:space="preserve">Action Object '</token-text>
              <token-src-name/>
              <token-text xml:space="preserve">' failed to complete
successfully:  "</token-text>
              <token-attr name="cccFSFactoryActionResult"/>
              <token-text xml:space="preserve">"</token-text>
            </arg-string>
          </do-trace-message>
        </arg-actions>
      </do-if>
      <do-strip-xpath expression="$current-op"/>
    </actions>
  </rule>
</policy>
```

Actions Reference

6

6.1 Actions

Individual actions are detailed below.

6.1.1 AssignPolicy

Description

The AssignPolicy operation assigns a named policy to a given object in eDirectory.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	AssignPolicy	
cccFSFactoryActionOption (IN)	Policy Name	Specifies unique policy name.
cccFSFactoryActionTarget (IN)	FDN of policy assignment	This object will be added to the list of associations for the named policy.
cccFSFactoryActionResult (OUT)	Success	
	<Error Message>	

Return Values

The cccFSFactoryActionResult attribute is set to Success if successful or an error message otherwise.

Notes

- The trustee FDN specified for the cccFSFactoryActionTarget attribute may be entered in either typeful or typeless FDN format such as `bob.hq.org` or `CN=bob.OU=hq.O=org`, or in LDAP format such as `cn=bob,ou=hq,o=org`.
- The policy name must match an existing policy in the configured system, however the policy name does not need to be a case exact match.
- Policies will only be successfully assigned if the policy type is allowed to be assigned to the specified object type. Allowed assignment types are as follows:

Policy Type	Object Type (Class)	Notes
User Policy	◆ Container: Organization, Organizational Unit, Domain, Locality ◆ Group ◆ User	
Collaborative Group Policy	◆ Container: Organization, Organizational Unit, Domain, Locality ◆ Group	Not valid for user objects
Collaborative Container Policy	◆ Container: Organization, Organizational Unit, Domain, Locality	Not valid for user or group objects
User Auxiliary Policy	Not valid	Currently, auxiliary policies may only be assigned to primary policies, not to objects in the directory service.

Example

The following LDIF file shows how to assign policy UserPolicy1 to the container hq.org.

```
#Example LDIF for AssignPolicy Action Object
version: 1
dn: cn=assignpolicy,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionOperation: AssignPolicy
cccFSFactoryActionOption: UserPolicy1
cccFSFactoryActionTarget: hq.org
cccFSFactoryActionTrigger: Ready
```

6.1.2 ClearTrustee

Description

The ClearTrustee operation removes the specified trustee from the designated path.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	ClearTrustee	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath	Path of the directory in SYN_PATH format.
cccFSFactoryActionTarget (IN)	FDN of the trustee to remove	This user is removed as a trustee of Path1.

Attribute	Value	Details
cccFSFactoryActionResult (OUT)	Success	
	<Error Message>	

Return Values

The cccFSFactoryActionResult attribute is set to Success if it is successful; otherwise, an error message displays.

NOTE: The trustee FDN specified for the cccFSFactoryActionTarget attribute should be entered as a typeless, dotted, fully distinguished name such as bob.hq.org.

Example

The following LDIF file shows how to remove user1.hq.org as a trustee of Server1/Vol1:path1/subpath1.

```
#Example LDIF for ClearTrustee Action Object
version: 1
dn: cn=cleartrusteetest,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,ou=resources,o=org#
 0#/path1/subpath1
cccFSFactoryActionOperation: ClearTrustee
cccFSFactoryActionTarget: user1.hq.org
cccFSFactoryActionTrigger: Ready
```

6.1.3 CopyDir

Description

The CopyDir operation recursively copies the contents from Path1 to Path2.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	CopyDir	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath	Path of the directory in SYN_PATH format.
cccFSFactoryActionTarget (IN)	VolumeDN#0#/subpath	This user is removed as a trustee of Path1.

Attribute	Value	Details
cccFSFactoryActionOption	<Option> <ParentPaths>1</ParentPaths> </Option>	ParentPaths : 0 – (or missing tag) Do not create the target path if any parent paths are missing. 1 – Create any missing parent paths as well as the target path
cccFSFactoryActionResult (OUT)	Success <Error Message>	

Return Values

cccFSFactoryActionResult is set to Success if it is successful; otherwise, an error message displays.

Notes

- This action generates a Generic Copy Data event which is eligible for Agent delegation.
- CopyDir only works with contents under a specified directory, and not with single files. Use the CopyFile action to handle an individual file.

Example

The following LDIF shows how to perform a copy of the entire contents of Server1/Vol1:path1/subpath1 to Server2/Data:users/bob.

```
#Example LDIF for CopyDir Action Object
version: 1
dn: cn=copydirtest,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,ou=resources,o=org#
 0#/path1/subpath1
cccFSFactoryActionPath2: cn=SERVER2_DATA,ou=servers,o=org#
 0#/users/bob
cccFSFactoryActionOption: <Option><ParentPaths>1</Parent
 Paths></Option>
cccFSFactoryActionOperation: CopyDir
cccFSFactoryActionTrigger: Ready
```

6.1.4 CopyFile

Description

The CopyFile operation copies the contents of a single file from Path1 to Path2.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	CopyFile	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath/file	Path of source file in SYN_PATH format
cccFSFactoryActionPath2 (IN)	VolumeDN#0#/subpath/file	Path of target file in SYN_PATH format
cccFSFactoryActionOption	<Option> <SourceObject/> <TargetObject/> </Option>	ParentPaths : 0 – (or missing tag) do not create the target file if any parent paths are missing 1 – create any missing parent paths Overwrite: 0 – do not overwrite 1 – (or missing tag) overwrite an existing file 2 – overwrite file only if newer 3 – overwrite file if different size 4 – overwrite file if newer and different size
cccFSFactoryActionResult (OUT)	<Error Message> Success	

Return Values

cccFSFactoryActionResult is set to Success if successful or an error message otherwise.

Notes

- This action generates a Generic Copy Data event which is eligible for Agent delegation.
- CopyFile requires that the target path include the target file name, not just the target parent folder.

Example

The following LDIF shows how to perform a copy of the contents of file Server1/Vol1:path1/subpath1/file-a.txt to Server2/Data:users/bob/file-b.txt

```
#Example LDIF for CopyFile Action Object
version: 1
dn: cn=copyfiletest,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,ou=resources,o=org#
 0#/path1/subpath1/file-a.txt
cccFSFactoryActionPath2: cn=SERVER2_DATA,ou=servers,o=org#
 0#/users/bob/file-b.txt
cccFSFactoryActionOption: <Option><ParentPaths>1</Parent
 Paths><Overwrite>1</Overwrite></Option>
cccFSFactoryActionOperation: CopyFile
cccFSFactoryActionTrigger: Ready
```

6.1.5 CopyTrustee

Description

The CopyTrustee operation copies the rights of a specified trustee for Path1 to a specified trustee for Path2.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	CopyTrustee	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath	Path of the source entry in SYN_PATH format.
cccFSFactoryActionPath2 (IN)	VolumeDN#0#/subpath	Path of the target entry in SYN_PATH format.
cccFSFactoryActionOption (IN)	<Option> <SourceObject/> <TargetObject/> </Option>	SourceObject: FDN of the trustee for path1. TargetObject: FDN of the trustee for path2.
cccFSFactoryActionResult (OUT)	Success <Error Message>	

Return Values

The cccFSFactoryActionResult attribute is set to Success if it is successful; otherwise, an error message displays.

NOTE: The FDNs specified for the `<SourceObject>` and `<TargetObject>` tags should be entered as a typeless, dotted, fully distinguished name such as bob.hq.org.

Example

The following LDIF shows how to perform a copy the trustee assignment of user1 on Server1/Vol1:path1/subpath1 and assign those same rights to bob on Server2/Data:users/bob.

```
#Example LDIF for CopyTrustee Action Object
version: 1
dn: cn=copytrustee,ou=ActionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,ou=resources,o=org#
 0#/path1/subpath1
cccFSFactoryActionPath2: cn=SERVER2_DATA,ou=servers,o=org#
 0#/users/bob
cccFSFactoryActionOption: <Option><SourceObject>user1.user
 s.org</SourceObject><TargetObject>bob.hq.org</TargetObject
 ></Option>
cccFSFactoryActionTrigger: Ready
```

6.1.6 CreateDir

Description

The CreateDir operation creates a directory at the specified path.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	CreateDir	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath	Path of the source entry in SYN_PATH format.
cccFSFactoryActionOption	<Option>	ParentPaths:
	<ParentPaths>1</ParentPaths>	0 – (or missing tag) Do not create the target path if any parent paths are missing.
	</Option>	1 – Create any missing parent paths as well as the target path.
cccFSFactoryActionResult (OUT)	Success	
	<Error Message>	

Return Values

cccFSFactoryActionResult is set to Success if it is successful; otherwise, an error message displays.

NOTE: Even though the complete path is specified in cccFSFactoryActionPath1, all parent paths of the newly specified directory must exist, or the function fails.

Example

The following LDIF example shows how to create subpath1 underneath Server1/Vol1:path1.

```
#Example LDIF for CreateDir Action Object
version: 1
dn: cn=createdirtest,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,o=org#0#/path1/
 subpath1
cccFSFactoryActionOption: <Option><ParentPaths>1</Parent
 Paths></Option>
cccFSFactoryActionOperation: CreateDir
cccFSFactoryActionTrigger: Ready
```

6.1.7 DeleteDir

Description

The DeleteDir operation deletes a directory at the specified path.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	DeleteDir	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath	Path of directory in SYN_PATH format

Attribute	Value	Details
cccFSFactoryActionOption	<Option> <Level>0</Level> <Recursive>0</Recursive> </Option>	Level: 0 – safe. No attribute or rights overrides are performed. This is the default if no Level tag is available 2 – (or tag not available) intermediate. Overrides any read-only / read-inhibit attributes 3 – aggressive. Overrides any read-only / read-inhibit attributes; grants explicit rights, takes ownership if needed to override rights filters or other inherited rights issues Recursive: 0 – Only delete contents of the immediate folder 1 – (or tag not available) Delete contents of the immediate folder and all subfolders
cccFSFactoryActionResult (OUT)	Success <Error Message>	

Return Values

cccFSFactoryActionResult is set to Success if it is successful; otherwise, an error message displays.

Notes

- The specified path must point to a directory, not a file.
- Lack of an Option tag defaults to recursive mode with intermediate level of aggression for delete.

Example

The following LDIF example shows how to delete the directory Server1/Vol1:path1/subpath1.

```
#Example LDIF for DeleteDir Action Object
version: 1
dn: cn=deletetest,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,o=org#0#/path1/subpath
cccFSFactoryActionOperation: DeleteDir
cccFSFactoryActionOption: <Option><Level>3</Level><Recursive>
 0</Recursive></Option>
cccFSFactoryActionTrigger: Ready
```

6.1.8 DeleteFile

Description

The DeleteFile operation deletes a directory at the specified path.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	DeleteFile	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath/file	Path of file in SYN_PATH format
cccFSFactoryActionOption	<Option> <Level>0</Level> </Option>	0 – 1 safe. No attribute or rights overrides are performed. This is the default if no Level tag is available 2 – (or not tag available) intermediate. Overrides any read-only / read-inhibit attributes 3 – aggressive. Overrides any read-only / read-inhibit attributes; grants explicit rights, takes ownership if needed to override rights filters or other inherited rights issues
cccFSFactoryActionResult (OUT)	<Success> <Error Message>	

Return Values

cccFSFactoryActionResult is set to Success if successful or an error message otherwise.

NOTE: The specified path must point to a single file. File globbing or wildcards are not currently supported at this time.

Example

The following LDIF example shows how to delete the file Server1/Vol1:path1/subpath1/file-a.txt

```
#Example LDIF for DeleteFile Action Object
version: 1
dn: cn=deletefiletest,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,o=org#0#/path1/
 subpath1/file-a.txt
cccFSFactoryActionOperation: DeleteFile
cccFSFactoryActionOption: <Option><Level>3</Level></Option>
cccFSFactoryActionTrigger: Ready
```

6.1.9 Rename

Description

The Rename operation renames an existing directory entry (file or folder).

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	Rename	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath	Path of the directory in SYN_PATH format.
cccFSFactoryActionTarget	newpath	New file or directory name.
cccFSFactoryActionResult (OUT)	Success	
	<Error Message>	

Return Values

cccFSFactoryActionResult is set to Success if it is successful; otherwise, an error message displays.

WARNING: Currently all Rename actions override (ignore) any Rename Inhibit / Delete Inhibit flags that would normally prevent renaming a file or directory.

Example

The following LDIF file shows how to rename the file at Server1/Vol1:path1/file1 to file2.

```
#Example LDIF for Rename Action Object
version: 1
dn: cn=renametest,ou=ActionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,ou=resources,o=org
 #0#path1/file1
cccFSFactoryActionTarget: file2
cccFSFactoryActionOperation: Rename
cccFSFactoryActionTrigger: Ready
```

6.1.10 SetFlags

Description

The SetFlags operation sets the file system flags for Path1 to the specified values.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	SetFlags	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath	Path of the directory in SYN_PATH format.
cccFSFactoryActionOption (IN)	RiDiHiArPiIcDcRo	String of flags to set.
cccFSFactoryActionResult (OUT)	Success	
	<Error Message>	

Return Values

The cccFSFactoryActionResult attribute is set to Success if it is successful: otherwise, an error displays.

Notes

- Currently, this action only overwrites the current flags. It does not add to existing flags that might already be set, but instead sets the flags to what has been explicitly listed in the cccFSFactoryActionOption attribute. Any flags not specified are cleared.
- The following flags can be set:
  - Di – delete inhibit
  - Ri – rename inhibit
  - Hi – hidden
  - Ar – archive
  - Pi – purge immediate
  - Ic – compress immediate

- Dc – do not compress
- Ro – read only

- Flag options are case sensitive

Example

The following LDIF file shows how to set the rename inhibit and delete inhibit flags for the path Server1/Vol1:path1.

```
#Example LDIF for SetFlags Action Object
version: 1
dn: cn=setflagstest,ou=ActionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,ou=resources,o=org
 #0#path1
cccFSFactoryActionOption: RiDi
cccFSFactoryActionOperation: SetFlags
cccFSFactoryActionTrigger: Ready
```

6.1.11 SetOwner

Description

The SetOwner operation sets the specified eDirectory object as the owner of Path1.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	SetOwner	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath	Path of the directory in SYN_PATH format.
cccFSFactoryActionTarget (IN)	FDN of new owner.	
cccFSFactoryActionResult (OUT)	Success	
	<Error Message>	

Return Values

The cccFSFactoryActionResult attribute is set to Success if it is successful; otherwise, an error message displays.

NOTE: The owner FDN specified for the cccFSFactoryActionTarget attribute should be entered as a typeless, dotted, fully distinguished name such as bob.hq.org.

Example

The following LDIF file shows how to set bob.hq.org as the owner for the path Server1/Vol1:path1.

```
#Example LDIF for SetFlags Action Object
version: 1
dn: cn=setownertest,ou=ActionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,ou=resources,o=org
 #0#path1
cccFSFactoryActionTarget: bob.hq.org
cccFSFactoryActionOperation: SetOwner
cccFSFactoryActionTrigger: Ready
```

6.1.12 SetQuota

Description

The SetQuota operation is used to set, add, subtract, or clear a directory quota for a given target path.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	SetQuota	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath	Path of the directory in SYN_PATH format.
cccFSFactoryActionOption (IN)	<Option> <SubCmd>#</SubCmd> <Quantity>##</Quantity> </Option>	SubCmd: 0 – Clear quota 1 – Set quota 2 – Add to quota 3 – Subtract from quota Quantity: Integer representing quota value in to set, add, or remove in megabytes.
cccFSFactoryActionResult (OUT)	Success <Error Message>	

Return Values

The cccFSFactoryActionResult attribute is set to Success if it is successful; otherwise, an error message displays.

NOTE: Depending on the initial quota setting of the target path, the value of <SubCmd> produces the results in the table below.

<SubCmd>	No Initial Quota (Unlimited)	Initial Quota of 10 MB	Initial Quota of 0 MB (Locked Out)
0 (clear quota)	No quota (unlimited)	No quota (unlimited)	No quota (unlimited)

<SubCmd>	No Initial Quota (Unlimited)	Initial Quota of 10 MB	Initial Quota of 0 MB (Locked Out)
1 (set quota)	Set to specified <Quantity>	Set to specified <Quantity>	Set to specified <Quantity>
2 (add quota)	No quota (unlimited)	Set to <Quantity> + 10	Set to 10 MB
3 (subtract quota)	No quota (unlimited)	Set to 10 - <Quantity> if result < = 0, then quota set to 0 MB (Locked Out)	Left as 0 MB (Locked Out)

If <SubCmd> is set to any value other than 0, 1, 2, or 3, the operation fails and cccFSFactoryActionResult is set to the error message.

Example

The following LDIF file shows how to set a directory quota of 500 MB on the path Server1/Vol1:path1/subpath1.

```
#Example LDIF for SetQuota Action Object
version: 1
dn: cn=setquotatest,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,ou=resources,o=org#
 0#/path1/subpath1
cccFSFactoryActionOperation: SetQuota
cccFSFactoryActionOption: <Option><SubCmd>1</SubCmd><Quanti
 ty>500</Quantity></Option>
cccFSFactoryActionTrigger: Ready
```

6.1.13 SetTrustee

Description

The SetTrustee operation sets the specified rights for the trustee to the designated path.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	SetTrustee	
cccFSFactoryActionPath1 (IN)	VolumeDN#0#/subpath	Path of the directory in SYN_PATH format.
cccFSFactoryActionOption (IN)	RWCEMFA	Specify initials for file system rights.
cccFSFactoryActionTarget (IN)	FDN of the new trustee	This user will be assigned the rights listed in Option to Path1.

Attribute	Value	Details
cccFSFactoryActionResult (OUT)	Success	
	<Error Message>	
cccFSFactoryActionTrigger	Ready	Needed to activate the action object.

Return Values

The cccFSFactoryActionResult attribute is set to Success if it is successful; otherwise, or an error message displays.

Notes

- The trustee FDN specified for the cccFSFactoryActionTarget attribute should be entered as a typeless, dotted, fully distinguished name such as bob.hq.org.
- The rights are a simple text field with each letter representing a specified access control:
  - R – Read
  - W – Write
  - C – Create
  - E – Crase
  - M – Modify
  - F – File scan
  - A – Access control
- SetTrustee overwrites any previous rights a trustee might have had to the specified path.

Example

The following LDIF file shows how to set user1.hq.org as a trustee of Server1/Vol1:path1/subpath1 with read and file scan rights.

```
#Example LDIF for SetTrustee Action Object
version: 1
dn: cn=settrusteetest,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,ou=resources,o=org#
 0#/path1/subpath1
cccFSFactoryActionOperation: SetTrustee
cccFSFactoryActionOption: RF
cccFSFactoryActionTarget: user1.hq.org
cccFSFactoryActionTrigger: Ready
```

6.2 Required Attributes

- Section 6.2.1, “Trigger,” on page 51
- Section 6.2.2, “Operation,” on page 51

6.2.1 Trigger

Description

The cccFSFactoryActionTrigger attribute is used to notify the Event Monitor that an Action Object is ready to be processed. Upon receipt of the event, the Event Monitor notifies the Engine of the pending Action Object, and processing of the object's contents starts.

In order for the Engine to start processing of an Action Object, the trigger value must be set to the string value Ready.

Parameters

Attribute	Value	Details
cccFSFactoryActionTrigger (IN)	Ready	This is a case-insensitive string value set to the string Ready.

Notes

- If an Action Object must be manually retriggered, be sure to delete the current cccFSFactoryActionTrigger value or change it to a different string other than Ready first, apply the change, then re-add or modify the attribute to have the trigger of Ready again.

  Failure to perform this as two separate actions may prevent the modification of the trigger event from occurring at all, resulting in no event.
- The cccFSFactoryActionTrigger attribute is a Case-Ignore-String syntax attribute. As such, the trigger value may be all uppercase, lowercase, or any mixture in between.

Example

```
# Example LDIF for CreateDir Action Object with the trigger
# attribute set
version: 1
dn: cn=createdirtest,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,o=org#0#/path1
cccFSFactoryActionOperation: CreateDir
cccFSFactoryActionTrigger: Ready
```

6.2.2 Operation

Description

The cccFSFactoryActionOperation attribute specifies the type of action to perform, such as CreateDir or SetQuota.

Parameters

Attribute	Value	Details
cccFSFactoryActionOperation (IN)	AssignPolicy	This is a case-insensitive string value.
	ClearTrustee	
	CopyDir	
	CopyFile	
	CopyTrustee	
	CreateDir	
	DeleteDir	
	DeleteFile	
	Rename	
	SetFlags	
	SetOwner	
	SetQuota	
	SetTrustee	

NOTE: he cccFSFactoryActionOperation attribute is a Case-Ignore-String syntax attribute. As such, the operation value may be all uppercase, lowercase, or any mixture in between.

Example

```
# Example LDIF for CreateDir Action Object with the operation
# attribute set
version: 1
dn: cn=createdirtest,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,o=org#0#/path1
cccFSFactoryActionOperation: CreateDir
cccFSFactoryActionTrigger: Ready
```

6.3 Optional Attributes

The following attributes can be added to any Action Object.

- Section 6.3.1, “Execute Time,” on page 53
- Section 6.3.2, “Link Next,” on page 53
- Section 6.3.3, “Cleanup,” on page 55

6.3.1 Execute Time

Description

The cccFSFactoryActionExecuteTime attribute provides a way to set a deferral time on an action. The value for the attribute is an integer based on UNIX time, which is calculated as the number of seconds since midnight UTC of January 1, 1970.

Parameters

Attribute	Value	Details
cccFSFactoryActionExecuteTime (IN)	0 to 2^{31} -1	This follows the UNIX time _t standard, indicating the number of seconds since midnight UTC Jan. 1, 1970.

Return Values

The cccFSFactoryActionResult attribute is set to Success if it is successful; otherwise, an error message displays.

NOTE: The current release does not provide a mechanism for using human readable time-date stamps.

One helpful site for converting time to UNIX time format is found at the Time Stamp Generator Web site (http://www.timestampgenerator.com).

Example

To set a CreateDir action to take place on February 1, 2009 at 7 pm EST (Eastern Standard Time), the corresponding value for the scheduling attribute is 1233532800.

```
#Example LDIF for CreateDir Action Object with deferred time
version: 1
dn: cn=createdirtest,ou=actionObjects,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,o=org#0#/path1
cccFSFactoryActionExecuteTime: 1233532800
cccFSFactoryActionOperation: CreateDir
cccFSFactoryActionTrigger: Ready
```

6.3.2 Link Next

Description

The cccFSFactoryActionLinkNext attribute sets the next Action Object to process after the current one.

Parameters

Attribute	Value	Details
cccFSFactoryActionLinkNext	<Link> <Process> 0 | 1 </Process> <NextAction> [FDN] </NextAction> </Link>	Process: 0 – Engine processes the next Action Object trigger. 1 – External processing of the next Action Object trigger. NextAction: Dotted, typeless, fully distinguished name of the next action object to trigger.

Notes

- If the attribute is present, the Engine tries to process any valid <NextAction> tags.
- If the value of <Process> is set to anything other than 0, the attribute is left alone. This is useful for cases where an external system such as Novell Identity Manager will drive the triggering of the next linked action.

Example

The following example shows how to link a CreateDir action to a SetTrustee action that follows it:

```
#Example LDIF for CreateDir Action Object with deferred time
version: 1
dn: cn=createdir-1,ou=actions,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,o=org#0#/path1
cccFSFactoryActionLinkNext: <Link><Process>0</Process>
 <NextAction>cn=trustee-1,ou=actions,o=org</NextAction>
 </Link>
cccFSFactoryActionOperation: CreateDir
cccFSFactoryActionTrigger: Ready

dn: cn=settrustee-1,ou=actions,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,ou=resources,o=org#
 0#/path1/subpath1
cccFSFactoryActionOperation: SetTrustee
cccFSFactoryActionOption: RF
cccFSFactoryActionTarget: user1.hq.org
```

NOTE: Only the first action in the sequence should have the cccFSFactoryActionTrigger attribute set to Ready.

6.3.3 Cleanup

Description

The cccFSFactoryActionCleanup attribute specifies whether the Action Object should be deleted from eDirectory after processing for it has completed.

Parameters

Attribute	Value	Details
cccFSFactoryActionCleanup	[System | OnSuccess]	System – the Engine will delete the action object unconditionally at the end of processing, whether or not the defined action completed with a result of “Success”. OnSuccess – the Engine will delete the action object only if its action has completed with a result of “Success”.. Any other value, or absence of the attribute leaves the Action Object behind.

NOTE: Once the Engine has verified the trigger on an Action Object, it is eligible for automatic cleanup regardless of whether the defined action is successful or not.

Example

```
#Example LDIF for CreateDir Action Object with cleanup
version: 1
dn: cn=createdir-1,ou=actions,o=org
changetype: add
objectClass: cccFSFactoryAction
cccFSFactoryActionPath1: cn=SERVER1_VOL1,o=org#0#/path1
cccFSFactoryActionCleanup: System
cccFSFactoryActionOperation: CreateDir
cccFSFactoryActionTrigger: Ready
```

Schema Extensions

7

Table 7-1 *Schema Extensions*

Attribute	Properties	Notes
cccFSFactoryActionCleanup	SYN_CI_STRING Single-valued Sync immediate	Indicates whether the Engine should delete the action object upon completion of the command.
cccFSFactoryActionExecuteTime	SYN_INTERVAL	Specifies date and time to defer Action Object processing.
cccFSFactoryActionLinkNext	SYN_CI_STRING Single-valued Sync immediate	Placeholder for link reference used with Action Object chaining.
cccFSFactoryActionOperation	SYN_CI_STRING Single-valued Sync immediate	Required attribute which determines what type of action to perform.
cccFSFactoryActionOption	SYN_CI_STRING Single-valued Sync immediate	Attribute providing optional parameters needed for some operations.
cccFSFactoryActionPath1	SYN_PATH Single-valued Sync immediate	Primary target path for many operations. When two paths are specified, this path is generally is used as the source path.
cccFSFactoryActionPath2	SYN_CI_STRING Single-valued Sync immediate	Secondary path, typically used as a destination path.
cccFSFactoryActionResult	SYN_CI_STRING Single-valued Sync immediate	Operation result message. This attribute should only be set by the Engine.
cccFSFactoryActionStatus	SYN_CI_STRING Single-valued Sync immediate	System maintained attribute for operation status.
cccFSFactoryActionTarget	SYN_CI_STRING Single-valued Sync immediate	Optional parameter needed for some operations. Generally this is used as a FDN reference to some other object in the tree.

Attribute	Properties	Notes
cccFSFactoryActionTrigger	SYN_CI_STRING Single-valued Sync immediate	Set to the value “Ready” when an action object is ready for processing.

Release Notes

8

- CopyDir has been updated to include parameters for overwrite options.
- DeleteDir has been updated to include parameters for aggression level.
- The CopyFile and DeleteFile actions have been added.
- Redrive capability has been added to Action Object processing. Any invalid parameters that are caught before actual action processing flag the event as “redrivable”, so that adjustments to the Action Object data can be made and re-processed from with the NSMAdmin Pending Events interface without the need to reset the trigger.
- Installation no longer extends the schema with the following unused or obsolete attributes: cccFSFactoryActionLinkStart, cccFSFactoryActionExecuteOption, cccFSFactoryActionAssociation. cccFSFactoryActionControl.
- Certain events, such as CopyDir and CopyFile, may generate secondary dependent events such as GenericCopyData which are eligible for Agent delegation.