Novell
ID Provider Driver

3.6

www.novell.com

IMPLEMENTATION GUIDE

July 22, 2008

Novell®

Legal Notices

Novell, Inc., 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, Novell, Inc., 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, Novell, Inc., 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, Novell, Inc., reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

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. See the Novell International Trade Services Web page (http://www.novell.com/info/exports/) for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Novell, Inc., has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed on the Novell Legal Patents Web page (http://www.novell.com/company/legal/patents/) and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the latest online documentation for this and other Novell products, see the Novell Documentation Web page (http://www.novell.com/documentation).

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 guide explains the purpose of the ID Provider driver and how to implement the driver.

Audience

This guide is intended for Identity Manager administrators.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments 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 this guide, visit the Identity Manager Drivers Documentation Web site (http://www.novell.com/documentation/idm36drivers).

Additional Documentation

For documentation on Identity Manager, see the Identity Manager Documentation Web site (http://www.novell.com/documentation/idm36/index.html).

Documentation Conventions

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

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

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* or UNIX*, should use forward slashes as required by your software.

Understanding the ID Provider Driver

1

The ID Provider driver enables you to create and maintain a central source of unique IDs that can be consumed by client applications or systems. When the driver receives an ID request from a client, it generates an ID based on policies you define, passes it to the client, and then stores it in the Identity Vault.

1.1 Why Use the Driver?

There are many different reasons why you would want to use the ID Provider driver. For example:

- For administrators it is convenient to have one basic ID for each objects in the system, and to have complete control of the ID. No other system can change this ID.
- You can use the ID Provider driver in conjunction with the WorkOrder driver to verify that each WorkOrder ID that is created is unique.
- You can use the driver to help manage UIDs and GIDs in Linux.

1.2 Design Architecture

Identity Manager drivers listen for events and then apply the proper Identity Manager policies for the event. That information is then passed to the Metadirectory engine that executes the policies.

The ID Provider driver is different from all other Identity Manager drivers. It also listens for events, but it has two sets of policies: the Identity Manager policies and the ID Provider policies. The ID Provider policies allow the driver to generate and assign unique IDs to objects.

The driver has three major components:

- **ID Client:** The ID client communicates with the ID Provider driver to obtain a unique ID. The client can be another Identity Manager driver (for example, the WorkOrder driver) or a standalone Java* application.
- **ID Provider Driver:** The driver receives ID requests from clients, generates unique IDs that are stored in the Identity Vault, and passes the unique IDs back to the client. The driver uses LDAP to access the Identity Vault and uses Java RMI (Remote Method Invocation) to communicate with ID clients.
- **Identity Vault:** The Identity Vault provides the location for storing unique IDs and also contains the policies used to generate the IDs. All IDs and policies are stored in the ID Policy Container.

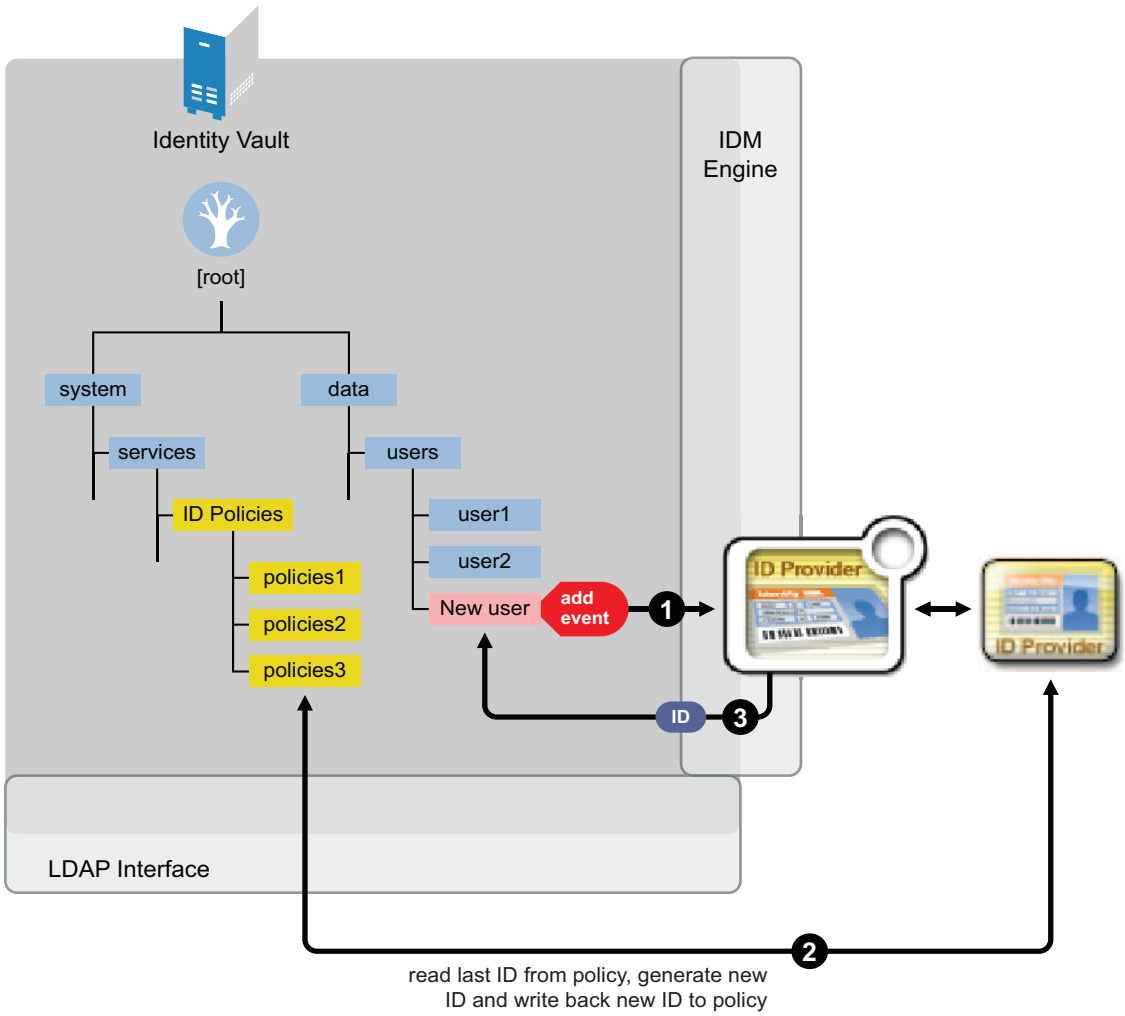
The ID Provider driver can be used in two different scenarios:

- “Scenario 2: Using an LDAP Database to Store the ID Provider Policies” on page 11

Scenario 1: Using the Identity Vault to Store the ID Provider Policies

This is the most commonly used scenario with the driver. The ID Provider policies are created and stored in the Identity Vault when the driver is created and configured. Figure 1-1 shows how an unique ID is generated.

Figure 1-1 *Identity Vault Stores the ID Provider Policies*



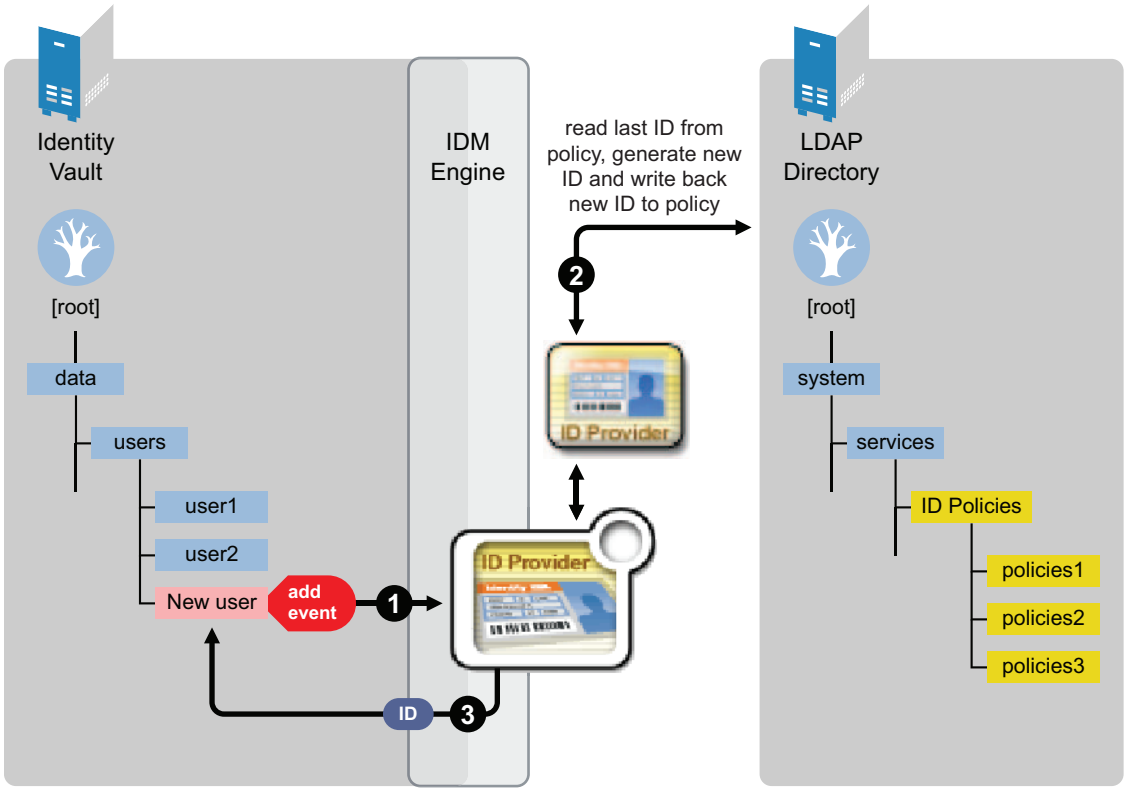
1. A new User object is created in the Identity Vault, then the ID Provider driver picks up the Create event.
2. The ID Provider driver reads the last ID that was generated from the ID Provider polices in the Identity Vault and generates a new ID. The ID is then written back to the ID Provider policies in the Identity Vault to track the unique IDs.
3. The ID Provider driver then assigns the new ID to the new User object.

All events are tracked and stored in the Identity Vault.

Scenario 2: Using an LDAP Database to Store the ID Provider Policies

This scenario allows you to use an LDAP database to store the ID Provider policies instead of using the Identity Vault. Figure 1-2 shows how a unique ID is generated with the LDAP database.

Figure 1-2 *LDAP Database Stores the ID Provider Policies*



1. A new User object is created in the Identity Vault, then the ID Provider driver picks up the Create event.
2. The ID Provider driver reads the last ID that was generated from the ID Provider policies in the LDAP database. The ID is then written back to the ID Provider policies in the LDAP database to track the unique IDs.
3. The ID Provider driver then assigns the new ID to the new User object in the Identity Vault.

1.3 Schema Architecture

The Identity Vault's schema must be extended to support the ID Provider driver functionality. The following two tables describe the schema attributes and classes.

Table 1-1 *Schema Attributes*

Attribute Name	Syntax	Attribute Flags	Description
DirXML-IDPolName	Case Ignore String	Single valued Synchronize immediately	ID Policy object name

Attribute Name	Syntax	Attribute Flags	Description
DirXML-IDPolLastID	Numeric String	Single-valued Synchronize immediately	Last delivered ID
DirXML-IDPolMin	Numeric String	Single-valued	Minimum value for an ID
DirXML-IDPolMax	Numeric String	Single-valued	Maximum value for an ID
DirXML-IDPolPrefix	Case Ignore String	Single-valued	Prefix for a new ID
DirXML-IDPolFill	Boolean	Single-valued	True: Fill ID with 0 up to maximum length False or Empty: Do nothing
DirXML-IDPolArea	Case Ignore String	Single-valued	Exclude/Include list for generated IDs
DirXML-IDPolAreaEI	Boolean	Single-valued	True = IDPolArea = Include list False or Empty = IDPolArea = Exclude list
DirXML-IDPolAccessControl	Boolean	Single-valued	True = IDPolACL list is used False or Empty -- IDPolACL list is not used
DirXML-IDPolACL	Case Ignore String	Single-valued	Comma-delimited list of ID clients to be allowed to request an ID from the ID server
DirXML-IDPolicyContainerDN	Distinguished Name	Single-valued	Link to the ID Policy Container

Table 1-2 *Schema Classes*

Class Name	Contained By	Attributes Contained
ID Policy Container	Country, Domain, Locality, Organization, Organizational Unit, Tree Root	OU
ID Policy	ID Policy Container	IDPolACL IDPolAccessControl IDPolArea IDPolAreaEI IDPolFill IDPolLastID IDPolMax IDPolMin IDPolName IDPolPrefix

Installing the ID Provider Driver

2

The ID Provider Driver is a service driver that is included with the base Identity Manager product. The driver is installed when the Metadirectory engine and drivers are install. For the installation instructions, see “ Installing Identity Manager” in the *Identity Manager 3.6 Installation Guide*.

Configuring the ID Provider Driver

3

The ID Provider driver allows you to generate user IDs for various purposes from a central source. The driver generates IDs based on the policies you define, then stores the user IDs in the Identity Vault.

You can run the driver as a native Java module or as an Identity Manager driver on any supported platform.

The there are two parts to configuring the driver. You must create the ID policies before you create the driver.

3.1 Creating ID Policies

An ID Policy container  is a repository for ID policies and is used in conjunction with the ID Provider driver. An ID policy  allows the ID Provider driver to generate unique IDs. When the ID Provider driver receives an ID request from a client, it generates an ID based on the ID policy specified in the request and passes it to the client.

By default, there are three ID policies that are created when the driver is imported. The three policies are sample policies. You can use these policies or create your own. The default policies are:

- **pid:** The pid policy generates unique ids between the range of 100000 to 2000000000. It also adds the prefix of PID to each unique ID.
- **wfid:** The wfid policy generates unique ids between the range of 10000000 to 99999999. It also adds the prefix of WFID to each unique ID for the workforce ID.
- **woid:** The woid policy generates unique ids between the range of 100000 to 2000000000. It also adds the prefix of WOID to each unique ID.

To create an ID policy:

1. In Designer, right-click the ID Policy container in the *Outline* tab, then click *New* > *ID Policy*.

   The ID Policy container is created when the ID Provider driver is created. The ID Policy container can only reside under the ID Provider driver.
2. Specify the name for the ID policy, then click *OK*.
3. Double-click the ID policy to access the properties page.
4. Use the information in Table 3-1 to create your ID policy, then click *OK* to save the information.

Table 3-1 *The ID Policy’s General Settings*

Field	Description
Policy Name	The name of the ID policy.

Field	Description
Policy's Last ID	The last ID number that was used by this ID policy. If you have deployed this ID policy, use the *Connect* icon to update this field to the last ID number that was stored in the Identity Vault for this ID policy. **NOTE:** Only the ID Provider driver can update the last value stored in the Identity Vault.
Constraints:	
Minimum/Maximum	Numbers must be between 0 and 2147483647. If you have a fixed system that can only handle eight digits, set the *Maximum* to 99999999.
Exclude/Include	Allows you to include or exclude a set of numbers that you type in. Numbers can be typed in a coma-delimited list and you can use ranges, such as 10,100,1000,5000-10000,1099, etc.
Prefix:	Allows you to give a prefix to the IDs that are generated using this ID policy. If you create multiple ID policies, a prefix is useful to see which ID policies are being used. An example is WFID, for workforce IDs.
Fill: Yes/No	If you choose *Yes*, the ID is filled with leading zeros (0) up to the maximum length. This helps keep generated IDs at the same length. If you select *No*, it does nothing and the ID lengths increment over time.
Access Control:	
Enabled	Check this box if you want to enable access control lists.
ACL:	Type in the access control lists you want to use. Access control must be enabled before you can type in ACLs.

3.2 Creating the Driver

You can create the driver through Designer or iManager. It is recommended to use Designer during the planning and implementation phases of the Identity Manager deployment.

1. In Designer, drag and drop the ID Provider driver, from the *Service* folder, onto the Modeler.

2. Select the ID Provider driver from the list, then click *Run*.
3. Specify the following information:
   - **Driver name:** Specify the name of the driver object for your environment.
   - **LDAP server:** Specify the IP address of the LDAP server that contains the ID policies.

- **LDAP port:** Specify the TCP port of the LDAP server. The default is 389 for non-SSL and 636 for SSL.
- **Policy Container DN:** Specify the DN of the policy container.
- **Authenication ID:** Specify the LDAP DN of a user with read/write access to the ID Policy container and its child objects.
- **Authenication Password:** Specify the password of the user used in the *Authenication ID* field.

4 Click *Next*.

5 Click *Configure* if you want to change additional settings on the driver, or click *Close* to create the driver.

If you want to make additional changes to the driver, the following sections contain information about the driver parameters.

- "ID Policy Repository" on page 17
- "Client Options" on page 18
- "Server Options" on page 18

ID Policy Repository

The ID policy repository parameters contain information about the location and how to access any ID policies.

Table 3-2 *ID Policy Repository*

Parameter	Default Value	Description
LDAP Server	127.0.0.1	The IP address or DNS name of the LDAP server holding the ID policies
LDAP Port	636	The TCP port that the LDAP server listens on. The value is usually 389 for non-SSL connections and 636 for SSL connections.
Use SSL	True	Specify whether or not you want to use SSL.
Always trust	True	Specify whether or not you want to trust all servers. If this option is set to True, the ID provider trusts all LDAP servers even if the server certificate is untrusted.
Policy Container DN	LDAP DN for the policy container under the driver object. For example cn=id-policies,cd=id-provider,cn=driverset1,dc=idm, dc=services,dc=system.	Specify or browse to the DN of the policy container in your tree. The policy container can only be created under the ID Provider driver.

Client Options

The client options are for the ID Provider clients. For more information, see Chapter 4, “Configuring ID Clients,” on page 21.

Table 3-3 *Client Options*

Parameter	Default Value	Description
Client name	ID-Provider Driver	This is the name the driver uses when it acts as an ID client and requests and ID from the provider. This is useful for tracing and if access control is enabled on any of the ID policies. If access control is enabled, a list of ID client names can be specified that are allowed to obtain an ID from the policy. If the client name associated with the request is not in that list, the provider does not issue an ID.
ID Generation Map	workforceID=wfid	Enter a comma-separated list of attribute=policy pairs. For example, workforceID=wfid,uniqueID=uid. This example configures the driver to request IDs from the wfid policy and stores them in the workforceID attribute whenever a new object is created or whenever someone tries to change this attribute. Similarly, IDs from the UID policy are used from the uid attribute. The driver only issues IDs for any attribute if that attribute and the object class holding the attribute are in both the Subscriber, Publisher, Filter, and are set to synchronize. Note: Attribute names must be in the Identity Namespace (not LDAP) and must be case-exact.

Server Options

Allow you to setup clients other than the ID Provider driver using Java remote method invocation (RMI). It also allows you to set ID Provider trace level.

Table 3-4 *Server Options*

Parameter	Default Value	Description
Start RMI	True	Controls whether the ID provider starts an RMI service or not. You only need a running RMI service if you request IDs from other clients than the driver (for example, DirXMLScript policies.) If all IDs are managed through this driver’s filter and ID Generation Map settings, then no RMI service is needed.

Parameter	Default Value	Description
RMI server	172.17.2.117	Specify the IP address the RMI server binds to. Leave this field empty if you want the server to bind to all IP addresses.
RMI port	1199	Specify the TCP port the RMI service listens on. The defined standard port for RMI is 1099. If that port is already in use (you see errors in the trace when you start the driver), use a different port lower than 1024. This configuration assumes a port of 1199 to avoid common port conflicts.
Use legacy ID-server schema?	False	Enables the backward compatibility mode when migrating an existing ID-Server configuration to run with the new ID Provider driver. Setting this to True allows you to keep using legacy ID policies, which do not use the new schema that ships with the ID Provider.
Trace level	ALL	This is not the driver trace level, but the ID Provider trace level. The levels are: ◆ **OFF:** Tracing is turned off. ◆ **FATAL:** Displays only fatal messages. ◆ **ERROR:** Displays only error messages. ◆ **WARN:** Displays only warning messages. ◆ **INFO:** Displays only informational messages. ◆ **DEBUG:** Displays only debug messages. ◆ **ALL:** Displays all messages.

Configuring ID Clients

4

An ID client can be run as a standalone Java process or included in another Identity Manager driver. All clients must use the Java RMI (Remote Method Invocation) interface to request a new ID from the ID Provider driver.

4.1 ID Client

The ID client can be used inside of DirXML® style sheets calling the getNextID function of the com.novell.ncs.idsrv.IDClient Java class.

```
xmlns:id=http://www.novell.com/nxsl/java/
com.novell.idm.idprovider.IDClient
```

To obtain the next available ID from an ID Policy object in the Identity Vault, the ID client uses the following parameters to communicate with the ID Provider driver.

Table 4-1 *ID Client Parameters*

Parameter	Description	Sample
$RMIServer	RMI server host address	localhost
$RMIPort	RMI server port	1099
$UIDRule	ID Policy object name to retrieve an ID from	uniqueCN
$IDClient	ID Client name to identify this client at the RMI server	Client-No2
$Tracelevel	Trace level Through the trace level setting it’s possible to see specific trace information in the DirXML ID Servers main screen. The trace level is a bit mask and can be combined. Trace values and levels: 0 = off 1 = low 2 = medium 3 = high 4 = exceptions	1

```
<xsl:variable name="RMIServer" select="'192.168.65.100'"/>
<xsl:variable name="RMIPort" select="'1099'"/>
<xsl:variable name="UIDRule" select="'Unix UID'"/>
<xsl:variable name="IDClient" select="'NIS-SUB-A-CST'"/>
<xsl:variable name="GIDRule" select="'Unix GID'"/>
<xsl:variable name="Tracelevel" select="'9'"/>

<xsl:variable name="uid"
select="id:getNextID($RMIServer,$RMIPort,$UIDRule,$IDClient,$Trace
level)"/>
```

4.2 Standalone Client

The standalone client is run as a Java process that calls the main function of the com.novell.ncs.idsrv.IDClient Java class.

```
%JRE_HOME%\java -noverify -classpath %CLASSPATH%
com.novell.idm.idprovider.IDClient <parameters>
```

To obtain the next available ID from an ID Policy objects in the Identity Vault, the client uses the following parameters to communicate with the driver.

Table 4-2 *Standalone ID Client Parameters*

Parameter	Description	Sample
-h	RMI server host address	-h localhost
-p	RMI server port	-p 1099
-o	ID Policy object name to retrieve an ID from	-o uniqueCN
-c	ID Client name to identify this client at the RMI server	-c Client-No1
-t	Trace level Through the trace level setting it's possible to see specific trace information in the DirXML ID Servers main screen. The trace level is a bit mask and can be combined. Trace values and levels: 0 = off 1 = low 2 = medium 3 = high 4 = exceptions	-t 1

Parameter	Description	Sample
-m	Remote RMI server command to be executed at the RMI server console	-m reinitialize

```
%JRE_HOME%\java -noverify -classpath %CLASSPATH%
com.novell.idm.idprovider.IDClient -h localhost -p 1099 -o Policy -t 1
-c Client -l 1
```

Managing the Driver

5

As you work with the ID Provider driver, there are a variety of management tasks you might need to perform, including the following:

- Starting and stopping the driver
- Viewing driver version information
- Using Named Passwords to securely store passwords associated with the driver
- Monitoring the driver's health status
- Backing up the driver
- Inspecting the driver's cache files
- Viewing the driver's statistics
- Using the DirXML Command Line utility to perform management tasks through scripts
- Securing the driver and its information

Because these tasks, as well as several others, are common to all Identity Manager drivers, they are included in one reference, the *Identity Manager 3.6 Common Driver Administration Guide*.