



## **IDM Connector for Google Apps**

Version 2.1

<b>1</b>	<b>Introduction .....</b>	<b>3</b>
1.1	Overview .....	3
1.2	Driver Features.....	3
1.2.1	Local and Remote Platforms .....	3
1.2.2	Role-Based Entitlements.....	3
1.2.3	Synchronizing Data.....	3
1.2.4	Password Synchronization .....	3
1.2.5	Queries.....	3
1.2.6	Commands .....	3
<b>2</b>	<b>Installing.....</b>	<b>4</b>
2.1	Where to install the connector .....	4
2.2	Prerequisites .....	4
2.3	Installing the connector .....	4
2.4	Choosing the right install source.....	4
2.5	Installing the connector on Netware .....	4
2.6	Installing the connector on Windows .....	5
2.7	Installing the connector on UNIX/Linux.....	5
<b>3</b>	<b>Configure Google Apps.....</b>	<b>6</b>
<b>4</b>	<b>Importing the example configuration file.....</b>	<b>7</b>
4.1	Importing the example configuration file through iManager.....	7
4.2	Importing the example configuration file into Designer project.....	7
<b>5</b>	<b>Configure the connector.....</b>	<b>7</b>
<b>6</b>	<b>Using the connector .....</b>	<b>10</b>
6.1	The connector schema.....	10
6.2	Supported events.....	10
6.2.1	Sample <add/> events .....	11
6.2.2	Sample <modify/> events .....	12
6.2.3	Sample <modify-password/> events .....	13
6.2.4	Sample <delete/> events .....	14
6.2.5	Sample <query/> events .....	15

# 1 Introduction

## 1.1 Overview

The Identity Manager 3.5.1 connector for Google Apps synchronizes data from the Identity Vault to Google's online service Google Apps. The connector can create and manage user accounts and email lists. The connector supports password synchronization.

## 1.2 Driver Features

### 1.2.1 Local and Remote Platforms

The IDM Connector for Google Apps runs in any Identity Manager 2.0 - 3.5.1 installation or Remote Loader installation. See Identity Manager Components and System Requirements in the Identity Manager Installation Guide.

### 1.2.2 Role-Based Entitlements

The IDM Connector for Google Apps does not implement entitlements.

### 1.2.3 Synchronizing Data

The connector only supports synchronizing data on the subscriber channel.

### 1.2.4 Password Synchronization

The IDM Connector for Google Apps does support password synchronization. The connector supports password migration of SHA-1 hashed passwords.

### 1.2.5 Queries

The connector supports Subscriber queries. Query support is somewhat limited, general queries such as a query to return all Google Apps users with a given name of "John" is not supported. Queries to return a specific Google Apps user such as a query to return an instance of a Google Apps user with username "SmithJ" are supported.

### 1.2.6 Commands

The connector supports direct commands by implementing the CommandProcessor interface.

## 2 Installing

### 2.1 Where to install the connector

You can install the IDM Connector for Google Apps on the same server as the Identity Management engine or remotely on a server running the Identity Manager Remote Loader.

See the Identity Management documentation about how to setup the Identity Manager remote loader.

### 2.2 Prerequisites

The prerequisites for installing the IDM Connector for Google Apps are;

- Novell® Identity Manager 2.0 - 3.5.1
- Designer for Identity Manager, a version of Novell iManager with Identity Manager plug-ins.

### 2.3 Installing the connector

This section assumes that you have already installed the Meta Directory engine on the server and need to install only the IDM Connector for Google Apps. For instructions on installing Identity Manager, see Installing Identity Manager in the Identity Manager Installation Guide.

### 2.4 Choosing the right install source

The IDM Connector for Google Apps has two sets of files for the connector. One set of files are built for Identity Manager versions using Java Runtime Environment 1.4 and one set of files for Identity Manager versions using Java Runtime Environment 1.5.

Older versions of Identity Manager are using Java Runtime Environment 1.4. Since Identity Manager 3.5 Java Runtime Environment 1.5 is the default Java Runtime Environment for all platforms except for Identity Manager running on Netware.

The source of the set of files for a Java Runtime Environment 1.4 install is found in the *jre-1.4-install* folder, the set of files for a Java Runtime Environment 1.5 install is found in the *jre-1.5-install* folder from the install media.

### 2.5 Installing the connector on Netware

To install the connector on a Netware server copy all the files from *jre-1.4-install/lib* folder on the install medium to the *sys:system/lib* folder on the Netware server.

## 2.6 Installing the connector on Windows

To install the connector on a Windows server copy all the files from either *jre-1.4-install\lib* or the *jre-1.5-install/lib* folder on the install medium to the *<eDirectory\_root>\lib* folder on the Windows server, where *<eDirectory\_root>* is the folder where eDirectory is installed. The default installation folder for eDirectory on the Windows platform is *C:\Novell\NDS*.

## 2.7 Installing the connector on UNIX/Linux

To install the connector on a UNIX/Linux server copy all the files from either *jre-1.4-install/lib* or the *jre-1.5-install/lib* folder on the install medium to the appropriate folder on the UNIX/Linux server.

For eDirectory versions before 8.8 the default directory to place the files are */usr/lib/dirxml/classes*.

For eDirectory versions from 8.8 and later the default directory to place the files are */opt/novell/eDirectory/lib/dirxml/classes*.

### 3 Configure Google Apps

To enable the connector to be able to provision, de-provision and manage users and email lists the *Google Apps provisioning API* needs to be enabled from within the Google Apps administration pages.

1. Go to <https://www.google.com/a/<your-domain-name>>.
2. Log in with an account with administrative privileges.
3. Click the *User Accounts* menu.
4. Click the *Settings* tab.
5. Check the *Enable provisioning API* check box
6. Click *Save Changes*.

## 4 Importing the example configuration file

### 4.1 Importing the example configuration file through iManager

To import the sample configuration follow the following steps:

1. In iManager click the *New Driver* task in the *Identity Manager Utilities* category.
2. Select an existing DriverSet or select to install the example configuration in a new DriverSet. Then click *next*.
3. Select to *Import a configuration from the client*. Browse and select the *config/GoogleApps - 2007-11-18.xml* file from the installation media. Click *next*.
4. Configure the driver by filling in the configuration parameters.
  - a. Enter a name for the connector in the *Driver name* textbox.
  - b. Enter the driver password if necessary (See the Identity Manager documentation)
  - c. Enter the authentication password. This is the password of the Google Apps Administrator account.
  - d. Click *next*.
5. Select Define Security Equivalences.  
Click Add, then browse to and select a user object that has the rights the driver needs to have on the server.
6. Select Exclude Administrative Roles.  
Click Add, then browse to and select all objects that represent administrative roles and exclude them from replication with the driver.
7. Click *next*
8. Click *Finish*

### 4.2 Importing the example configuration file into Designer project

To import the sample configuration follow the following steps:

1. In the Designer right click on an existing DriverSet object, select *New* and then *Driver*.
2. Click the *Browse* and select the *config/GoogleApps - 2007-11-18.xml* file from the installation media.
3. Click *Run*.
4. Configure the driver by filling in the configuration parameters.
  - a. Enter a name for the connector in the *Driver name* textbox.
  - b. Enter the driver password if necessary (See the Identity Manager documentation)
  - c. Enter the authentication password. This is the password of the Google Apps Administrator account.
  - d. Click *Finish*.

## 5 Configure the connector

After the Connector has been imported then it needs to be configured. This can be done through *iManager* or the *Designer*.

Driver Configuration:

Configuration Section	Configuration Parameter	Configuration Value Description
<b>Driver module</b>		
	Driver type radio buttons	Select Java unless a Remote Loader configuration is required.
	Name of Java Class	com.cosmoskey.novell.GADriverShim
<b>Authentication</b>		
	Authentication ID	The username of the Google Apps Account. For example "admin" or "administrator".
	Authentication Context	The DNS domain name of the Google Apps connector. For Example "cosmoskey.com" or "students.schooloflife.edu".
	Application Password	The password of the Google Apps administrator defined in the Authentication ID field.
<b>Driver Options</b>		
	Domain	The DNS domain name of the Google Apps connector. For Example "cosmoskey.com" or "students.schooloflife.edu".
	Serial	Enter a valid serial number issued for the connector.
<b>Subscriber Options</b>		
	Default Password Hash Strategy	<p>This field takes either of three values. These are "as-sha1", "to-sha1" or "in-clear-text".</p> <p><b>to-sha1:</b> If the value is set to "to-sha1" then the connector will hash and synchronize the values from &lt;modify-password/&gt; events as SHA1 hashed passwords. This is the default and recommended value.</p> <p><b>as-sha1:</b> If the value is set to "as-sha1" then the connector will synchronize the values from &lt;modify-password/&gt; events as SHA1 hashed values. This assumes that the value in the &lt;modify-password/&gt; element is a hash of a password. For example the password "CosmosKey" would have to be sent to the connector as</p>

		<p><code>&lt;modify-password&gt;&lt;password&gt;b5da14e409c7f67c336aa2dcad6ae5d062be07e1&lt;/password&gt;&lt;/modify-password&gt;</code></p> <p><b>to-sha1:</b> If the value is set to “in-clear-text” then the connector will synchronize the values from <code>&lt;modify-password/&gt;</code> events as passwords in clear text to Google Apps.</p> <p><b>Note 1:</b> All requests and all data sent to Google Apps from the connector is made over secure SSL/TLS connections.</p> <p><b>Note 2:</b> Google’s password policy for Google Apps will not be applied to the two options which synchronize hashed passwords to Google Apps. See this <a href="#">link</a> for Google’s Password Policy.</p>
	<p>Generate Random Password on ADD events</p>	<p>This field takes either true or false. If set to true, the connector will generate random passwords for new Google Apps users.</p> <p><b>Note 1:</b> Google Apps requires passwords to be set at all times for new users, even if Google Apps Single Sign-On feature is used. This setting randomizes passwords for new users in Google Apps and will not require the <code>&lt;add/&gt;</code> element to contain a <code>&lt;password/&gt;</code> element.</p>

## 6 Using the connector

### 6.1 The connector schema

The IDM connector for Google Apps supports two object classes. These are *User* and *EmailList*. The following table describes the supported attributes for these classes.

Parent Class	Attribute Name	Attribute Type	Multi Valued	Required	Naming
User	userName	String	False	True	True
	givenName	String	False	True	False
	familyName	String	False	True	False
	suspended	Boolean	False	True	False
	nickName	String	True	True	False
EmailList	emailListName	String	False	True	True
	email	String	True	True	False

### 6.2 Supported events

Identity Manager Events are processed as XML documents. Novell's XML format which Identity Manager and its connector are using is called XDS. The XDS events supported by the IDM Connector for Google Apps SubscriberShim are;

- <add/>
- <modify/>
- <delete/>
- <modify-password/>
- <query/>

Other XDS events such as <move/> and <rename/> are not implemented. This is because Google Apps do not have a concept of context so that a <move/> event would make sense neither has Google Apps any functionality for renaming accounts.

### 6.2.1 Sample <add/> events

This is a sample <add/> XDS event for a user object consumed by the SubscriberShim of the IDM Connector for Google Apps;

```
<nds dtdversion="2.0" ndsversion="8.x">
<source>
<product version="2.0.11.20051121 ">DirXML</product>
<contact>Novell, Inc.</contact>
</source>
<input>
<add class-name="User" dest-dn="DoeJ"
event-id="ADINT-NDS#20080104004855#1#5"
qualified-src-dn="O=CosmosKey\OU=Students\CN=DoeJ"
src-dn="\META\CosmosKey\Students\DoeJ"
src-entry-id="34817">
<add-attr attr-name="familyName">
<value timestamp="1199407423#2" type="string">Doe</value>
</add-attr>
<add-attr attr-name="givenName">
<value timestamp="1199406497#1" type="string">John</value>
</add-attr>
<password><!-- content suppressed --></password>
</add>
</input>
</nds>
```

The most important bit in the <add/> event is that the username attribute is not used for setting the Google Apps username of the new user. The username of a new Google Apps user is retrieved from the *dest-dn* attribute of the *add* element.

This is a sample <add/> XDS event for an EmailList object;

```
<nds dtdversion="2.0" ndsversion="8.x">
<source>
<product version="2.0.11.20051121 ">DirXML</product>
<contact>Novell, Inc.</contact>
</source>
<input>
<add class-name="EmailList" dest-dn="Customers"
event-id="ADINT-NDS#20080104010212#1#1"
qualified-src-dn="O=CosmosKey\OU=EmailLists\CN=Customers"
src-dn="\META\CosmosKey\EmailLists \Customers"
src-entry-id="34818" timestamp="1199408532#1"/>
</input>
</nds>
```

As with User objects the EmailList name of a new Google Apps Email List is retrieved from the *dest-dn* attribute of the *add* element.

## 6.2.2 Sample <modify/> events

This is a sample <modify/> XDS event for a user object where the user has been disabled and had a nickname added at the same time in eDirectory;

```
<nds dtdversion="2.0" ndsversion="8.x">
<source>
<product version="2.0.11.20051121 ">DirXML</product>
<contact>Novell, Inc.</contact>
</source>
<input>
<modify class-name="User" event-id="ADINT-NDS#20080104010221#1#2"
qualified-src-dn="O=CosmosKey\OU=Students\CN=DoeJ"
src-dn="\META\CosmosKey\Students\DoeJ" src-entry-id="34817"
timestamp="1199408541#2">
<association state="associated">DoeJ@cosmoskey.com</association>
<modify-attr attr-name="suspended">
<add-value>
<value timestamp="1199408541#2" type="state">>true</value>
</add-value>
</modify-attr>
<modify-attr attr-name="nickName">
<add-value>
<value timestamp="1199408726#1" type="string">Johnny</value>
</add-value>
</modify-attr>
</modify>
</input>
</nds>
```

This is a sample <modify/> XDS event for an EmailList object;

```
<nds dtdversion="2.0" ndsversion="8.x">
<source>
<product version="2.0.11.20051121 ">DirXML</product>
<contact>Novell, Inc.</contact>
</source>
<input>
<modify class-name="EmailList" event-id="ADINT-NDS#20080104010834#1#1"
qualified-src-dn="O=CosmosKey\OU=Students\CN=Customers"
src-dn="\META\CosmosKey\Students\Customers" src-entry-id="34818"
timestamp="1199408977#3">
<association state="associated">Customers@lab1.cosmoskey.com</association>
<modify-attr attr-name="email">
<remove-value>
<value timestamp="1199408914#1" type="string">customer@nosпам.com</value>
</remove-value>
<add-value>
<value timestamp="1199408977#2" type="string">customer@test.com</value>
</add-value>
<add-value>
<value timestamp="1199408977#3" type="string">info@cosmoskey.com</value>
</add-value>
</modify-attr>
</modify>
</input>
</nds>
```

### 6.2.3 Sample <modify-password/> events

This is a sample <modify-password/> XDS event for an User object where the *default password hash strategy* subscriber variable is set to *to-sha1* (The password value has not been suppressed in the XDS event below for educational reasons normally this value is suppressed in logs by the Identity Management engine);

```
<nds dtdversion="2.0" ndsversion="8.x">
  <source>
    <product version="2.0.11.20051121 ">DirXML</product>
    <contact>Novell, Inc.</contact>
  </source>
  <input>
    <modify-password class-name="User"
      event-id="ADINT-NDS#20080104004855#1#5" src-
        dn="\META\CosmosKey\Students\DoeJ" src-entry-id="34817">
      <association>DoeJ@cosmoskey.com</association>
      <password>N0v311</password>
    </modify-password>
  </input>
</nds>
```

The sample below shows how a *SHA-1* hashed password can be migrated without changing the default password strategy subscriber variable;

```
<nds dtdversion="2.0" ndsversion="8.x">
  <source>
    <product version="2.0.11.20051121 ">DirXML</product>
    <contact>Novell, Inc.</contact>
  </source>
  <input>
    <modify-password class-name="User"
      event-id="ADINT-NDS#20080104004855#1#5" src-
        dn="\META\CosmosKey\Students\DoeJ" src-entry-id="34817">
      <association>DoeJ@cosmoskey.com</association>
      <operation-data>as-sha</operation-data>
      <password>8d47cd18152739a681794d6f4ffbbb1f654ef311</password>
    </modify-password>
  </input>
</nds>
```

The *operation-data* element can contain the string *to-sha1*, *from-sha1* or *clear-text*. See the *default password hash strategy* variable setting in the *Configure the connector* section for a description of these values. If there is an *operation-data* element present with either of these values then this value will determine how the password is handled for the event and the *default password hash strategy* variable is ignored.

## 6.2.4 Sample <delete/> events

This is a sample <delete/> XDS event for an User object;

```
<nds dtdversion="2.0" ndsversion="8.x">
<source>
<product version="2.0.11.20051121 ">DirXML</product>
<contact>Novell, Inc.</contact>
</source>
<input>
<delete class-name="User" event-id="ADINT-NDS#20080104002757#1#1"
qualified-src-dn="o=CosmosKey\OU=Students\CN=DoeJ"
src-dn="\META\CosmosKey\Students\DoeJ" src-entry-id="34817"
timestamp="1199406477#1">
<association state="associated">DoeJ@cosmoskey.com</association>
</delete>
</input>
</nds>
```

This is a sample <delete/> XDS event for an EmailList object;

```
<nds dtdversion="2.0" ndsversion="8.x">
<source>
<product version="2.0.11.20051121 ">DirXML</product>
<contact>Novell, Inc.</contact>
</source>
<input>
<delete class-name="EmailList" event-id="ADINT-NDS#20080104010212#1#1"
qualified-src-dn="o=CosmosKey\OU=Students\CN=Customers"
src-dn="\META\CosmosKey\Students\Customers"
src-entry-id="34818"
timestamp="1199408532#1">
<association state="associated">Customers@cosmoskey.com</association>
</delete>
</input>
</nds>
```

### 6.2.5 Sample <query/> events

This is a sample <query/> XDS event for a User object. The query is generated by a <do-find-matching-object/> token in a matching policy;

```
<nds dtdversion="2.0" ndsversion="8.x">
  <source>
    <product version="2.0.11.20051121 ">DirXML</product>
    <contact>Novell, Inc.</contact>
  </source>
  <input>
    <query class-name="User" event-id="0" scope="subtree">
      <search-class class-name="User"/>
      <search-attr attr-name="userName">
        <value>SmithJ</value>
      </search-attr>
      <read-attr/>
    </query>
  </input>
</nds>
```

This is the response to the query;

```
<nds dtdversion="2.0">
  <output>
    <instance class-name="User" dest-dn="SmithJ">
      <association>SmithJ@cosmoskey.com</association>
      <attr attr-name="givenName">
        <value>John</value>
      </attr>
      <attr attr-name="familyName">
        <value>Smith</value>
      </attr>
      <attr attr-name="suspended">
        <value>false</value>
      </attr>
    </instance>
  </output>
</nds>
```