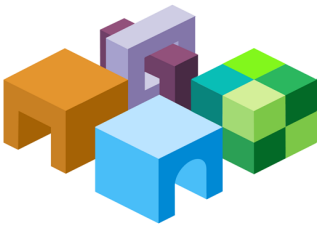

HYPERION® SHARED SERVICES

RELEASE 9.3.1

IMPORT/EXPORT UTILITY

ORACLE | Hyperion

Importing and Exporting Native Directory Data

This document contains the following topics:

Overview

The Import/Export utility, a standalone, command-line utility, is primarily a tool to manage provisioning by facilitating the bulk-provisioning of user and groups with Hyperion product roles. It allows Oracle's Hyperion® Shared Services Administrators to use an XML or CSV file as the source file to create Native Directory users, groups, and provisioning information. Shared Services Administrators can use the Import/Export utility to export, import, and validate data related to various entities:

- Users
- Groups and their relationships
- Roles and their relationship with other roles
- User and group provisioning data

- Delegated lists
- Internal identities of users and groups defined in Native Directory

The utility can be used to export data from all user directories configured with Shared Services but not to import data into external user directories. Hyperion recommends that you run the utility on the computer that hosts Shared Services.

You can use the Import/Export utility to create, update, replace, and delete users, groups, and roles that originate from Native Directory. You can also use it to modify groups and role relationships. The utility also validates the quality of the files used for import operations.

Components of the Import/Export utility:

- Batch (Windows) or shell (UNIX) file to invoke the operation
- Properties file to configure the utility
- Sample XML data file
- Sample CSV (comma-separated values) data file

Use Scenarios

Move Provisioning Data Across Environments

Shared Services Administrators can use Import/Export utility to move users, groups and provisioning data across environments, for example from a development environment to a production environment.

Moving data across environments involves these steps:

- Exporting the data from the source environment into an XML or CSV file
- Modifying the XML or CSV file, if needed
- Validating the updated XML or CSV file
- Importing the XML or CSV file into the target environment

Manage Users and Groups in Native Directory

Shared Services Administrators can create an XML or CSV file containing user and group data, which can then be imported into a target Native Directory to manage users and groups. Bulk creation of users and groups involves these steps:

- Creating a properly formatted XML or CSV file that defines users and groups. See "Import File format" on page 11.
- Validating the XML or CSV file

- Importing the XML or CSV file into the target environment

Bulk Provision Users and Groups

Shared Services Administrators can bulk-provision users and groups using the Import/Export utility. Bulk provisioning involves these steps:

- Exporting the data from Native Directory into an XML or CSV file or creating a properly formatted XML or CSV file
- Modifying the XML or CSV file to include information on role assignment to users and groups
- Validating the XML or CSV file
- Importing the XML or CSV file back into the Native Directory to update it

Installing the Import/Export Utility

An archive containing the utility is installed into `<Hyperion_Home/common/utilities/CSSImportExportUtility`. Extract the contents of the archive into a directory to which the user who performs the import/export operation has read, write, and execute permissions. The extraction process creates the `importexport` directory and copies the required files into it. This directory is referred to as `<ImpEx_home>` in this discussion.

Before Starting Import/Export Operations

- Create a back up of the source Native Directory by exporting data to an LDAP Data Interchange File (LDIF).
- Ensure that all user directories configured in Shared Services (including Native Directory) are running.
- Ensure that Shared Services is running.
- If you are running the Import/Export utility from a server that does not host Shared Services, verify that the prerequisites indicated in "Prerequisites for Running Import/Export Utility from a Remote Host" on page 10 are met.

Sample importexport.properties File

```
#import export operations
importexport.css=file:/C:/Hyperion/deployments/Tomcat5/SharedServices9/
config/CSS.xml
importexport.cmshost=localhost
importexport.cmsport=58080
importexport.username=admin
importexport.password={CSS}MRcYv323uzxGr8rFdvQLcA==
importexport.enable.console.traces=true
importexport.trace.events.file=trace.log
importexport.errors.log.file=errors.log
```

```
importexport.locale=en
# importexport.ssl_enabled = true

# export operations
export.fileformat=xml
export.file=C:/exportNew.xml
export.internal.identities=true
export.native.user.passwords=true
export.provisioning.all=true
export.delegated.lists=false
export.user.filter=*@Native Directory
export.group.filter=*@Native Directory
export.role.filter=*
export.producttype=HUB-9.2.0
#export.provisioning.apps=(HUB=Global Roles)

# import operations
import.fileformat=xml
import.file=C:/exportNew.xml
import.operation=update
import.failed.operations.file=c:/failed.xml
import.maxerrors=0
```

Sequence of Operations

- "Preparing the Property File" on page 4
- Exporting the data into an export file. See "Running the Utility" on page 10.
- (**Optional**): Modifying the data in the export file. See "XML File Format" on page 11 and "CSV File Format" on page 15.
- Validating the import file. See "Running the Utility" on page 10.
- Importing the data. See"Running the Utility" on page 10.

Preparing the Property File

The `importexport.properties` file is a Java properties file that the Import/Export utility uses during runtime to identify the system components to use for the operation.

The `importexport.properties` file contains three sections:

- **`Import export operations:`** The settings in this section are used during import and export operations. These settings identify the Shared Services instance and the user credentials.
- **`Import operations:`** This section contains the parameters for import operations.
- **`Export operations:`** This section contains the parameters for export operations.

➤ To prepare `importexport.properties` file:

1 **Make a backup copy of the `importexport.properties` file. This file is available in the `<ImpEx_home>/samples` directory; for example, `C:\hyperion\common\utilities`**

`\CSSImportExportUtility\importexport\samples` **(Windows) or** `apps/Hyperion/common/utilities/CSSImportExportUtility/importexport/samples` **(UNIX).**

Note:

Hyperion recommends that the `importexport.properties` file used for the operation be stored in *<ImpEx_home>*.

2 **Using a text editor, open the** `importexport.properties` **file. See** **"Sample importexport.properties File" on page 3**.

3 **Update properties. Typically, you should update the properties in** `import export operations` **and one other section, depending on the operation you want to perform:**

- Update `import operations` to import data into Native Directory or to validate an import file
- Update `export operations` to export data into an `.xml` or `.csv` file.

Table 1 Properties for Import–Export Operations

Property	Description
import export operations	
`importexport.css`	The URI where the Shared Services configuration file is stored. For import operations, use the configuration file of the Shared Services instance that manages the Native Directory instance into which data is to be imported. For export operation, use the configuration file of the Shared Services instance that manages the Native Directory instance from which data is to be exported. **Note:** The `CSS.xml` file used by Shared Services server is preferred. However, a local copy in any directory can be used. **Examples:** ● `http://MyServer:<port>/framework/getCSSConfigFile` **Note:** If Shared Services is deployed in SSL-enabled environment, specify the secure URL ● *`file:/<HSS_home>`*`/config/CSS.xml`
`importexport.cmshost`	The DNS name or IP address of the machine that hosts Shared Services. **Example:** `myserver`
`importexport.cmsport`	The Shared Services port number. **Example:** `58080`
`importexport.username`	User account with which to access Shared Services. This user must be able to perform update operations in Native Directory. **Example:** `admin`

Property	Description
importexport.password	Password of the user identified in importexport.username. The utility encrypts this password if you enter a plain text password. **Example:** password
importexport.enable.console.traces	Indicates whether trace information should be displayed in the console where the Import/Export utility is executed. Set this property to true to display trace information in the console. **Example:** true
importexport.trace.events.file	The name and location of the trace log file. If you do not plan to capture trace information in a file, do not set this value. **Example:** impExtrace.log
importexport.errors.log.file	The name and location of the error log file that should capture information on failed transactions during the import or export operation. **Note:** Import/Export utility does not create the error log if you do not specify a file name. **Example:** impExerror.log
importexport.locale	Locale (two-letter language code) to use for the operation. Supported locales are en, fr, it, de, es, pt_BR, nl, ja, ko, zh_CN, zh_TW, ru, tr. The utility attempts to retrieve only data in the specified locale. If data in the specified locale is not available, Native Directory data in the default locale of the server where the utility is run is exported or imported. **Example:** en
importexport.ssl_enabled	Indicates if the import/export operation uses SSL connection. Set the value of this property to true for SSL connections. **Example:** true **Note:** If using SSL connection, make sure that the value of importexport.cmsport indicates the SSL port where Shared Services is available.
export operations	
export.fileformat	The format of the export file. You can export data into XML or CSV files. **Example:** xml
export.file	Location of the file into which the data is to be exported. Import/Export utility creates the file as part of the export process.

Property	Description
	Example: `C:/hyperion/common/utilities/CSSImportExportUtility/importexport/export.xml`
`export.internal.identities`	Indicates whether to export the internal identities of Native Directory users and groups. Internal identity, a component of user and group DN, is unique to each user and group. Shared Services uses an auto-generated identifier as the internal identity. Hyperion products utilize the DN for provisioning purposes. Provisioning information becomes invalid if internal identity is not available, or if it was changed. If you are migrating users from one system to another, you must export the internal identity of users and groups to preserve provisioning information. **Example:** `true`
`export.native.user.passwords`	Indicates whether to export the encrypted passwords of the Native Directory users. **Note:** You cannot perform the `CREATE` import operation if passwords are not specified in the source file. **Example:** `true`
`export.provisioning.all`	Indicates whether to export all provisioning data. Set this property to `false` to export a subset of the provisioning data by using these properties in tandem: ● `export.projectnames` ● `export.applicationnames` Alternatively, you can select a subset by setting `export.provisioning.apps`. **Note:** The values of these properties are ignored if `export.provisioning.all` is set to `true`. **Example:** `true`
`export.delegated.lists`	Indicates whether to export delegated lists. **Example:** `true`
`export.user.filter`	(Optional.) Filter to use to select users for export. See "Considerations for Setting Filters" on page 9. **Example:** `*`
`export.group.filter`	(Optional.) Filter to use to select groups for export. See "Considerations for Setting Filters" on page 9. **Example:** `*`
`export.role.filter`	(Optional.) Filter to use to select roles for export. See "Considerations for Setting Filters" on page 9 **Example:** `*`

Property	Description
`export.producttype`	(Optional.) A comma-separated list of product types for which roles are to be exported (must be specified as `<product code>-<product version>`). See "Product Codes" on page 9. **Example:** `HAVA-9.3.1`
`export.provisioning.apps`	A list of applications (in `(projectname=application name)` format) from which provisioning data is to be exported. Applications names are listed in the Oracle's Hyperion® Shared Services User Management Console. **Example:** `(HUB=Global Roles)(HAVA=Report Designer)`
import operations	
`import.fileformat`	The format of the import file. You can import data from XML or CSV files. **Example:** `xml`
`import.file`	Location of the file to import or validate. You can import data from XML or CSV files, created through an export operations. If you manually create the file, be sure to format it correctly. Use the sample CSV and XML files available in `<ImpEx_home>/samples` as reference. **Example:** `C:/hyperion/common/utilities/CSSImportExportUtility/importexport/import.xml`
`import.operation`	The option for the import operation. Valid options are: ● `create`—Users, groups, and roles are created. Group, role, and provisioning relationships are augmented. ● `update`—Users, groups, and roles are updated. Group, role, and provisioning relationships are replaced. ● `create/update`—A create operation is attempted on each entity in the file. If the operation fails, an update operation is attempted. ● `delete`—Deletes users, groups, and roles. Group, role, and provisioning relationships are deleted. **Example:** `create`
`import.failed.operations.file`	The name and location of the file where the Import/Export utility should record information on failed transactions. **Example:** `impFailedOps.log`
`import.maxerrors`	(Optional.) The maximum number of allowable errors during the import operation. The import operation aborts after the limit is reached. **Example:** `100`

4 Save and close the file.

Product Codes

Table 2 Hyperion Product Codes

Product Code	Product Name
EDS	Analytic High Availability Services
ESB	Essbase Server
ESBAPP	Essbase Application
ESVP	Oracle's Hyperion® Smart View for Office
HAVA	Oracle's Hyperion® Reporting and Analysis – System 9
HBR	Oracle's Hyperion® Business Rules
HFM	Oracle's Hyperion® Financial Management – System 9
HP	Oracle's Hyperion® Planning – System 9
HPS	Oracle's Hyperion® Performance Scorecard – System 9
HSF	Oracle's Hyperion® Strategic Finance
HTM	Oracle's Hyperion® Translation Manager
HUB	Shared Services

Considerations for Setting Filters

The Import/Export utility uses the settings specified in `importexport.properties` to identify the components (Shared Services, Native Directory, and other user directories) to use for the import or export operation.

During an export operation, Import/Export utility exports users, groups, and roles based on the filters set for each. The filters are independent of each other.

If a user directory is not specified in the `export.user.filter` or `export.group.filter` value, the filter is applicable to only the user directory where the filter condition is first encountered; other user directories are ignored. User directories are searched (encountered) in the order specified in the Shared Services configuration file (`CSS.xml`). Because roles are available only in Native Directory, directory specification is irrelevant to role filters.

Note:

If a filter is not specified, data is not exported. *, which is the default filter, exports all data.

Examples: Setting the value of `export.user.filter`, `export.group.filter`, and `export.role.filter` to *`k*@Native Directory`* exports all Native Directory users, groups, and roles that have names starting with `k`.

Setting the value of `export.user.filter`, `export.group.filter`, and `export.role.filter` to `*` exports all users and groups from the first user directory in the search order and all roles from Native Directory.

To export users and groups from a specific user directory, set the value of `export.user.filter` and `export.group.filter` to specify the user directory. For example, to export all users and groups from an LDAP-enabled user directory called `LDAP-West`, set the value of these filters to `*@LDAP-West`.

While updating `importexport.properties`, you can specify how you want to access trace information. You can view trace information in the console where the Import/Export utility is executed or store the information in a trace log file, or choose not to generate trace information. You can also view trace information in the console and record it in a file.

The trace log file can be voluminous. Generate a trace file only if you need to debug the import or export operation. Use the information in the error log to identify failed transactions in the trace file.

Note:

Generating trace information will impact the performance of the Import/Export utility

Prerequisites for Running Import/Export Utility from a Remote Host

If the Import/Export utility is being run from a remote host that does not host Shared Services server:

- Verify that Sun JDK 1.5 is installed on the machine from which the Import/Export utility is run.
- Update the `JAVA_HOME` declaration in `CSSExport`, `CSSImport`, and `CSSValidate` batch files (Windows) or scripts (UNIX) with the location of Sun JDK 1.5 on the machine from which the Import/Export utility is run.

Running the Utility

The Import/Export utility comprises three batch files (Windows) or scripts (UNIX).

- `CSSExport`
- `CSSImport`
- `CSSValidate`

Before running the utility verify that Oracle's Hyperion® Shared Services is running.

➤ To run the Import/Export utility:

1 **Open a command prompt (Windows) or console (UNIX) window.**

2 **Navigate to** *<ImpEx_home>*, **for example,** `C:\hyperion\common\utilities\CSSImportExportUtility\importexport` **(Windows) or** `apps/Hyperion/common/utilities/CSSImportExportUtility/importexport` **(UNIX).**

3 **Execute a command:**

- To export data, run

  `CSSExport.bat importexport.properties` (Windows) or

  `CSSExport.sh importexport.properties` (UNIX)

- To import data, run

  `CSSImport.bat importexport.properties` (Windows) or

  `CSSImport.sh importexport.properties`(UNIX)

- To validate data, run

  `CSSValidate.bat importexport.properties` (Windows) or

  `CSSvalidatealidate.sh importexport.properties`(UNIX)

Note:

If the `importexport.properties` file is not in the directory from which the command is being executed, be sure to use the appropriate path in the commands.

Summary information about the operations is displayed in the console. If transactions fail, review the error log and trace log to determine the cause of the problem and make necessary corrections.

Import File format

Import source file can be an XML file or a CSV file.

- "XML File Format" on page 11
- "CSV File Format" on page 15

XML File Format

The data to be imported or validated using the Import/Export utility can be formatted using XML elements and attributes.

Sample XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<css_data>
     <user id="Test1" provider="Native Directory">
         <login_name>Test1</login_name>
         <first_name>Test</first_name>
         <last_name>User1</last_name>
         <description>Test user 1</description>
```

```
        <email>jch@example.com</email>
                  <internal_id>Test12222</internal_id>
                  <password>{SHA}D1E0sCEVJhyNL3ukAwldcwRJCG4=</password>>
    </user>
    <group id="mygroup01" provider="Native Directory">
        <name>mygroup01</name>
        <description>mygroupDescr</description>
                  <internal_id>39e706a46ad531be:-48fd959f:
112005bb52e:-8000
           </internal_id>
    </group>
    <group_members group_id="G1">
        <group id="CONNECT" provider="orcl">
           <name>CONNECT</name>
        <user id=”myUser” provider="orcl">
           <login_name=”myUser” </login_name">
        </user>
    </group_members>
    <role id="Administrator" product_type="HUB-9.0.0">
        <name>Administrator</name>
        <description>Have unrestricted access</description>
    </role>
    <role_members role_id="Administrator" product_type="HUB-9.0.0">
        <role id="Provisioning Manager" product_type="HUB-9.0.0">
        <name>Provisioning Manager</name>
        </role>
    </role_members>
    <provision project_name="HUB" application_name="Global Roles">
        <roles>
           <user id="Test1" provider="Native Directory">
           <login_name>Test1</login_name>
           </user>
           <role id=Administrator" product_type="HUB-9.0.0">
                     <name>Administrator</name>
                     <description>Complete access</description>
           </role>
        </roles>
      </provision>
    <delegated_list id="test2">
        <name>test2</name>
        <description>List description</description>
        <manager>
           <user id="admin" provider="Native Directory">
              <login_name>admin</login_name>
           </user>
        </manager>
                  <user id="admin" provider="Native Directory">
           <login_name>admin</login_name>
        </user>
        <group id="G1" provider="Native Directory">
           <name>G2</name>
                  </group>
    <delegated_list>
  </css_data>
```

Table 3 XML Schema for Import Files

Element	Attribute	Description and Example
`css_data`		Root element of the file (a container for all other elements).
`user`		A container for attributes of a user.
	`id`	A unique user id on the user directory (typically, the same as `login_name`) **Example:** `pturner`
	`provider`	Name of the source user directory **Example:** `Native Directory`
	`login_name`	Login name of the user **Example:** `pturner`
	`first_name`	First name of the user **Example:** `Paul`
	`last_name`	Last name of the user **Example:** `Turner`
	`description`	User description **Example:** `Administrative User`
	`email`	Email address of the user. **Example:** `pturner@example.com`
	`internal_id`	The auto-generated internal identity of the Native Directory user. **Example:** `911`
	`password`	Encrypted password of the user. **Example:** `{SHA}W6ph5Mm5Pz8GgiULbPgzG37mj9g=`
`group_members`		A container for the definitions of groups that contain subgroups or users.
	`group_id`	Name of the nested group. **Example:** `test-group`
`group`		A container for group attributes.
	`id`	Group identifier. Same as group name **Example:** `testgroup`
	`provider`	Source user directory for the group **Example:** `LDAP-West`
	`name`	Group name

Element	Attribute	Description and Example
		Example: `testgroup`
	`description`	Group description **Example:** `Test group`
	`internal_id`	The auto-generated internal identity of the Native Directory group. **Example:** `611`
`role`		A container for the attributes of a role
	`id`	Unique role identifier **Example:** `Basic User`
	`product_type`	Product type to which the role belongs (specified as `<product code>-<product version>`) **Example:** `HAVA-9.3.1`
	`name`	Unique role name **Example:** `Basic User`
	`description`	Role description **Example:** `Launch and view business rules and objects.`
`role_members`		A container for attributes of aggregated roles.
	`id`	Unique role identifier **Example:** `Basic User`
	`product_type`	Product type to which the role belongs (specified as `<product code>-<product version>`) **Example:** `HAVA-9.3.1`
	`name`	Unique role name **Example:** `Basic User`
`provision`		A container for provisioning information for a project-application combination. This element contains a definition for each user and/or group who is provisioned to a role in a specific application that belongs to a project.
	`project_name`	The project to which the application belongs **Example:** `Business Rules`
	`application name`	The application to which the role belongs. **Example:** `local host`

Element	Attribute	Description and Example
`Delegated List`		Container for delegated lists. The users and groups that are managed through a list must also be defined within this container.
	`id`	Unique list identifier, typically the same as the delegated list name. **Example:** `Basic User`
	`name`	Name of the delegated list. **Example:** `MyList1`
	`description`	List description **Example:** `Delegated list for application creators`
	`manager`	Users and groups who manage the list. Each manager definition may contain user and group definitions. The `provider` identified must be the user directory that contains the manager's account.

CSV File Format

The CSV file format is a tabular data format that contains fields separated by commas and enclosed in double quotation marks. The Import/Export utility supports only Excel-compliant CSV files. The CSV files that Excel outputs differ from the standard CSV files:

- Leading and trailing white space is significant.
- Backslashes are not special characters and do not escape anything.
- Quotes inside quoted strings are escaped with double quotes rather than backslashes.

Excel converts data before putting it in CSV format.

Conversions that Excel performs on CSV files:

- Tabs are converted to single spaces.
- New lines are always represented as the UNIX new line ("\n").
- Numbers greater than 12 digits are represented in truncated scientific notation form.

The Import/Export utility categorizes the CSV file into six entities:

- User
- Group
- Role
- Group_children
- Role_children
- Provisioning
- Delegated list

Each section is identified by two mandatory lines: entity and header. The entity line is identified by a predefined entity name preceded by the # character. The header line follows the entity line. The header line is a comma-separated list of predefined attributes for the entity.

The order of attributes in the header line is not significant. However, the data lines, which follow the header line, must present data in the order in which the header line presents attributes. If data is not to be specified, you use a comma to indicate that a value is not to be set. The entity line, header line, and data lines provide the information required for processing.

Boundaries applied to create, update, and delete operations on CSV files:

- Users, groups, and roles are processed one data line at a time.
- Group members are processed with multiple data lines under one header and one parent group.
- Role members are processed with multiple data lines under one header and one parent role.
- User provisioning is processed with multiple data lines under one header and one group or user.

Error handling is based on the process boundaries. One error is counted for each failure in a process boundary.

Sample CSV file:

```
#user
id,provider,login_name,first_name,last_name,description,email,internal_id,p
assword
admin,Native Directory,admin,admin,none,Administrative User,,911,{SHA}**=
MyDemoTest,Native Directory,MyDemoTest,admin,none,Administrative
User,-,MyDemoTest222,{SHA}**
#group
id,provider,name,description,internal_id
G1,Native Directory,G1,,39e71be:-4859f:11252e:-8000
WORLD,Native Directory,WORLD,All users are members of this group,611
#group_children
id,group_id,group_provider,user_id,user_provider
G1,CONNECT,orcl,,
G1,,,myUser,orcl
#group_children
id,group_id,group_provider,user_id,user_provider
G2,G1,Native Directory,,
#group_children
id,group_id,group_provider,user_id,user_provider
G2Test,,,,
#group_children
id,group_id,group_provider,user_id,user_provider
G3,G2,Native Directory,,
#role
id,product_type,name,description
Administrator,HUB-9.0.0,Administrator,Administrators have unrestricted
access
#role_children
id,product_type,role_id,member_product_type
Administrator,HUB-9.0.0,Provisioning Manager,HUB-9.0.0
#provisioning
project_name,application_name,role_id,product_type,user_id,user_provider,gr
```

```
oup_id,group_provider
HUB,Global Roles,Administrator,HUB-9.0.0,TestUser1,Native Directory,,
#delegated_list
id,name,description,manager_id,manager_provider,user_id,user_provider,group
_id,group_provider
test2,test2,testDescription,admin,Native Directory,admin,Native Directory,,
test2,test2,testDescription,admin,Native Directory,,,G2,Native Directory
```

Table containing attribute descriptions:

- Table 4 on page 17
- Table 5 on page 18
- Table 6 on page 19
- Table 7 on page 19
- Table 8 on page 20
- Table 9 on page 20
- Table 10 on page 21

The following user delineation in an import CSV file can be used to create the user `Test_1` in a Native Directory with the login name `Test_1`, first name `New1`, last name `User1`, description `Test User`, e-mail id `Test1@example.com`, internal id `myid222`, and encrypted password `mypwd`:

```
id,provider,login_name,first_name,last_name,description,email,internal_id,p
assword
Test_1,,Test_1,New1,User1,Test User,Test1@example.com,myid222,mypwd
```

Note:

The utility encrypts plain text passwords specified in the import file.

Table 4 User Entity Attributes

Attribute	Description and Example
`id`	A user id **Example:** `admin`
`provider`	(Optional.) Name of the source user directory **Example:** `Native Directory`
`login_name`	Login name of the user **Example:** `admin`
`first_name`	(Optional.) First name of the user **Example:** `admin`
`last_name`	(Optional.) Last name of the user **Example:** `none`

Attribute	Description and Example
description	(Optional.) User description **Example:** Administrative User
email	(Optional.) Email address of the user **Example:** admin@example.com
internal_id	The auto-generated internal identity of the Native Directory user. **Example:** 911
password	The password of the user. **Example:** password

The following group delineation in an import CSV file can be used to create the WORLD in a Native Directory with the group id WORLD, description Contains all users, and internal id 611:

```
id,provider,name,description, internal_id
WORLD,,WORLD,Contains all users,611,
```

Table 5 Group Entity Attributes

Attribute	Description and Example
id	Group identifier **Example:** testgroup
provider	Source user directory for the group **Example:** LDAP-West
name	Group name **Example:** testgroup
description	(Optional.) Group description **Example:** Test group
internal_id	The auto-generated internal identity of the Native Directory group. **Example:** 911

The following role delineation in an import CSV file can be used to create an aggregated role in Native Directory with role id Designer_rep for product hava-9.3.1 (Reporting and Analysis, version 9.3.1), role name Designer_rep, and description Report Designer. Product type indicates the product to which the aggregated role belongs.

```
id,product_type,name,description
Designer_rep,hava-9.3.1,Designer_rep,Report Designer
```

Table 6 Role Entity Attributes

Attribute	Description and Example
id	Role identifier **Example:** Basic User
product_type	Product type (specified as <product code>-<product version>) to which the role belongs **Example:** HBR-4.1.1.1
name	Role name **Example:** Basic User
description	(Optional) Role description **Example:** Launch and view Business rules and objects.

The following child group delineation in an import CSV file can be used to create the nested group childGp1 with group id childGp1. User member of this group is Test1. Both the user and group are defined in Native Directory:

```
id,group_id,group_provider,user_id,user_provider
childGp1,childGp1,Native Directory,Test1,Native Directory
```

Table 7 Group_Children Entity Attributes

Attribute	Explanation
id	Identifier of the nested group **Example:** test-group
group_id	Name of the nested group **Example:** test-group
group_provider	The source user directory of the group. **Example:** Native Directory
user_id	Unique identifier of a user who belongs to this group **Example:** pturner
user_provider	The source user directory of the user assigned to the group. **Example:** LDAP-West

The following child role delineation in an import CSV file can be used to create the nested role Designer_rep, which belongs to the product hava-9.3.1 (Oracle's Hyperion® Reporting and Analysis – System 9, version 9.3.1), and is assigned to the user Test1:

```
id,product_type,role_id,member_product_type
Test1,hava-9.3.1,Designer_rep,hub-9.3.1
```

Table 8 Role_Children Entity Attributes

Attribute	Explanation and Example
id	Unique identifier of a user to whom the role is assigned **Example:** Test1
product_type	Product type (specified as <product code>-<product version>) to which the role belongs **Example:** hava-9.3.1
role_id	Unique role identifier **Example:** Designer_rep
member_product_type	The product type (specified as <product code>-<product version>) to which the child role belongs. **Example:** hava-9.3.1

The following provisioning delineation in an import CSV file can be used to create a role assignment for application name Global Roles that is assigned to the project test_proj. The role id is Administrator, which belongs to product type HUB-9.0.0. User Test1 and group Group1 defined in Native Directory are provisioned with this role.

```
project_name,application_name,role_id,product_type,user_id,user_provider,gr
oup_id,group_provider
HUB,Global Roles,Administrator,HUB-9.0.0,Test1,Native
Directory,Group1,Native Directory
```

Table 9 Provisioning Entity Attributes

Attribute	Description and Example
app_id	The application to which the role belongs **Example:** WebAnalysis
product_type	Product type (specified as <product code>-<product version>) to which the role belongs **Example:** hava-9.3.1
role_id	Unique role identifier **Example:** Provisioning Manager
user_id	Unique identifier of a user who is provisioned to the role **Example:** pturner
group_id	Unique identifier of a group that is provisioned to the role. **Example:** testgroup

The following delegated list definition in an import CSV file can be used to create delegated list with list id and name testlist, and description my_list. Users admin and Test1 defined in Native Directory are delegated administrators of this list which allows them to manage group testGroup defined on Native Directory.

```
id,name,description,manager_id,manager_provider,user_id,user_provider,group
_id,group_provider
testlist,testlist,my_list,admin,Native Directory,,testGroup,NativeDirectory
testlist,testlist,my_list,Test1,Native Directory,,testGroup,NativeDirectory
```

Table 10 Delegated List Entity Attributes

Attribute	Description and Example
`id`	The list identifier. Typically, the same as the list name. **Example:** `testlist`
`name`	Delegated list name. **Example:** `testlist`
`description`	Delegated list description. **Example:** `my_list`
`manager_id`	Unique identifier of a user or group who manages the list. Each manager must be identified in a separate definition. **Example:** `admin`
`manager_provider`	The user directory that stores the manager's account. **Example:** `Native Directory`
`user_id`	Unique identifier of a user member of the list. Each member must be identified in a separate definition. **Example:** `pturner`
`manager_provider`	The user directory that stores the user member's account. **Example:** `Native Directory`
`group_id`	Unique identifier of a group that is a member of the list. Each member must be identified in a separate definition. **Example:** `myGroup`
`group_provider`	The user directory that stores the group's account. **Example:** `Native Directory`