Oracle Information Rights Management

Oracle IRM Directory Gateway User Guide

10gR3

August 2008

ORACLE

Oracle Information Rights Management, Oracle IRM Directory Gateway User Guide, 10gR3

Primary Author: Martin Abrahams

Contributing Author: Martin Wykes

30 July 2008

Table of Contents

Introduction

This document describes how to use the Oracle IRM Directory Gateway (the Gateway) to manage user accounts and user groups in the Oracle IRM Server database. The Gateway enables you to synchonize users and groups with an existing data source, such as an LDAP database or a file.

The Gateway meets the needs of large organizations that already have databases containing user and group information, and want to synchronize the Oracle IRM Server database with such resources.

You can use the Gateway to select information from an existing data source, and populate the Oracle IRM Server database. You can also use the Gateway to keep the Oracle IRM Server database up to date. For example, you can run the Gateway on a schedule to ensure that relevant changes in the data source are reflected in the Oracle IRM Server database.

If required, you can synchronize the Oracle IRM Server database with multiple data sources, and you can create users and groups manually.

To use the Gateway, you create a configuration file that defines one or more **jobs**. A job applies an **operation** to **objects** in a data **source**. The Gateway provides a number of example configuration files that you can modify to meet your needs. If required, you can develop scripts to run the Gateway according to particular business logic.

Note. Synchronization jobs create an audit record for each affected user account and group. All audit records for a given synchronization job will have the same timestamp. It is advisable to disable auditing during large synchronization jobs, to avoid creating large numbers of identically timestamped records. This avoids problems for users of the Oracle IRM Management Console V4 when running audit reports.

Having defined a configuration file, you can use a Windows Script Host file to run it.

The following section provides an overview of Gateway configuration files.

Gateway Configuration Files

To use the Gateway, you need a configuration file. The following is a simple example. You can find more example files in the Gateway *samples* folder.

```
<gateway>
    <parameter name="log_filename" value="log.txt"/>
    <parameter name="log_append" value="no"/>
    <source name="accounts file" type="file">
        <parameter name="filename" value="myfile.xml"/>
    </source>
    <operation name="sync my server" type="account_sync">
        <parameter name="url" value="myserver:80"/>
        <parameter name="account" value="gateway_account"/>
        <parameter name="password" value="password"/>
    </operation>
    <job name="daily job" source="accounts file" operation="sync my server"/>
</gateway>
```

The example shows that there are three key elements to Gateway configuration:

- Source

  This identifies the source from which objects are retrieved. In this example, the source is an XML file called myfile.xml. For some source types, you can specify an object filter. For example, if the data source is an LDAP database, the source can specify an LDAP search that selects users and/or groups from the database.

- Operation

  This specifies the type of Gateway operation to be executed, the URL of the Oracle IRM Server ("the license server"), and the credentials of the user account on the license server that is to perform the operation. If you want the Gateway to use Windows authentication, you can omit the account and password parameters from the operation. In this case, the operation uses the credentials of Windows account that is running the Gateway.

- Job

  This defines a named pairing of a source and an operation. A configuration file can contain many sources and operations. When you run the Gateway, you specify a job name, and the relevant operation is applied to the relevant source.

The file also specifies a log file and whether to overwrite or append to an existing log file when running a job.

Note. If the specified source is an LDAP database, the configuration must include field mappings.

Object Types

The Gateway supports two types of object: accounts (user accounts) and groups.

Objects are data structures that contain named fields. The Gateway uses data from objects to add or update fields in the corresponding user accounts and groups in the Oracle IRM Server database.

When adding or updating an object, the Gateway also adds or updates a source field in the user account or group. The source field identifies the source from which the object data was retrieved.

Account Objects

The Gateway can add and update the following fields in user accounts.

Field	Description
`AccountName`	The unique name of the account within the license server.
`FirstName`	The account's first name. Can be empty.
`LastName`	The account's last name. Can be empty.
`EmailAddress`	The email address associated with the account.
`Password`	The password of the account. Can be empty if NT authentication is being used. When user details gleaned from an LDAP server are used to create users on a license server, the password specified in the Gateway configuration file will not be used. Instead, the users will be created but will be disabled. An administrator will subsequently have to enable each user, giving them an initial password.
`ExternalAccount`	The external account name as used in the external authentication system. Can be empty.
`Key`	Used to identify an account for operations that manage group membership. If this value is not present, then the `AccountName` value is used instead. For objects retrieved from an LDAP database, the key should map to the object's distinguished name. A plug-in may be necessary if using this feature with Oracle Virtual Directory.

Group Objects

The Gateway can add and update the following fields in groups.

Field	Description
`Name`	The unique name of the group within the license server
`Description`	Description. Can be empty.
`Members`	A collection of group members.
`Key`	Used to identify a group for operations that manage group membership. If this value is not present, then the `Name` value is used. For objects retrieved from an LDAP database, the key should map to the object's distinguished name. A plug-in may be necessary if using this feature with Oracle Virtual Directory.

Implementation of group membership varies across different LDAP implementations. The Oracle Directory Gateway imports group members via a single multi-value property. In Active Directory, this property is "member". On other LDAP implementations it is usually "uniqueMember". Hierarchical representations of group membership, such as those used by OpenLDAP can be supported by using a script source.

Defining Sources

A source defines how to retrieve the objects that will be passed to an operation.

A source has the following attributes:

Source attribute	Description
`name`	A string of up to 32 characters that identifies the source uniquely within a configuration file. The source name is added to each user or group in the Oracle IRM Server database that is created or updated by a job. By default, a job only affects entries that have the relevant source name. For example, if a job involves account removal, only accounts that have the relevant source name are removed by default.
`description`	Describes the source for reference purposes.
`type`	Identifies the source type. See the table below for supported source types.

The Gateway supports the following source types:

Source type	Description
`ldap`	Retrieves objects from a specified LDAP repository according to specified search criteria.
`file`	Retrieves objects stored in an XML representation in a designated file.
`compound`	Retrieves objects from several sources.
`exclude`	When referenced by a compound source, specifies objects to be excluded.
`group_compound`	Combines the results of sources that produce objects of type `account` with the results of sources that produce objects of type `group`, resolving group members into account and group objects.
`script`	Retrieves objects as defined by a JScript or VBScript. This enables you to use data sources and selection criteria that are beyond the scope of the other source types.

The following section describes the types of object that the Gateway can retrieve from a source.

For some source types, the configuration file needs to define a set of parameters. For example, for LDAP sources, the file needs to define mappings between LDAP attributes and the fields of user accounts or groups. These parameters are described in sections relating to each source type.

ldap

`ldap` sources retrieve objects from a specified LDAP repository according to specified search criteria.

You need to define a set of parameters for LDAP sources. For example:

```
<gateway>
. . .
 <source name="Accounts Source" type="ldap">
   <parameter name="mapping" value="Account Mapping"/>
   <parameter name="url" value="LDAP://LDAPSVR/ou=Sales,dc=domain2,dc=local"/>
   <parameter name="user_id" value="cn=Admin,ou=Admin,dc=domain2,dc=local"/>
   <parameter name="password" value="your_password"/>
   <parameter name="search_filter" value="(objectclass=user)"/>
   <parameter name="search_scope" value="subtree"/>
   <parameter name="encrypt_password" value="false"/>
   <parameter name="Command.Cache results" value="false"/>
   <parameter name="Command.Page size" value="50000"/>
   <parameter name="adsi_flag" value="513"/>
 </source>
. . .
</gateway>
```

The following table describes the set of valid parameters for an LDAP source.

Parameter	Description	Default
`mapping`	The name of the mapping to use.	
`url`	The URL of the LDAP server and the base object beneath which users and groups for this source are stored. For example: `LDAP://LDAPSVR/ou=Sales,dc=domain2,dc=local`	
`user_id`	The ID of the user account used to connect to the LDAP server.	
`password`	The password of the user account used to connect to the LDAP server.	
`encrypt_password`	Whether the supplied password is encrypted or not.	true
`search_filter`	The LDAP search filter that identifies all the LDAP data you wish to have converted into objects, as defined by RFC2254. Consult the article on ADSI search filter syntax on Microsoft's web site.	
`search_scope`	The LDAP search scope. One of: `base, onelevel, subtree`	`subtree`
`adsi_flag`	The LDAP source uses the Microsoft ActiveX Data Objects API. This integer setting specifies the LDAP binding authentication options. For details of possible values, see the article on the ADS_AUTHENTICATION_ENUM on Microsoft's web site.	0
`ignore_duplicates`	If duplicates are found in the key field defined in the mapping object while running the query, the default behavior is to ignore the duplicates. If you want the API to report duplicate keys, set this parameter to "no".	Yes

Parameter	Description	Default
`Command.X`	The LDAP source uses the Microsoft ActiveX Data Objects API. Parameters for ADO Commands can be passed directly through using a parameter with the prefix "Command".	
`source_name`	The value that will be placed in the source field of each object added or updated.	
`member_paging`	By default, paged member retrieval mode is used. To use unpaged member retrieval mode, set this parameter to "no". If using Active Directory, use the default value "yes". Otherwise, set this parameter to "no".	Yes

For LDAP sources, the configuration file needs to include a mapping statement that maps LDAP attributes to the fields of a Oracle IRM user account, as described in the following section.

LDAP Attribute Mapping

For LDAP sources, the configuration file needs to include a mapping statement. For example, you might map an LDAP `givenName` to the `FirstName` field in a user account.

The following example shows how to map fields for accounts.

```
<gateway>
. . .
    <mapping name="account" object="account">
        <property name="AccountName" attribute="sAMAccountName"/>
        <property name="FirstName" attribute="givenName"/>
        <property name="LastName" attribute="sn"/>
        <property name="Key" attribute="distinguishedName"/>
    </mapping>
. . .
</gateway>
```

Similarly, you could specify `object=group` and define mappings as required for groups.

Note. The `key` field is a special case, and does not ordinarily exist in the Oracle IRM Server database. It is necessary for certain Gateway operations. Always map `Key` to `distinguishedName` as shown.

The above example maps Oracle IRM account names directly to users' Windows identities (SAM account names). However, there is no requirement for Oracle IRM account names to match Windows identities, and in some cases it is better that they do not match. Oracle usually recommends that you use email addresses as Oracle IRM account names because these are usually both familiar and user friendly. This helps ensure that rights are assigned to the right users.

Nevertheless, you might want users to be able to use Windows authentication to access sealed documents. The following mapping statement maps Oracle IRM account names to email addresses, but also uses an

ExternalAccount property to configure Oracle IRM accounts for Windows authentication using Windows identities.

```
<gateway>
. . .
    <mapping name="account" object="account">
        <property name="AccountName" attribute="emailAddress"/>
        <property name="ExternalAccount" attribute="sAMAccountName"/>
        <property name="FirstName" attribute="givenName"/>
        <property name="LastName" attribute="sn"/>
        <property name="Key" attribute="distinguishedName"/>
    </mapping>
. . .
</gateway>
```

For example, this might create an account for a user with first name **Ruby**, last name **Red**, a Oracle IRM account name of **ruby.red@abc.com**, and an association with a Windows identity of **abc-domain\rr123**. Anyone needing to assign rights to Ruby will find it relatively easy to identify her as **ruby.red@abc.com** and need never know that her Windows identity is **abc-domain\rr123**.

file

A `file` source must contain an XML representation of the user accounts and groups to be passed to an operation. See File source reference for details of the XML representation required.

You need to define a set of parameters for a file source, for example:

```
<gateway>
. . .
  <source name="Groups Source" type="file">
    <parameter name="filename" value="Groups Source.xml"/>
  </source>
. . .
</gateway>
```

The following table describes the set of valid parameters for a file source.

Parameter	Description	Default
`filename`	The path and file name for the XML file containing the source data.	
`source_name`	The value that will be placed the source field of the object created or synchronized from this source.	

compound

compound sources allow you to additively combine data from multiple sources into a single source, so that you can pass the results into a operation. For example if you have user data stored in a number of repositories, you can create a source for each repository and then combine the sources into a single compound source.

The following example shows a compound source that references two file sources. This retrieves objects from both sources and passes them to the operation.

```
<gateway>
. . .
  <source name="Source 1" type="file">
     <parameter name="filename" value="Accounts Source 1.xml"/>
  </source>
  <source name="Source 2" type="file">
    <parameter name="filename" value="Accounts Source 2.xml"/>
  </source>
  <source name="Accounts Compound Source" type="compound">
    <parameter name="sources" value="Source 1,Source 2"/>
</source>
. . .
</gateway>
```

The following table describes the set of valid parameters for a compound source.

Parameter	Description	Default
sources	A comma delimited list of source names.	
source_name	The value that will be placed in the source field of the object added or updated from this source. Only the source_name of the compound source is used. The source_name of each referenced source is not used.	

exclude

`exclude` sources are only useful when referenced by a compound source. A compound source enables you to combine the set of objects retrieved by multiple sources. An exclude source enables you to exclude the results of a particular source from the compound result set. For example, an exclude source might enable particular accounts to be ignored because you do not want them updated for some reason.

The following configuration file extract shows a compound source that references an exclude source:

```
<gateway>
. . .
  <source name="Source A" type="file">
     <parameter name="filename" value="Accounts Source 1.xml"/>
  </source>
  <source name="Source B" type="file">
    <parameter name="filename" value="Accounts Source 2.xml"/>
  </source>
  <source name="Not Source B" type="exclude">
    <parameter name="source" value="Source B"/>
  </source>
  <source name="Accounts Compound Source" type="compound">
    <parameter name="sources" value="Source A,Not Source B"/>
</source>
. . .
</gateway>
```

In this example, the compound source retrieves objects for Source A, then retrieves objects for Source B, and then excludes the objects retrieved for Source B. If Source A retrieved any objects that are also retrieved by Source B, then those objects are not passed to the operation.

The following table defines the only valid parameter for an exclude source.

Parameter	Description	Default
`source`	The name of a source that retrieves objects you want to exclude from the results of a compound source.	

A compound source can reference multiple exclude sources.

group_compound

`group_compound` sources enable operations that update groups and group membership lists. They combine the results of sources that retrieve accounts with the results of sources retrieve groups, to enable the Gateway to update groups and group membership lists in the Oracle IRM Server database.

The Gateway uses the `key` field of each object to determine which objects belong to which groups.

Note. If you do not use a group compound source when updating groups, or the `key` field is not defined, group membership lists might not be updated correctly.

The following example shows a group compound source. In this example, the group compound source combines a set of accounts retrieved by one source with a set of groups retrieved by another source.

```
<gateway>
. . .
  <source name="Accounts Source" type="ldap">
. . .
  </source>
  <source name="Groups Source" type="ldap">
. . .
  </source>
  <source name="Accounts and Groups" type="group_compound">
    <parameter name="sources" value="Accounts Source,Groups Source"/>
    <parameter name="source_name" value="ldap"/>
  </source>
. . .
</gateway>
```

The following table defines the valid parameters for a group compound source.

Parameter	Description	Default
`sources`	A comma-delimited list of sources to be combined.	
`source_name`	The value that will be placed in the source field of the object added or updated from this source. Only the source_name of the group compound source is used; source_names of referenced sources are not used.	

script

A `script` source enables you to define a source using *JScript* or *VBScript* script files rather than XML. Sample script sources are provided in the Gateway *samples* folder. Using scripts gives you greater control over the retrieval of objects from sources, enabling you to retrieve objects from data sources and to use selection criteria that are not supported by other types of source.

Parameters	Name	Default
File path for the script file.	filename	

The language of the script file. Valid values are
❑ JScript
❑ VBScript
JScript
The script file must contain the following function prototype:
`function Retrieve(objSource, objResults)`
`{`
`}`
VBScript
The script file must contain the following function prototype:
`Sub Retrieve(objSource, objResults)`
`End Sub`
	language	JScript

The following example shows how to configure a VBScript script source.

```
<gateway>
. . .
  </source>
    <source name="VBScript Source" type="script">
    <parameter name="filename" value="SampleSource.vbs"/>
    <parameter name="language" value="VBScript"/>
    <parameter name="source_name" value="script"/>
 </source>
. . .
</gateway>
```

Defining Operations

An operation defines how the Gateway processes the objects passed to it from a source.

An operation has the following attributes.

Attribute	Description
`name`	Identifies the operation uniquely within a configuration file.
`description`	Describes the operation for reference purposes.
`type`	Identifies the operation type. See the table below for supported operation types.

The Gateway supports the following types of operation:

Operation	Description
`account_addition`	Adds accounts specified in the source to the license server.
`account_removal`	Removes accounts specified in the source from the license server.
`account_sync`	Updates the license server with accounts in the source, adding, removing and updating accounts as necessary.
`Group_addition`	Adds groups specified in the source to the license server.
`Group_removal`	Removes groups specified in the source from the license server.
`Group_sync`	Updates the license server with groups in the source, adding, removing and updating groups as necessary.
`compound`	Performs multiple operations sequentially on the same set of source data
`script`	Enables you to use JScript or VBScript files to process objects in ways beyond the scope of the other operation types.

An operation can also have parameters depending on its type. For further information, see the sections for the operation types.

Handling Oracle IRM Server Authentication Fields

When you use the Gateway to add or update user accounts, the Gateway must add or update fields that enable users to authenticate to the license server. These fields are unlikely to be stored in any external source, so the Gateway needs to create and update them when processing an operation.

By default, the Gateway configures user accounts to use Oracle IRM authentication. You can use operation parameters to configure accounts to use NT authentication.

The following table summarizes how the Gateway handles the authentication settings when adding new user accounts, and when updating accounts. The behaviour depends on whether a password is passed in to the operation, and whether the `external_system` parameter is set for the operation.

Password provided	external_system parameter set	Result when creating user	Result when updating user
No	No	• Oracle IRM Authentication • Gateway generates random password • Password needs to be reset manually	Authentication fields ignored.
No	Yes	• NT Authentication	Authentication details updated.
Yes	No	• Oracle IRM Authentication • The provided password is used • Cannot Change Password is set	Authentication details updated.
Yes	Yes	• NT Authentication.	Authentication details updated, but the provided password is irrelevant.

Oracle IRM Authentication

By default, when the Gateway adds a new user account, it configures it to use Oracle IRM authentication ("standard authentication", and adds a password to the account.

The password can be:

- Retrieved from an external data source, such as a file source
- Passed in as a source override parameter when running the job or from a script source
- Randomly set by the Gateway itself, if no password is retrieved or passed in

Note. Randomly set passwords are just placeholders, and there is no way to discover their values. Someone with administrative permissions to manage user accounts on the license server needs to reset the passwords and tell the users what passwords to use. It is therefore preferable to use one of the methods of supplying the passwords to an operation.

NT Authentication

If you want user accounts to support NT authentication, you need to use the following operation parameters.

Parameter	Description	Default
`external_system`	If set to NT, indicates that NT authentication is configured. If null, Oracle IRM authentication is configured.	`null`
`NTDomain`	If `external_system` is set to NT and you use Active Directory, use this parameter to specify the NT domain. Otherwise, this parameter is not needed or used.	

If the `external_system` parameter is set, the necessary mapping between the Oracle IRM user account and the NT account is automatically configured for each user account in the source. If the NT domain does not contain an account that maps to an object in the source, the update for that account fails.

account_addition operations

The `account_addition` operation adds an account for each object in the source.

Note. If the source includes objects for which accounts already exist on the license server, the existing accounts are always updated.

Parameter	Description	Default
`url`	The hostname and external port of the license server where accounts are to be added. For example: `myserver.mydomain.net:80`	
`account`	The account name that the Gateway uses to authenticate to the license server. The specified account needs administrative rights to add and modify accounts.	
`password`	The password of the specified account.	
`timeout`	This is the timeout for the connection to the license server in seconds. By default, it is 60 seconds (this is the default in the Context API).	60
`external_system`	Specifies whether to configure the account for NT authentication or Oracle IRM authentication. Valid values are: • Null. Configure the account for Oracle IRM authentication • NT. Configure the account for NT authentication See Handling Oracle IRM Server Authentication Fields for details of how the Gateway configures authentication fields in user accounts depending on the setting of this parameter.	null
`NTDomain`	If external_system = NT, this parameter can specify the NT domain. If using Active Directory, this parameter is required. Otherwise, this parameter is not required.	

The following example shows how to configure an `account_addition` operation. The operation takes objects from the source and adds the accounts to the license server `myserver:80`. The added accounts with have their source property set to the value `file`.

```
<gateway>
. . .
  <operation name="Add Accounts" type="account_addition">
    <parameter name="url" value="myserver.mydomain.net:80"/>
    <parameter name="account" value="superuser"/>
    <parameter name="password" value="password"/>
  </operation>
. . .
</gateway>
```

account_removal operations

The `account_removal` operation removes user accounts from the Oracle IRM Server database if they correspond to objects retrieved from the source.

Note. The operation only removes accounts that are marked as having been updated from the current job's source. If the source field in a particular account does not match the current job's source name, the account is not removed.

Name	Parameters	Default
`url`	The hostname and port of the license server, for example: myserver.mydomain.net:80	
`account`	The account name that the Gateway uses to authenticate to the license server. The specified account needs administrative rights to remove accounts.	
`password`	The password of the specified account.	
`timeout`	This is the timeout for the connection to the license server in seconds. By default, it is 60 seconds (this is the default in the Context API).	60

The following example shows how to configure an `account_removal` operation.

```
<gateway>
. . .
  <operation name="Add Accounts" type="account_removal">
    <parameter name="url" value="myserver.mydomain.net:80"/>
    <parameter name="account" value="superuser"/>
    <parameter name="password" value="password"/>
  </operation>
. . .
</gateway>
```

account_sync operations

The `account_sync` operation synchronizes accounts in the source with the accounts in the Oracle IRM Server database. The main options of the synchronization process are to:

- Create new accounts.
- Update accounts that already exist in the Oracle IRM Server database.
- Delete accounts that are marked as having come from the current job's source, but are no longer in the source.

Each of these options can be controlled by operation parameters.

Parameter	Description	Default
`url`	The hostname and port of the license server , for example, myserver.mydomain.net:80	
`account`	The account name that the Gateway uses to authenticate to the license server. The specified account needs administrative rights to add, modify, and remove accounts.	
`password`	The password of the specified account.	
`timeout`	This is the timeout for the connection to the license server in seconds. By default, it is 60 seconds (this is the default in the Context API).	60
`create`	Controls whether the Gateway creates new accounts for objects in the source that do not correspond to accounts in the Oracle IRM Server database. Valid values are: • `yes`. Create accounts. • `no`. Do not create accounts.	yes
`delete`	Controls whether the Gateway deletes accounts that are marked as having been updated from the job's source, but are no longer in the source. Valid values are: • `yes`. Delete accounts. • `no`. Do not delete accounts.	yes
`update`	Controls whether the Gateway updates existing accounts that are marked as having previously been updated from the same source. Valid values are: • `yes`. Update accounts. • `no`. Do not update accounts.	yes
`update_duplicate`	Controls whether the Gateway updates existing accounts that are not marked as having previously been updated from the same source. Valid values are: • `True`. Update the account. • `False`. Do not update the account. If this parameter is false, the update_source parameter is irrelevant.	true

Parameter	Description	Default
`create_random_password`	Controls whether or not the Gateway creates new accounts with an initial random password. Valid values are: • `yes`. New accounts are created with an initial random password. Users newly created with this option set to "yes" are created in an "enabled" state, with an initial random password known to no-one. These users must reset their password on first use. • `no`. New accounts are created "disabled".	no
`update_source`	Controls whether the Gateway updates an account's source field if updating an account that was previously updated from some other source. • `True`. Update the account's source field to match the current job's source. • `False`. Do not update the account's source field, so that it still appears to have been updated by a previous job's source.	true
`external_system`	Controls whether the Gateway configures accounts for Oracle IRM authentication or NT authentication. Valid values are: • null. Configure accounts for Oracle IRM authentication. • `NT`. Configure accounts for NT authentication See Handling Oracle IRM Server Authentication Fields for details of how the Gateway configures authentication fields in user accounts depending on the setting of this parameter.	null
`NTDomain`	If `external_system` = NT, specifies the NT domain. If you use Active Directory, this parameter is mandatory. Otherwise, this parameter is irrelevant.	

The following example shows how to configure an `account_sync` operation.

```
<gateway>
. . .
  <operation name="Sync Accounts" type="account_sync">
    <parameter name="url" value="myserver.mydomain.net:80"/>
    <parameter name="account" value="superuser"/>
    <parameter name="password" value="password"/>
    <parameter name="external_system" value="NT"/>
    <parameter name="ntdomain" value="mydomain"/>
  </operation>
. . .
</gateway>
```

group_addition operations

Group addition operations retrieve groups from the source and add them to the license server. Groups that are already in the license server are updated.

Note. When processing a group_addition operation, the Gateway does NOT set up a membership list to the groups that it adds. If you want the Gateway to add groups AND set up a membership list for each group, you need to use a compound operation.

Name	Parameters	Default
`url`	The hostname and port of the license server, for example: myserver.mydomain.net:80	
`account`	The account name that the Gateway uses to authenticate to the license server. The specified account needs administrative rights to add groups.	
`password`	The password of the specified account.	
`timeout`	This is the timeout for the connection to the license server in seconds. By default, it is 60 seconds (this is the default in the Context API).	60
`members_update`	Defines how to handle updates to the list of members. Valid values are: • `add` • `remove` • `match` This parameter is only relevant if the group_addition operation is being used as part of a compound operation.	add

The following example shows how to configure a `group_addition` operation.

```
<gateway>
. . .
  <operation name="Add Groups" type="group_addition">
    <parameter name="url" value="myserver.mydomain.net:80"/>
    <parameter name="account" value="superuser"/>
    <parameter name="password" value="password"/>
    <parameter name="members_update" value="add"/>
  </operation>
. . .
</gateway>
```

group_removal operations

The group removal operation removes groups from the license server if they correspond to objects in the source.

Note. The operation does not remove group members, only the groups.

Note. The operation only removes groups that are marked as having been updated from the current job's source. If the source field in a particular group does not match the current job's source name, the account is not removed.

Name	Parameters	Default
`url`	The hostname and port of the license server , for example: myserver.mydomain.net:80	
`account`	The account name used by the Gateway to access the license server	
`password`	The password used by the Gateway to access the license server.	
`timeout`	This is the timeout for the connection to the license server in seconds. By default, it is 60 seconds (this is the default in the Context API).	60

The following example shows how to configure a `group_removal` operation.

```
<gateway>
. . .
  <operation name="Remove Groups" type="group_removal">
    <parameter name="url" value="myserver.mydomain.net:80"/>
    <parameter name="account" value="superuser"/>
    <parameter name="password" value="password"/>
  </operation>
. . .
</gateway>
```

group_sync operations

The `group_sync` operation synchronizes groups in the source with groups in the Oracle IRM Server database. The main options of the synchronization process are to:

- Create new groups.
- Update groups that already exist in the Oracle IRM Server database.
- Delete groups that are marked as having come from the current job's source, but are no longer in the source.

Each of these options can be controlled by operation parameters.

Note. When processing a group_sync operation, the Gateway does NOT synchronize membership lists for the groups that it processes. If you want the Gateway to synchronize groups AND synchronize membership lists for each group, you need to use a compound operation

Group synchronization operations are of type `group_sync`.

Name	Description	Default
`url`	The hostname and port of the license server where accounts are to be synchronized, e.g. `myserver.mydomain.net:80`	
`account`	The account name used by the Gateway to access the license server	
`password`	The password used by the Gateway to access the license server.	
`timeout`	This is the timeout for the connection to the license server in seconds. By default, it is 60 seconds (this is the default in the Context API).	`60`
`create`	Whether the synchronization process should create groups that it finds in the source but not in the license server. Valid values are: • `yes`. Create groups. • `no`. Do not create groups.	`Yes`
`update`	Whether the synchronization process should update groups that it finds in the source and in the license server. Valid values are: • `yes`. Update groups. • `no`. Do not update groups.	`Yes`
`delete`	Whether the synchronization process should delete groups that have been removed from the source since the last synchronization but are still in the license server. Valid values are: • `yes`. Delete groups. • `no`. Do not delete groiups.	`Yes`

Name	Description	Default
update_source	How to handle objects that exist in both the license server and the external source. • True. The object's source field is updated with the value from the external source. (default). • False. Do not update the object's source field.	True
update_duplicate	How to handle objects that exist in both the license server and the external source. • True. The object's data is updated with the value from the external source. (default). • False. Do not update the object's data.	True
members_update	Defines how to handle updates to the list of members. This parameter is mandatory. Valid values are: • add • remove • match This parameter is only relevant if the group_sync operation is being used as part of a compound operation.	

The following example shows how to configure a group_sync operation. The objects loaded from the source are used to add and update groups on the license server. Groups that are not listed in the source will not be deleted from the license server. Each group's list of members will be updated to match the list stored in the external source.

```
<gateway>
. . .
  <operation name="Sync Groups" type="group_sync">
    <parameter name="url" value="myserver:80"/>
    <parameter name="account" value="superuser"/>
    <parameter name="password" value="password"/>
    <parameter name="delete" value="no"/>
    <parameter name="members_update" value="match"/>
  </operation>
. . .
</gateway>
```

compound operations

`compound` operations enable you to run multiple operations for a given source. This is more efficient than running a series of jobs, as the source needs to be retrieved only once for the entire compound operation.

Parameter	Description	Default
`operations`	A comma-delimited list of operations to be performed in sequence.	

The following example shows how to configure a `compound` operation that references two operations.

```
<gateway>
. . .
  <operation name="Sync Accounts" type="account_sync">
    <parameter name="url" value="myserver:80"/>
    <parameter name="account" value="superuser"/>
    <parameter name="password" value="password"/>
    <parameter name="external_system" value="NT"/>
    <parameter name="ntdomain" value="yourcompany"/>
  </operation>
  <operation name="Sync Groups" type="group_sync">
    <parameter name="url" value="myserver:80"/>
    <parameter name="account" value="superuser"/>
    <parameter name="password" value="password"/>
  </operation>
  <operation name="Sync Accounts and Groups" type="compound">
    <parameter name="operations" value="Sync Accounts,Sync Groups"/>
  </operation>
. .
</gateway>
```

The operations are processed in the order that they are listed.

Using Compound Operations To Synchronize Group Membership

One use of `compound` operations is to update group membership lists. Using `group_addition` or `group_sync` operations on their own does not affect membership lists, but running them as part of a compound operation does.

For a `compound` operation to synchronize group membership successfully:

- The source needs to retrieve not only the group(s) that you want to update, but also the accounts that are members of the group(s).

  The Gateway needs the account objects in the source so that it can correctly map each group member onto a user account in the license server.

- The mapping for group objects needs to specify which field of a group contains the membership list.

  For Active Directory, the membership list is held in a multivalued `members` field.

- The mapping for group objects needs to specify how that field identifies each member.

  For Active Directory, each member is identified by distinguished name.

- The `members_update` parameter of the `group_addition` or `group_sync` operation referenced by the compound source must specify whether to `add`, `delete`, or `match` group membership.

The following example shows a `compound` operation that uses `group_sync` to update membership lists.

```
<gateway>
. . .
  <mapping name="Group Mapping" object="group">
    <property name="Name" attribute="name"/>
    <property name="Description" attribute="description"/>
    <property name="Members" attribute="member"/>
    <property name="Key" attribute="distinguishedName"/>
  </mapping>
  <operation name="Sync Groups" type="group_sync">
    <parameter name="url" value="myserver.mydomain.net:80"/>
    <parameter name="account" value="superuser"/>
    <parameter name="password" value="password"/>
    <parameter name="members_update" value="match"/>
  </operation>
  <operation name="Sync Membership" type="compound">
    <parameter name="operations" value="Sync Groups"/>
  </operation>
. .
</gateway>
```

The source for this operation (not shown) must retrieve the groups and the accounts that are members. You can use a compound source to retrieve the necessary information.

file operations

The `file` operation writes the data retrieved from the source into a text file. You can use `file` operations to test a source before running an operation that affects the Oracle IRM Server database.

The data fields that are written out are determined by the `properties` parameter. These fields correspond to the properties of the `account` or `group` object, such as `FirstName` or `Name`.

The operation can take the following parameters.

Parameter	Description	Default
`filename`	The name of the file to write to.	
`verbose`	Output mode (see below). Valid values are: • `Yes`. Verbose mode • `No`. Quiet mode.	`Yes`
`properties`	The list of object properties to write to the file.	
`delimiter`	The delimiter used to separate property data in quiet mode.	`,`
`object`	The type of object to write to file. Valid values are: • `account` • `group` If the source contains both types of object, then you must tell the operation which type to write out.	`null`

The `verbose` parameter controls the format of the file.

A typical output of verbose mode is shown below for an account object:

```
AccountName: Test.User
FirstName: Test
LastName: User
EmailAddress: Test.User@testdomain.local
```

In quiet mode, using the default comma delimiter, the same data would output as:

```
Test.User,Test,User,Test.User@testdomain.local
Test.User2,Test,User2,Test.User2@testdomain.local
```

script operations

A `script` operation enables you to define an operation using *JScript* or *VBScript* script files rather than XML. Samples of this way of defining operations are provided in the Gateway samples folder. Using scripts gives you greater control over an operation.

Parameters	Name	Default
File path for the script file.	filename	

The language of the script file. Valid values are
❑ JScript
❑ VBScript
JScript
The script file must contain the following function prototype:
`function Run(objOperation, objCollection)`
`{`
`}`
VBScript
The script file must contain the following function prototype:
`Sub Run(objOperation, objCollection)`
`End Sub`
	language	JScript

The following example shows how to define a VBScript script operation:

```
<gateway>
. . .
 <operation name="VBScript Operation" type="script">
  <parameter name="filename" value="SampleOperation.vbs"/>
  <parameter name="language" value="VBScript"/>
 </operation>
. . .
</gateway>
```

Defining Jobs

A job is a definition of which source you want to feed into which operation.

The following example shows a job that retrieves objects from a specified source, and passes those objects to a specified operation:

```
<job name="daily job" source="accounts file" operation="sync my server"/>
```

The job name, the source name, and the operation name must all be unique within the configuration file.

A job has the following attributes.

Attributes	Description	Default
`name`	The name of the job. You specify the job name when you invoke the Gateway.	
`source`	The name of the source that defines how to retrieve objects.	
`operation`	The name of the operation to apply to the source data.	

Running Jobs

The Gateway installation folder contains a Windows Script Host file called `smgateway.wsf`.

You can run this file from the command line, using parameters to specify the Gateway configuration file to use, the job to run, and to control some aspects of the job.

For example, the following command line shows how to invoke a job defined in config.xml.

```
cscript smgateway.wsf -config "config.xml" -job "daily job"
```

The set of parameters that you can pass is described in the following table.

Parameter	Description	Notes
`-config`	The name of the Gateway configuration file that defines the job you want to run.	Mandatory
`-job`	The job you want to run.	Mandatory
`-sparameter`	Overrides the value of a source parameter specified in the job. Takes the value *`source_name`*,*`param_name`*=*`param_value`*. Note that you need to identify the source, as a job can reference multiple sources.	Optional
`-oparameter`	Overrides the value of a parameter of the operation specified in the job. Takes the value *`param_name`*=*`param_value`*.	Optional
`-validate`	If specified, validates the job without changing the Oracle IRM Server database.	Optional

The following example shows how to override some parameters specified in a particular source.

```
cscript smgateway.wsf -config "config.xml" -job "daily job"
 -sparameter "ldapSource1.password=ABC123xyz"
 -sparameter "ldapSource1.search_scope=onelevel"
```

The script reports errors, failures, and warnings at the end of the job and in the log file, if the configuration file identifies a log file. The job also reminds you of the location of the log file, if appropriate.

The Gateway reports the following error levels:

- **Error.** Usually a fatal event that causes the Gateway to exit before completing the job.
- **Failure.** A failure that does not stop the Gateway processing the job.
- **Warning.** Indicates a problem with the configuration file. The Gateway continues processing.

Scheduling Jobs

The Gateway does not provide a scheduling mechanism for jobs. However, you can use the scheduling facilities provided by the operating system. Here are the simple steps you need to take:

1. Create a batch file that will run the job you wish to schedule. For example:

```
rem smjob.bat
@echo off
@echo Running the 'daily job'
cscript smgateway.wsf -config "config.xml" –job "daily job"
```

2. From the Windows Control Panel, select **Scheduled Tasks**.
3. Select **Add Scheduled Task**.
4. Complete the wizard, selecting the batch file that you created in Step 1.

Configuration File Reference

This section describes the format of the XML used in a Gateway configuration file.

gateway

Details

Mandatory	Parent Node	Repeatable
Yes	None	No
Child Elements		
mapping, source ,operation ,job, parameter		

Description

The root element of the configuration file.

Attributes

None

Example

```
<gateway>
    <parameter name=”log_filename” value=”log.txt”/>
    <parameter name=”log_append” value=”no”/>
</gateway>
```

source

Details

Mandatory	Parent Element	Repeatable
No	gateway	Yes
Child Elements		
parameter		

Description

A source definition specifies the name for the source and the type of source it is. A source element will typically include parameter elements within it that define how the source is configured.

Attributes

Attribute	Mandatory	Description
name	yes	Name of the source.
description	no	Description for the source.
type	yes	The type of the source.

Example

```
<source name=”LDAP Users” type=”ldap”>
          <parameter name=”url” value=”ldap://ldapserver”/>
     <parameter name=”searchfilter” value=”(objectClass=user)”/>
</source>
```

operation

Details

Mandatory	Parent Element	Repeatable
No	gateway	Yes
Child Elements		
parameter		

Description

An operation definition specifies the name for the operation and the type of operation it is. An operation element will typically include parameter elements within it that define how the operation is configured.

Attributes

Attribute	Mandatory	Description
name	yes	Name of the operation.
description	no	Description for the operation.
type	yes	The type of operation.

Example

```
<operation name="My server sync" type="account_sync">
    <parameter name="url" value="seal://myserver:2001"/>
</operation>
```

job

Details

Mandatory	Parent Element	Repeatable
No	gateway	Yes
Child Elements		
none		

Description

A job that retrieves data from a source and passes it to an operation.

Attributes

Attribute	Mandatory	Description
name	yes	Name of the job
description	no	Description for the job
operation	yes	The operation the job will run
source	yes	The source the job will use to retrieve data and then pass it to the operation

Example

```
<job name="sync"
     operation="sync"
     source="users"/>
```

parameter

Details

Mandatory	Parent Element	Repeatable
No	source, operation, gateway	Yes
Children		
None		

Description

An operation parameter.

Attributes

Attribute	Description
name	Name of the parameter
value	Value of the parameter

Example

```
<parameter name=”licenseserver” value=”seal://myserver:2001”/>
```

mapping

Details

Mandatory	Parent Element	Repeatable
No	Gateway	Yes
Child Elements		
Property		

Description

Groups together the definitions of how to map an LDAP object to a gateway object.

Attributes

Attribute	Mandatory	Description
name	yes	Name of the mapping.
description	no	Description for the mapping.
object	yes	The type of the object being mapped.

Example

```
<mapping name=”user” object=”account”>
            <property name=”AccountName” attribute=”cn”/>
          <property name=”FirstName” attribute=”givenName”/>
</mapping>
```

PROPERTY

Details

Mandatory	Parent Element	Repeatable
No	Mapping	Yes
Child Elements		
None		

Description

A definition of how a LDAP attribute maps onto an object's property.

Attributes

Attribute	Mandatory	Description
name	yes	The property name of the object For example: AccountName on the Account object.
attribute	yes	The LDAP attribute that the data will come from For example: givenName

Example

```
<property name="AccountName" attribute="cn"/>
```

File source reference

This section describes the format of the XML used in a file source of accounts and groups for use with the Gateway.

source

Details

Mandatory	Parent Node	Repeatable
Yes	None	No
Child Elements		
account, group		

Description

The root element of the file source.

Attributes

None

Example

```
<source>
    <account>
      <parameter name=”AccountName” value=”bob.smith”/>
    </account>
</source>
```

account

Details

Mandatory	Parent Element	Repeatable
No	Source	Yes
Child Elements		
Parameter		

Description

An account definition specifies the account details.

Attributes

Attribute	Mandatory	Description
AccountName	Yes	Name of the account
FirstName	No	First name
LastName	No	Last name
Password	No	Password
EmailAddress	No	Email address

Example

```
<account>
  <parameter name=”AccountName” value=”bob.smith”/>
  <parameter name=”FirstName” value=”bob”/>
  <parameter name=”LastName” value=”smith”/>
</account>
```

group

Details

Mandatory	Parent Element	Repeatable
No	Source	Yes
Child Elements		
Parameter		

Description

A group definition specifies the group details.

Attributes

Attribute	Mandatory	Description
Name	Yes	Name of the group
Description	No	Group description
Member	No	Member of the group. This is a multi-value attribute. Each entry reference to a member of the group by name. In the case of accounts this is the "AccountName". In the case of groups, this is the "Name".

Example

```
<group>
  <parameter name="Name" value="Company"/>
  <parameter name="Description" value="The company group"/>
  <parameter name="Member" value="bob.smith"/>
  <parameter name="Member" value="dave.jones"/>
  <parameter name="Member" value="Finance"/>
  <parameter name="Member" value="Engineering"/>
</group>
```

parameter

Details

Mandatory	Parent Element	Repeatable
No	account, group	Yes
Children		
None		

Description

A name /value parameter node.

Attributes

Attribute	Description
name	Name of the parameter
value	Value of the parameter

Example

```
<parameter name=”licenseserver” value=”seal://myserver:2001”/>
```

Gateway configuration

Gateway samples can be found under *samples* in the Gateway installation directory.

The following samples are of particular interest.

file Source Sample

This example shows how to import a set of accounts and groups to a license server using a file source.

This comprises three files:-

- File Source Config.xml – The configuration file for running jobs.
- Accounts Source.xml – Source of accounts in XML format.
- Groups Source.xml – Source of groups in XML format.

To use the sample, you need to modify the operations in the configuration file to specify the URL of your license server, and the account and password that you want the Gateway to use for authentication to the license server.

ldap Source Sample

This example shows how to import a set of accounts and groups to a license server from an LDAP service. There are sample files for Active Directory and Sun ONE Directory.

To use the samples, you need to modify the operations to specify the URL of the license server, and the account and password that you want the Gateway to use for authentication to the license server.

You also need to modify the sources to specify the URL of the search base in the LDAP service, and the user id and password that you want the Gateway to use for authentication to the license server.

Within the configuration files, there are several sources and operations that you can experiment with. In particular, the Active Directory samples include sources that can filter for enabled and disabled Active Directory accounts.

script Sources and Operations

These examples show how you can use script files to define sources and operations for the Gateway in preference to using the built-in types of source and operation.

Toolkit

The Gateway can be customized to perform different kinds of account/group synchronization. For example, when importing accounts from LDAP to the license server, enforce an account name policy of <first_name.last_name>@company.com where the first and last names are derived from LDAP attributes. The toolkit posted on Oracle Technology Network (OTN) gives examples of how this is possible.