SailPoint IdentityIQ

Version 8.0

Administration Guide

Table of Contents

IdentityIQ Introduction

SailPoint IdentityIQ is an identity and access management solution for enterprise customers that delivers a wide variety of IAM processes-including automated access certifications, policy management, access request and provisioning, password management, and identity intelligence. Furthermore, IdentityIQ has a flexible connectivity model that simplifies the management of applications running in the datacenter or the cloud.

Compliance Manager — IdentityIQ Compliance Manager automates access certifications, policy management, and audit reporting through a unified governance framework. This enables you to streamline compliance processes and improve the effectiveness of identity governance, all while lowering costs.

Lifecycle Manager — IdentityIQ Lifecycle Manager manages changes to access through user-friendly self-service request and password management interfaces and automated lifecycle events. It provides a flexible, scalable provisioning solution for addressing the constantly evolving access needs of your business in a way that's both efficient and compliant.

Privileged Account Management Module — IdentityIQ Privileged Account Management module provides a standardized approach for extending critical identity governance processes and controls to highly privileged accounts, enabling IdentityIQ to be used as a central platform to govern standard and privileged accounts.

Connectors and Integration Modules — IdentityIQ offers Integration Modules that support the extended enterprise IT infrastructure. Third party provisioning and service desk integration enable multiple sources of fulfillment to access change. Service catalog integration supports a unified service request experience with integrated governance and fulfillment. Mobile device management integration mitigates risk posed by mobile devices through centralized visibility, control and automation. And IdentityIQ's IT security integration provides enhanced security with improved responsiveness and controls.

Open Identity Platform — SailPoint's Open Identity Platform lays the foundation for effective and scalable IAM within the enterprise. It establishes a common framework that centralizes identity data, captures business policy, models roles, and takes a risk-based, proactive approach to managing users and resources. The Open Identity Platform is fully extensible, providing robust analytics which transforms disparate and technical identity data into relevant business information, resource connectivity that allows organizations to directly connect IdentityIQ to applications running in the datacenter or in the cloud, and APIs and a plugin framework to allow customers and partners to extend IdentityIQ to meet a wide array of needs. An open platform allows organizations to build a single preventive and detective control model that supports all identity business processes, across all applications-in the datacenter and the cloud. SailPoint IdentityIQ applies consistent governance across compliance, provisioning and access management processes, maximizing investment and eliminating the need to buy and integrate multiple products.

Password Manager — IdentityIQ Password Manager delivers a simple-to-use solution for managing user passwords across cloud and on-premises applications policies from any desktop browser or mobile device. By providing intuitive self-service and delegated administration options to manage passwords while enforcing enterprise-grade password, IdentityIQ enables businesses to reduce operational costs and boost productivity.

Amazon Web Services (AWS) Governance Module — Enables organizations to extend existing identity lifecycle and compliance management capabilities within IdentityIQ to mission-critical AWS IaaS environments to provide a central point of visibility, administration, and governance across the entire enterprise. This includes policy discovery and access history across all organization accounts, provisioning AWS entities and objects, access review and certification, and federated access support.

SAP Governance Module — Improves the user experience by introducing a new integrated visual interface for navigating and selecting SAP identities and roles as part of IdentityIQ lifecycle management and compliance solution. SAP data is presented in a familiar hierarchy format that closely represents deployed system resources and organizational structures. New filtering capabilities enable more efficient browsing and selection of SAP data

so tasks can be performed faster. Improved granular support for separation of duty (SOD) violation policies provides flexibility for customers to craft more detailed identity governance policies that include SAP role details such as T-Codes and Authorization Objects.

Configuration

This section contains the following information:

Chapter 1: System Setup

Use System Setup to configure the different options for IdentityIQ. To access System setup options, click the gear icon on the Navigation menu bar and select from the list of options that can include:

- “IdentityIQ Global Settings” on page 5
- “Lifecycle Manager Setup” on page 59
- “Compliance Manager” on page 52

Note: Do not open multiple tabs or browsers. Opening multiple tabs might overwrite changes made in the other.

Note: Because configuration options are based on your deployment, your available options can be different.

IdentityIQ Global Settings

From the Navigation bar, click the gear icon and then select **Global Settings**. Use the Global Setting index page to select the items you want to configure. The following table displays the available options.

Note: You must be a System Administrator to access this page.

Table 1—System Setup Globals Settings Page Descriptions

Page	Description
IdentityIQ	
IdentityIQ Configuration	Set default values for use with notifications, work item policy, object expiration, user interface preferences, and identity history. See "IdentityIQ Configuration" on page 6.
Login Configuration	Set an application other than IdentityIQ for authentication verification and select the automatic identity creation rule. See "Login Configuration" on page 19.
Identity Mappings	Specify the applications, and application attributes, from which the identity data, is derived. See "Identity Mappings" on page 28.
Account Mappings	Specify the account attributes to be used in filters and searches throughout the application. See "Account Mappings" on page 32.
Application Attributes	Define application attributes in addition to those provided by the connectors. See "Application Attributes" on page 37.
Entitlement Catalog Attributes	Define custom extended entitlement attributes and role types. See "Entitlement Catalog Attributes" on page 39.
Quicklink Populations	Configure the quicklink populations for IdentityIQ. See “Quicklink Populations” on page 40.
Forms	Configure forms for workflows, role provisioning policies, and application provisioning policies in IdentityIQ. See "Forms" on page 42.

Table 1—System Setup Globals Settings Page Descriptions

Page	Description
Role Configuration	Define custom extended role attributes and role type. See "Role Configuration" on page 43.
Scopes	Define scopes for use throughout your enterprise. See "Scopes" on page 47.
Time Periods	Define the time periods for use in activity searches. See "Time Periods" on page 49.
Audit Configuration	Specify the actions that are audited and stored in the audit logs. See “Audit Configuration” on page 50.
Electronic Signatures	Configure electronic signatures and their displayed meanings. See "Electronic Signatures" on page 50.
API Authentication	Import files into IdentityIQ. See "API Authentication" on page 50.
IdentityAI Configuration	**Note: This link is only present if you have purchased the IdentityAI product.** Connect IdentityIQ to the IdentityAI product. See “IdentityAI Configuration” on page 51.
Import from File	Import files into IdentityIQ. See "Import From File" on page 52.

IdentityIQ Configuration

Use this page to set default values for use with notifications, work item policy, object expiration, user interface preferences, and identity history. This page contains the following tabs:

- “Mail Settings” on page 6
- “Work Items” on page 7
- “Identities” on page 9
- “Roles” on page 10
- “Password” on page 11
- “Miscellaneous” on page 13

Mail Settings

Table 2— System Setup - IdentityIQ - IdentityIQ Configuration - Mail Settings

Field	Description
Email Settings:	
Email Notification Type	Specify whether to send email using SMTP or to use a redirection email address or file name.
Redirection Email Address	Specify the email address to which email is redirect if Redirecting is selected as the **Email Notification Type**. **Note: This setting is ignored if a Redirection File Name is set. This setting is primarily a test setting.**

Table 2— System Setup - IdentityIQ - IdentityIQ Configuration - Mail Settings

Field	Description
Redirection File Name	Specify the name of the file to which email is redirect if Redirecting is selected as the **Email Notification Type.** **Note: This setting overrides the Redirection Email Address. This setting is primarily a test setting.**
Encryption	Select NONE, SSL, or TLS from the drop-down list.
Default SMTP Host	Specify a default mail host.
Default SMTP Port	Specify a default SMTP port.
Default From Address	Specify the address to be used as the From address for all notices automatically generated by IdentityIQ.
Username and Password	Enter the username and password required to access the SMTP host.
Maximum Email Retries	Specify the maximum number of times to retry sending emails if the SMTP server returns a temporary error. Set this to 0 to disable retries.
Suppress Duplicate Emails	Prevent the sending of multiple emails of the same type to the same recipient at one time. For example, if five work item reminders are sent to the same person at one time, they only receive an email for the first one. This option is enabled by default.
Email Templates:	
Specify the email template that corresponds with each notification type. Email templates are highly configurable.	
Email Task Alerts:	
Specify the configuration parameters in order to receive the status of different tasks after completion. These setting would be applicable in case of the email notification configured at the task level is disabled.	
Email Notification	Select a frequency for email notification to be sent upon task completion. **Disable** — no email notification sent on task completion **Warning** — send an email notification if the task results in a warning **Failure** — send an email notification if the task fails **Always** — always send an email notification upon task completion
Email Notification Template	Select a notification email template (Task Status) from the drop-down list.This option is disabled if **Email Notification** field is disabled.
Email Recipients	The list of users to receive the task completion notification. Use the drop-down arrow to display all identities, or type the first few letters of a name. select names from the list.

Work Items

Table 3— System Setup - IdentityIQ - IdentityIQ Configuration - Work Items Settings

Field	Description
Certification Related Work Item Policy:	

Table 3— System Setup - IdentityIQ - IdentityIQ Configuration - Work Items Settings

Field	Description
Days before expiration	Specify the number of days after which a work item should expire.
Days before expiration to send first reminder	Specify the number of days, before a work item expires, that IdentityIQ should begin sending the owner of that work item reminder notices.
Days between expiration reminders	Specify the frequency with which reminder notices should be sent to the owners of certifications and work items.
Number of notices before escalation	Specify the number of reminder notices that should be sent before the first escalation notice is sent to the manager of the owner of the assess certification or work item.
Send notification email on work item assignment	Select this option to send an email notification when a work item is assigned.
Send notification email on work item assignment removal	Select this option to send an email notification when a work item assignment is removed.
Allow priority editing on work items	Select this option to give work item recipients the ability to adjust the priority level of work items.
Disable forwarding work items from outbox	Select this option to disallow the forwarding of work items from an identities outbox. This option is turned on by default.
Work Item Archives:	
Work item types to archive	Select one or more work item types to be archived. Press the **<Ctrl>** key to select multiple items.
Work Item Rules:	
Inactive user work item escalation rule	Select the rule from the drop-down list for determining a new owner for work items from an inactive user.
Global work item forwarding rule	Use the drop-down list to select the rule used to determine general work item forwarding.
Self-certification work item forwarding rule	Use the drop-down list to select the rule used to determine work item forwarding in special cases. Allows for the specification of a fallback forwarding user in the case that configured automatic forwarding would cause self-certification. This rule only applies if a user has configured a forwarding rule, the rule does not apply when a pre-delegation rule causes self certification.

Identities

Table 4— System Setup - IdentityIQ - IdentityIQ Configuration - Identities Settings

Field	Description
Identity Risk:	
Number of Bands	Specify the number of colored bands, from 2 to 6, to display on all score card charts, graphs, and tables. These bands are used to indicate various levels of risk associated with ranges of Identity risk scores. Specify a number that best meets the needs of your enterprise.
Label	Select the default labels or create your own text label associated with the colored risk bank.
Range	Input the numeric risk score range associated with each risk band. Risk scores are determined by multiple contributing factors defined on the Configure Risk Scoring page. See “Configure Risk Scoring” on page 275.
Indicator	The indication color associated with the risk level.
Identity Attributes:	
Number of searchable attributes	Specify the number of attributes that can be configured for use as searchable attributes on the Identity Attributes page. This can be any number between 1 and 20. The default is 10. **Note: This number should match the number configured during the installation and deployment process. If no customization was performed during the installation and deployment process, the maximum number you can enter is 10.**
Index History Granularity:	
Identity history Group history	The increments at which to store history. For example, if the Identity history is set to ‘Week’, snapshots are preserved on a weekly basis. This means that when a snapshot is taken it overwrites any snapshots taken within the previous week (7 days). Any snapshot that is older than 7 days is saved.
Identity Snapshots:	
Snapshot frequency in days(2 equals every second day)	Specify the frequency with which identity snapshot should be taken. Identity snapshots are used to build the risk score card history that can be used to track trends and patterns for individual users, groups, departments, and your entire organization.
Account Attributes:	
Number of searchable account attributes	Specify the number of attributes that can be configured for use as searchable attributes. This can be any number between 1 and 20. The default is 5. **Note: This number should match the number configured during the installation and deployment process. If no customization was performed during the installation and deployment process, the maximum number you can enter is 5.**
Business Processes:	

Table 4— System Setup - IdentityIQ - IdentityIQ Configuration - Identities Settings

Field	Description
Identity update	Select which business process is executed when an identity is edited in IdentityIQ. This can perform role assignment approvals and send provisioning requests.
Identity refresh	Select which business process is executed when an identity is refreshed in a background task. This might perform role assignment approvals and send provisioning requests.
Identity Correlation	Select which business process is executed when an manual correlation of accounts is done.

Roles

Table 5— System Setup - IdentityIQ - IdentityIQ Configuration - Roles Settings

Field	Description
Role Sunrise/Sunset Dates:	
Enable Sunrise/Sunset Dates on Role Assignment	Enable the ability to set activation and deactivation dates on roles when they are assigned. Activation and deactivation dates can be used to grant temporary access to sensitive roles.
Enable Sunrise/Sunset Dates on Role Activation	Enable the ability to insert activation and deactivation events into roles from the role modeler. Activation events are used to automatically activate or deactivate roles using business processes.
Days before Sunset expiration to send notification	Send a notification to both the requestor and the requestee of the role or entitlement, when access is about to expire. This value determines when the notification is sent. To disable notifications, enter 0. The email template to use for notifications is configured on the **Mail Settings** tab in the **For notice of deprovisioning of sunsetted roles and entitlements** field.
Business Process Editor:	
Role create, update, and delete	Select which business process is executed when roles are created, modified, or deleted in the role modeler.
Schedule role activation	Select which business process is executed when a scheduled role assignment becomes due. This assigns the role and can perform provisioning.
Schedule role/entitlement assignment	Select which business process is executed when a scheduled role assignment or de-assignment becomes due. This de-assigns the role and can perform provisioning.
Additional Role Options:	
Show option to allow multiple application accounts	Enables an option on the Role Management page that enables a role to specify its own target account, or create a new account, during a role request, even if it is required by another role and included in that roles required roles list. If this option is not enabled, required roles are assigned to the same account as the top-level role.

Table 5— System Setup - IdentityIQ - IdentityIQ Configuration - Roles Settings

Field	Description
Show option to allow multiple assignments	Enable an option on the Role Management page that enables a role to be assigned to the same identity multiple times. This option is only available on assignable role types.
Allow multiple assignment for all assignable roles	Make all assignable role types available for multiple assignment to the same identity. This setting supersedes the settings on the individual role definitions.
Allow propagation of role changes	Enables a role change to propagate to all identities that have the role assigned.
Retain assigned entitlements when detected roles are removed	Do not remove assigned entitlements from an identity when a detected role with which they are associated is removed from that identity.
Retain assigned entitlements when assigned roles are removed	Do not remove assigned entitlements from an identity when an assigned role with which they are associated is removed from that identity.

Password

Use this tab to define the password policy for IdentityIQ. All of the users must set up their passwords based on the policy created on this tab.

Use the Define Character Types dialog to define a custom set of character that are allowed in passwords. These can be used to match password requirements for specific application types. Click **Define Character Types** to open the dialog and enter character sets by category, such as **Digits**, **Uppercase Characters**, **Lowercase** or **Non-English Characters**, **Special Characters**. All characters are allowed if these fields are empty.

For additional information on password management, see “Password Management” on page 479.

Table 6— System Setup - IdentityIQ Configuration - Password Settings

Field	Description
Configuration:	
Enable one-way hashing of secret values	**Note: You must run the Encrypt Sensitive Data Task after selecting this option to convert any saved values from encrypted to hashed.** These values include passwords, password history, and authentication questions. When this option is enabled, specific password policy options are disabled. When this option is selected, all values are hashed instead of encrypted. For more information, see “Data Encryption” on page 141.
Number of hashing iterations	The number of iterations performed in the hashing algorithm.
Password Policy:	

Table 6— System Setup - IdentityIQ Configuration - Password Settings

Field	Description
Minimum number of characters	The minimum number of characters, letters or digits, required for a valid password.
Maximum number of characters	The maximum number of characters, letters or digits, allowed in a valid password.
Minimum number of letters	The minimum number of letters required for a valid password.
Minimum number of character type constraints to meet	The minimum number of character types required for a valid password. Applicable character types are upper case, lower case, digits, and special characters. If no value is set, all of the character type constraints must be met.
Minimum number of digits	The minimum number of digits required for a valid password.
Minimum uppercase letters	The minimum number of uppercase letters required for a valid password.
Minimum lowercase letters	The minimum number of lowercase letters required for a valid password.
Minimum special characters	The minimum number of special characters required.
Number of repeated characters allowed	The maximum number of consecutive repeated characters allowed in a valid password. For example, if this option is set to 2, “cloudd” and “cclooud” is valid, “clouddd”, “cloooud” and “cccloud” are invalid. For “cclooudd” invalid password, the following error message is displayed: Password should not contain more than 2 occurrence(s) of the repeated characters. For “Clouddd” invalid password, the following error message is displayed: Password should not contain more than 2 consecutive repeated characters
	The maximum number of occurrences of repeat characters allowed in a valid password. For example, if this option is set to 1, “happy123” is valid, however, “happy123dd” and “happpy123” are not.
Password history length	The number of previous passwords stored by IdentityIQ. This number includes the current password so if the length is two, the history is the current password and one other. If the length is set to zero there is no history.
Triviality check against old password	Ensure that the shorter of the old and new password is not a substring of the other. Both passwords are changed to upper case prior to the check.
Minimum number of characters by position	The minimum number of unique characters by position for the new password. Can be used to ensure that not just the first or last character is changed. Select **Case sensitive check** to ensure that more than just the case is changing in the new password.

Table 6— System Setup - IdentityIQ Configuration - Password Settings

Field	Description
Days until expiration for manually set passwords	The number of days until a password set manually expires. If the days are zero passwords do not expire.
Days until expiration for generated passwords	The number of days until a password set by the identity create rule during aggregation expires. If zero the days are zero passwords do not expire.
Minimum Hours between password changes	The minimum number of hours that must past before a user's password can be changed again.
Validate passwords against the password dictionary	Ensures that the password to be created is unique.
Validate passwords against the identity's list of attributes	Check the new password for validity against the attributes assigned to the identity.
Require users to enter their current password when setting a new password	Require users to enter there current password before creating a new password.

Miscellaneous

Table 7— System Setup - IdentityIQ - IdentityIQ Configuration - Miscellaneous Settings

Field	Description
Other Object Expirations:	
Days before snapshot deletion	Specify the number of days to keep an identity snapshot in the system before it is deleted. Identity snapshots are used to build history.
Days before task result deletion	Specify the number of days to keep task results on the Task Results page before removing them from the system.
Days before certifications are archived	Specify the number of days after which to archive certifications. Leave the settings at zero (0) to never archive certifications. **Note: Certification archives are not visible from the IdentityIQ GUI. It is recommended that you do not change the default setting at this time.**

Table 7— System Setup - IdentityIQ - IdentityIQ Configuration - Miscellaneous Settings

Field	Description
Days before certification archive deletion	Specify the number of days to maintain the certification archive before deleting certifications records. Leave the settings at zero (0) to never delete certifications archives. **Note: Certification archives are not visible from the IdentityIQ GUI. It is recommended that you do not change the default setting at this time.**
Minutes before object locks are released	Specify the number of minutes to elapse before releasing an object lock. Leave the settings at zero (0) to have no time delay when objects are released.
Days before provisioning request logs expire	Specify the number of days to maintain provisioning request logs before deleting them. Leave the settings at zero (0) to never delete provisioning request logs.
UI Preferences:	
Disable Role Modeler Tree View	Disable the tree view on the Role Manager page. Disabling the tree view might enhance performance on that page.
Maximum Roles Page Size	The maximum number of roles to display per page on the **Role Management** page.
Show unsupported browser message	Display a message when an unsupported browser is used.
Accessibility: Color Contrast	Enable color contrast throughout the entire IdentityIQ instance.
Syslog Settings:	
Enable syslog	Enable the syslog.
Level at which syslog events will be stored	Select the lowest level of even which is stored in the syslog. Choose from FATAL, ERROR, and WARN.
Days before syslog event deletion	Input the number of days an event in the syslog must remain before becoming eligible for purging.
Provisioning Transaction Log Settings	
Enable Provisioning Transaction Log	Enable the Provisioning Transaction table and begging logging all, or some, of the provisioning action within IdentityIQ.
Maximum Log Level	The level at which transactions are logged based on their completion status. Success — all transaction are logged Retry — transactions that did not succeed and are in either the retry or failed state Failure — only log transaction that have failed and are setup for retry

Table 7— System Setup - IdentityIQ - IdentityIQ Configuration - Miscellaneous Settings

Field	Description
Days before provisioning transaction event deletion	The number of days before a provisioning transaction is removed from the table.
File Preferences:	
Temporary Directory	Input the path to a default temporary director for use by IdentityIQ. This is the directory where IdentityIQ stores temporary files, such as log files, during processing.
System Help Settings:	
Help Contact Email Address	Input an email address of a user responsible for supporting IdentityIQ in your enterprise. The email account is accessible from an **Email Help** button displayed at the bottom of some pages.
Localized Object Attributes:	
Allow applications to be configured with multi-language descriptions	Allow applications to be configured with multi-language descriptions. See “Multi-language Description Files” on page 17.
Allow roles to be configured with multi-language descriptions	Allow roles to be configured with multi-language descriptions. See “Multi-language Description Files” on page 17.
Allow policies to be configured with multi-language descriptions	Allow policies to be configured with multi-language descriptions. See “Multi-language Description Files” on page 17.
Allow entitlements to be configured with multi-language descriptions	Allow entitlements to be configured with multi-language descriptions. See “Multi-language Description Files” on page 17.
Multi-Languages Descriptions: **Note: You must add all supported languages to the <locale-configure> section of the faces-config.xml file before the application properly recognizes the languages.**	
Default Language	Select the language to use as a default from the list of supported languages.
Supported Languages	Enter the languages that your instance of IdentityIQ supports.
Business Processes:	
Entitlement Update	Select the business process to execute when a managed entitlement or group is created or edited.

Table 7— System Setup - IdentityIQ - IdentityIQ Configuration - Miscellaneous Settings

Field	Description
Password Intercept	Select the business process to execute when a password change interception event is received.
Caches:	
Enable asynchronous policy and role cache refresh	Disable the immediate cache refresh with each Lifecycle Manger request. When you enable this option, IdentityIQ does not check for changes to policy and role objects. When a Lifecycle Manager request is submitted, the cache is refreshed immediately. Using this option can speed the request process. However, the effects of a recent policy or role change might not display for a few minutes.
Reports:	
CSV Delimiter	The character used as the CSV delimiter when exporting report results. Comma is used by default.
Plugin Settings:	
Prohibit scripts from accessing plugin-loaded classes	**Note: All beanshell executions are referred to as scripts.** Restrict the access to classes loaded by plugins. Without this restriction, all class are available in IdentityIQ.
Relax strict declaration enforcement	Enable IdentityIQ to work fully with plugins that were created without explicitly declaring classes for export. By default, for a fresh installation of IdentityIQ this option is not selected. For an upgraded installation of IdentityIQ, this option is selected if plugins exist.
Attachment Settings: **Note: Attachments are only allowed on single-user requests.** **Note: Attachments are only available for manual access requests.** See “Access Request Attachments” on page 18 for more information.	
Enable Attachments	Enable the attachments feature. Allow users to add attachments to access requests.
Maximum file size (MB)	Maximum file size for any single attachment up to 20 MB. Maximum attachment size limits can be adjusted by a system administrator using the `attachmentsMaxFileSizeLimit` key in the system configuration file.
Supported file types	Comma separated list of file types. The dot prefix is not required.
Configuration Rules	**Note: Only the rules selected in this list are run during an access request.** This list contains all of the attachment configuration rules available in your installation of IdentityIQ. Use the Ctrl or Shift keys to select multiple rules.

Privileged Account Management

The SailPoint IdentityIQ Privileged Account Management Module (PAM) extends identity governance processes and controls to highly privileged access, enabling you to centrally manage access to privileged and non-privileged accounts.

Talk to your SailPoint representative or refer to the SailPoint IdentityIQ Privileged Account Management Module Guide for more information.

Multi-language Description Files

Some escaped HTML characters are not recognized and do not display in descriptions if they are formatted using those characters. You must ensure that all files are formatted correctly before importing them into IdentityIQ and referencing them from the product. Use the following examples to format the HTML correctly:

```
test (to appear in bold) - <b>test</b>
<test> - &lt;test&gt;
<test> (to appear in bold) - <b>&lt;test&gt;<b\>
<<test>> - &lt;&lt;test&gt;&lt;
“test” - &quot;test&quot;
‘test’ - ‘test’
&test - &test
```

Rule Editor

The Rule Editor page enables you to edit any existing rule to your specifications. Click the “**...**” icon next to a rule drop-down list to access the rule editor throughout IdentityIQ. Choose to either create a new rule, or edit an existing rule structure.

The Rule Editor panel includes the following items:

Table 8—Rule Editor Panel Field Descriptions

Option	Description
Copy from an existing rule	Select an existing rule from the drop-down list. This option is available if you did not select a rule from the drop-down list on the previous page.
Code input area	Field where code is input. IdentityIQ recognizes BeanShell programing language. You can edit code from an existing rule or create a new one from scratch.
Description	Enter the description of your new rule.
Rule Name	Enter the name of your rule.
Rule Type	Non-editable field which displays the type of rule (for example, Violation).
Return Type.	Non-editable field which displays the type of return (for example, PolicyViolation).
Arguments.	Non-editable field which displays the arguments used in the rule (for example, log, context, state, etc.).
Returns.	Non-editable field which displays the type of return the rule executes (for example, Violation).

When you have completed your rule edits, click **Save** to return to the previous page. The new rule is now available from the drop-down list.

Access Request Attachments

Note: Attachments are only allowed on single-user requests.

Note: Attachments are only available for manual access requests.

The attachments feature enables users to add attachments to single user access requests. For example you could attach training certificates or a notarized document of authorization.

By enabling attachments on the Configure IdentityIQ Settings -> Miscellaneous tab, you are enabling, but not requiring, any user to add an attachment to any single user access request. When the feature is enabled, requests display the attachment icon, paper clip, on each item in a request, but the icon is only active if an attachment is allowed for that item. When you click the icon, the attachment overlay is displayed and you can add an attachment by dragging and dropping or uploading a file.

Attachments are controlled through AttachmentConfig rules. If there are no AttachmentConfig rules for an item, or they all have null or empty prompts, the attachment overlay contains no additional information.

Attachment Configuration

Attachments are controlled through the AttachmentConfig rules. Each of these rules is run with every request made. Use the AttachmentConfig rules to require attachments for specific access request scenarios and customize the prompts displayed on the attachment overlay. When an attachment is required, the word required is displayed with the attachment icon and an error is displayed if a request is submitted without an attachment.

Activate the attachment configuration rules to run with access requests by selecting them from the **Configuration Rules** list on the Global Settings -> IdentityIQ Configuration -> Miscellaneous tab under the gear icon.

Import attachment configuration rules using the System Settings -> Import from File page under the gear icon.

To remove an attachment configuration rule from IdentityIQ, first de-select that rule from the **Configuration Rules** list and then delete the rule object.

These rules can be as simple or complex as the needs of your organization require.

These rules contain the following inputs:

- requestor — the user making the request
- requestee — the user for whom the request is being made
- requestItem— the item being requested
- action — the request action (add or remove)

Each attachment configuration rule is run once for each item being requested and returns a list of configuration objects.

The fields of an attachment configuration object are:

- required — boolean (true, false) where true means an attachment is required
- prompt — string – the prompt that is displayed in the attachment overlay when attaching files to this request item

Multiple attachment configuration objects can be associated with a single request item. In this case, the prompt strings are concatenated on the attachment overlay.

A file containing an example of attachment configuration rules is included in the IdentityIQ installation package. The `exampleattachmentconfigrules.xml` file is located in the `IdentityIQ_HOME/WEB-INF/config` directory.

Prune Unassociated Attachments

In rare cases attachments that are not associated with an access request might end up getting loaded into the database. IdentityIQ provides two system maintenance task to prune those attachments and clean them out of your database. See “Tasks” on page 283 for more information on using the System Maintenance and System Maintenance Object Prune tasks.

Login Configuration

Use the Login Configuration page to set an application for authentication verification. For example, if all of the users in your organization are set up with roles and authorization in an LDAP server, use that server to verify users logging into IdentityIQ. Login Configuration has the following tabs:

- “Login Settings” on page 19
- “User Reset” on page 21
- “Multi-Factor Authentication” on page 23
- “SSO Configuration” on page 26

Authentication Method Processing Order

IdentityIQ attempts to authenticate users by all enabled methods before reporting login failure. The methods are executed in this order, skipping any disabled methods:

Note: If configured, Multi-Factor Authentication follows the initial user authentication through any of these means.

1. Single Sign On (Rule-based or SAML)
2. Pass-Through Authentication
3. Internal IdentityIQ Authentication

Login Settings

Use the Login Settings tab to configure general settings for login criteria.

Note: Any user discovered by an aggregation task displays in the identities lists and can be assigned work items. Before a user can access IdentityIQ and the work item, they must be validated by an authentication verification server.

Use Auto create user rules when adding users to the application. The first time a user logs into the application, and is verified by the pass-through server, the **Auto create user rules** creates an IdentityIQ user based on specifications defined in this rule. Those rules are applied each time the user accesses product.

The following table describes the login settings.

Table 9— System Setup - IdentityIQ - Login Configuration - Login Settings

Field	Description
Pass through application	Specify an application to use as the authentication verification server for all users logging into IdentityIQ.
Auto create user rule	Specify an auto create user rule to use when creating IdentityIQ identities based on account attributes discovered during aggregations. Note: Click the “...” icon to launch the Rule Editor to make changes to your rules if needed. See "Rule Editor" on page 17

Table 9— System Setup - IdentityIQ - Login Configuration - Login Settings

Field	Description
Login error style	**Note: If you select Simple and are using the Lockout feature, users that are locked out do not receive a message providing that information.** Select a login error message style. **Simple** — shows an error with no information about what is incorrect. **Detailed** — provides information about the incorrect part of the login. For example, Invalid password for user admin.
Login after timeout returns to	Specify how navigation is handled after a session times out and you log back in to that session. If checked, the Home page is displayed. If not, the session returns to the page that was viewed at the time of the timeout.
Enable Authorization Lockout	Enable a lockout period for users who enter the wrong authorization information. Use the options that display to set the lockout parameters. **Note: This option is only associated with the IdentityIQ password. It does not apply to the pass through authentication application. For example, if a user is locked out of directly logging into IdentityIQ, but they enter the correct information on the pass through authentication server, they are allowed into the application.**
Number of Unsuccessful Login Attempts before lockout	Specify the number of login attempt failures allowed before the user is locked out of IdentityIQ.
Number of minutes a user will be locked out due to unsuccessful login	Specify the number of minutes a user is locked out of IdentityIQ before they can attempt to login again.
Enable Protected User Lockout	Select to lockout a protected user upon authorization failure the same as all other users.

User Reset

User Reset has the following options:

- Enable Authentication Questions — displays a **Forgot Password** link on the Login page and uses answers to pre-defined questions to authenticate a user's identity
- Enable SMS Reset — displays a **Forgot Password** link on the Login page and uses Short Message Service (SMS) to send the user a text message with a verification code

Authentication Questions

Note: The Authentication questions and settings are associated with the password set on the password application. These are not associated with a direct logon to IdentityIQ.

Authentication questions confirm a user's identity if they have forgotten their IdentityIQ password and the environment is configured to enable the question authentication feature. Question authentication is enabled using the **Enable Forgot Password** select box on the Login Settings tab.

These questions display when you click the **Forgot Password** link off of the Login page during the authentication process.

The Questions list can contain tags from the properties file configured when your IdentityIQ instance was deployed, text entered directly on this tab, or a combination of both. Mapping tags from a properties file is generally used for internationalization purposes.

Click the plus (**+**) icon to add a new question and the minus (**-**) icon to remove a question. You can enter as many questions as you deem necessary. A user who forgets their password must answer the designated number of the questions in the list. The number of questions a user must answer for authentication is defined in the Settings section below.

Use the **Settings** section to configure behaviors for password attempts.

SMS Reset

Before you set up SMS Reset, you need the following items from twilio.com:

- an active Twilio account
- Twilio ID
- Twilio credentials (authentication token)
- From phone number configured on account

The following table describes the password reset settings.

Table 10— System Setup - IdentityIQ - Login Configuration - Password Reset Settings

Field	Description
Authentication Question Configuration:	
Number of questions asked to authenticate an identity	Specify the number of questions that must be answered correctly in order to reset the password.
Number of authentication answers a user must have defined in IdentityIQ	Specify the number of authentications that must provide to set up password reset.
Prompt users for answers to unanswered authentication questions upon successful login	Adds an extra layer of security to logon screen. Select to have users prompted for answers until they define the required number, as defined in Edit Preferences page or if questions are added or changed. When enabled, users are automatically redirected to the Answer Authentication Questions page upon successfully entering user name and password.

Table 10— System Setup - IdentityIQ - Login Configuration - Password Reset Settings

Field	Description
Maximum number of unsuccessful authentication attempts before IdentityIQ lockout	Specify the number of failed authentication answer attempts before the user is locked out of IdentityIQ. **Note: After the maximum number of unsuccessful attempts, the SMS token is no longer accepted and the user must request a new code.**
Number of minutes a user will remain locked out due to unsuccessful authentication	Specify how long a user is locked out after the specified number of failed authentication question answer attempts is exceeded. A user with the proper capability can overwrite the lockout period.
SMS Reset Configuration:	
Twilio Account ID	Enter the account ID you receive from Twilio when you set up your company Twilio account.
Twilio Authentication Token	Enter the authentication token you receive from Twilio when you set up your company Twilio account.
'From' Phone Number	Specify the phone number to use for sending the SMS message. **Note: This phone number must be configured as the from number on your Twilio account.**
Phone Number Attribute on Identity	Select the identity attribute that represents the mobile phone number. To define a new identity attribute, see "How to Add or Edit Account Attributes" on page 34. **Note: For a user to reset their password using the SMS Reset feature, the field associated with their mobile phone number must contain a complete number including the area code. Using E.164 number formatting for all phone numbers in the "To" and "From" fields is strongly encouraged.** For more information, see "SSO Configuration" on page 26.
Reset Token Timeout (minutes)	Specify how long the user's reset token is valid (in minutes).

Multi-Factor Authentication

Multi Factor Authentication (MFA) adds an additional layer of security by requiring users to use multiple methods to authenticate their identity before they can log in to IdentityIQ. IdentityIQ supports the following MFA options:

- RSA Workflow
- Duo Workflow

To access MFA Login Configuration settings in IdentityIQ, click the **gear** icon in the menu bar and select **Global Settings > Login Configuration > Multi Factor Authentication** tab

This section includes the following topics:

MFA Prerequisites

Note: To use Duo, you must follow the Duo Auth API instruction to enable the AuthAPI in the Duo Admin Panel. To locate the Duo API documentation, go to https://duo.com, search for Auth API documentation and then follow the steps for adding Duo two-factor authentication.

MFA User Process Flow Overview

The basic process flow for using MFA to log in to IdentityIQ includes the following steps:

1. The user enters a valid username and password at the IdentityIQ login screen.
2. The IdentityIQ MFA workflow begins and displays the MFA provider's login page or process for login. If a user is assigned to multiple providers, the user must select a provider from the provider list before proceeding to the provider's login page.
3. The user completes the authentication process for their MFA provider.
4. The user is logged in to IdentityIQ and the Home page displays.

MFA Configuration Process Flow Overview

The basic process flow for configuring MFA for IdentityIQ includes the following steps:

1. Use a pre-defined MFA workflow or choose to create a custom workflow.
2. Install the workflow.
   - Import the workflow.
   - Configure the workflow as a business process.
   - Enable the populations to use with MFA.
3. Save your MFA configuration.

For more information, see "How to Install a Multi-Factor Authentication Workflow - DUO Example" on page 24

Multi-Factor Authentication Workflows

Each MFA provider has its own flow and process. MFA Providers contain the populations and providers are configured from an existing list of DynamicScopes/Populations. Workflows of type **MultiFactorAuthentication** can enable Multi-Factor Authentication for a particular provider.

Pre-defined workflows are provided. These workflows use existing pre-configured applications to perform Multi-Factor Authentication

You can choose to create a custom workflow. See "Custom Multi-Factor Authentication Workflows" on page 26.

How to Install a Multi-Factor Authentication Workflow - DUO Example

The following workflows are provided, however they are not installed by default. These workflows use existing pre-configured applications to perform Multi-Factor Authentication. The provided workflows are located:

- `WEB-INF/config/workflow_MultiFactor_DUO.xml`
- `WEB-INF/config/workflow_MultiFactor_RSA.xml`

Note: The following instruction are specific to using DUO as your MFA application. You can use these instructions to install RSA by changing the DUO-specific items to RSA. As noted in the instructions, you do not need to add API authentication credentials for RSA.

1. Review any prerequisites. See "MFA Prerequisites" on page 23.
2. To import the workflow, you can use the **Import From File** function in the **Global Settings** menu or use the IdentityIQ console. To use the IdentityIQ console, open the console and use the following command:

   ```
   import workflow_MultiFactor_DUO.xml
   ```

3. Configure the workflow as a business process:

a. Login to IdentityIQ using an administrator account and navigate to **Setup > Business Processes**.
b. Click the workflow named **MFA DUO**
c. Click **Process Variables**
d. Select a pre-configured application, of type **Duo**, for the field **Duo Application Name**. The workflow reads new properties added to the application used to authenticate with the Duo cloud Authentication service.
e. Click **Save**.

4. Use the following steps to add Duo authentication API credentials:

   IMPORTANT! These steps are not necessary for the MFA RSA workflow because RSA uses existing API credential information already configured in the RSA Application.

   a. Navigate to **Applications > Application Definition**.
   b. Select the Application of type Duo you configured in the previous step.
   c. Click **Configuration**.
   d. Complete the **Admin API Credentials** section using the credentials you obtained from the Duo Admin Panel.

      The first time you set up a Duo application, you must enter the Admin API information received from Duo after you completed Step 4. in “How to Install a Multi-Factor Authentication Workflow - DUO Example” on page 24. If you are modifying a previously configured Duo application, the Admin API credentials should already be configured.
   e. Click **Save**.

5. Next, enable a population of users that must use Multi-Factor Authentication to authenticate using the following steps:
   a. Click the **gear** icon.
   b. Navigate to **Global Settings > Quicklink Populations**.
   c. Verify you have an existing population of users you want to authenticate using Multi-Factor Authentication.

6. The population you enabled can allow a user in the population to request access for other users. If you do not want a user have that capability, you can create a new QuickLink population. You must select **No one** in the section **who can members request for?** when you create the new QuickLink population. This configuration separates Request Access type Quicklink Populations from Multi-Factor Authentication Populations.

7. Next, associate the population to the Multi-Factor Authentication workflow using the following steps:
   a. Click the **gear** icon and navigate to **Global Settings > Login Configuration > MFA tab**.
   b. Check the box for the MFA Workflow you want to enable.
   c. Add any populations to the multi-select list you want to enable for this MFA workflow.
   d. Click **Save**.

Custom Multi-Factor Authentication Workflows

Implementors can create custom Multi-Factor authentication workflows. Any workflow of type **MultiFactorAuthentication** displays in the MFA Configuration page. If you choose to create a custom workflow, review the following information:

- Adding an error message to the workflow case using:

  ```
  wfcase.addMessage(new Message(Type.Error, "An error has occurred that prevents
  Multi-Factor Authentication"))
  ```

  This adds an error to the workflow case and signals to the Multi-Factor framework the user should not be logged in.
- A workflow that was not marked **complete** will signal Multi-Factor authentication has failed. During normal workflow execution, if a workflow has not produced an error, the workflow is automatically marked complete.

SSO Configuration

IdentityIQ supports two different options for single sign-on (SSO) configuration, rule-based and SAML. SSO streamlines the login process for users even further than pass-through authentication by enabling the user to bypass signing in to each system, once they have completed the initial sign-on to the authenticating application.

SSO Configuration has the following options:

- Enable Rule-Based Single Sign-On (SSO) — uses rules for Single Sign-On and Validation
- Enable SAML Based Single Sign-On (SSO) — uses Security Assertion Markup Language (SAML) as an authentication protocol

  Note: To access the IdentityIQ Login page directly when Single Sign-On is configured, use a supported browser and enter http://<iiq server>/spt/login.jsf?prompt=true.

IdentityIQ supports specifying both types of SSO in the same installation's login configuration. The order in which they are consulted during user authentication will be determined as follows:

- If an ssoAuthenticators attribute is specified in the SystemConfiguration object, it will specify the configured SSO options in a CSV list, and the options will be checked in the order they are specified
- If that attribute is not present, SAML SSO will be used first and then rule-based SSO

Rules-Based SSO

In rule-based Single Sign-On (SSO) configurations, when the user accesses the IdentityIQ web application, the authentication source recognizes it as a secure resource, requires the user to authenticate to it (if the user has not already done so), and passes a “token”, containing contextual information, in the HTTP header to IdentityIQ. The SSOAuthenticationRule validates that information and maps the user to the appropriate IdentityIQ Identity.

SAML-Based SSO

In SAML SSO, the authorization request can be initiated with the Service Provider (the application itself - IdentityIQ) or with the SSO authentication application (known as the Identity Provider). In either case, the Identity Provider handles authentication of the user and provides a signed XML <Response>, or Assertion. This response contains information that IdentityIQ can match to an identity to determine the user's proper authorization to IdentityIQ functionality.

IdentityIQ has the following SSO configuration areas:

- Identity Provider (IDP)
- Service Provider (SP)
  - Identity ID / Issuer — string that represents how each side (IDP or SP) refers to itself
  - Login URL (also known as the SSO Service or SSO Login URL) — the URL on the IDP which understands SAML.
  - Public X.509 Certificate — the X509 certificate public key of the EDP.
- Assertion Consumer Service (ACS) — URL on the SP which understands SAML

Note: You can have an IDP EntityI D / Issuer and a Service Provider (SP) Entity ID / Issuer. Both IDs are very important for SAML 2.0 flows.

The following table describes the SSO settings.

Table 11— System Setup - IdentityIQ - Login Configuration - SSO Settings

Field	Description
Rule Based SSO:	
Single Sign-On Rule	Specify the rule to use when authorizing users through and single sign-on system, such as SiteMinder. **Note: Click the "..." icon to launch the Rule Editor to make changes to your rules if needed.** See "Rule Editor" on page 17.
Single Sign-On Validation Rule	Specify the rule to use to verify a single sign-on session to make sure a stale session is not actually a different user. The rule type (SSOValidation) runs on every request. If the request is valid, it returns null. If it returns a string, that string is an indication of an error and is used in the error that is displayed in the logs. If the session is invalidated, the request is redirect to logoutUrl configured in web.xml. This is designed to be used with the Single Sign-On Rule.
SAML Based SSO: Identity Provider Settings	
Entity ID / Issuer	Unique identifier defining the organization (IdentityIQ) to the IdP. This ID is usually the URL or domain name for the organization. For example, `https://identityiq-server.your-domain.com:your-port/`identityiq **Note: If you use the standard https port for communication, the :your-port is not necessary.**
SSO Login URL	IdP SAML SSO URL. Specify the url of the IdP SSO service provider. You can configure IdentityIQ to interact with other Identity Providers (IdP)s. You can obtain this address from your IdP. **Note: If the Identity Provider Issuer is not set, the configuration default to Identity Provider Single Sign-On service URL.**

Table 11— System Setup - IdentityIQ - Login Configuration - SSO Settings

Field	Description
Public X.509 Certificate	Select the NameId Format specified by the IdP.
Identity Provider Issuer URL	Specify the Unique Identifier that defines the IdP to IdentityIQ. This identifier is often in the form of a url but does not have to be. **Note: The Identity Provider Issuer URL field is only necessary if the SAML response does not contain an Issuer value that does not match the leading characters of the Identities Provider SSO Server URL field. For example,** Identity Provider SSO Server URL = `https://idp.your-domain.com/SSOApp/SSOLogin` SAML Response Issuer field = `https://idp.your-domain.com/SSOApp`
SAML Based SSO: Service Provider Settings	
SAML Response URL (Assertion Consumer Service)	Specify the IdentityIQ url where the SAML is to be accepted. For example, `https://identityiqserver.your-domain.com:your-port/identityiq/home.jsf`
Binding Method	Select **HTTP POST** or **HTTP Redirect** for the communications scheme.
NameId Format	Select the name format from the list. The IdP provides the formats listed in drop-down box.
SAML Correlation Rule	Select a rule to use to match a SailPoint identity with IdP results.

Identity Mappings

Use the Identity Attributes page to view and edit the identity attributes information for your configuration. These attributes are used throughout the product for certifications, searches, and to collect and correlate identity data from applications.

IdentityIQ also supports the use of Robotic Process Automation (RPA) or bot, an application that can perform automated tasks, especially simple, repetitive tasks such as requesting access and managing identities.

Bots require effective governance just as traditional identities do:

- The need to manage bots in your organization or under your control. You need to be able to see all the bots, along with their access and have the ability to add, remove access to bots.
- Your organization might have bots that do password resets for certain populations in the organization. You need to make sure that the bots have the right access and are the right version to do their job.
- The need to show auditors that your organization has owners who are accountable for managing bots, as part of certification.
- The need for an ability to define policies to ensure that bots do not get too much access.
- The need for an ability to define lifecycle events on bots, so that you can enforce controls on when bot access changes or when bots are retired.

IdentityIQ's governance capabilities for bots includes the abilities to:

- Manage bots and their attributes
- Request access for bots
- Certify bots

The Identity Attributes page contains the following information:

Table 12— System Setup - IdentityIQ - Identity Mapping - Identity Attributes Descriptions

Column	Description
Attribute	The display name of an identity attribute derived from the attribute and its associated application in the Primary Source Mapping column. The following attributes are required by IdentityIQ to perform correctly: ID manager email firstname lastname **Manager** and **role** are system attributes that are configured for grouping. However, you can use any identity attribute or grouping by defining it as a group factory in the Advanced Options.
Primary Source Mapping	The first of the list of application/attribute pairs from which employee attributes are derived. If the required data is unavailable on this primary source, the collection process continues down the list of configured sources until the information is found. Set up the list of sources on the Edit Identity Attributes page. **Note: Setting the same application and attribute as the source and target for an identity attribute creates circular references.** **Identity attributes with circular references between sources and targets can cause values to be continually changed on every attribute synchronization. This can be problematic when a transformation rule modifies a value without first checking the identity attribute value has already been transformed.**
Advanced Options	The advanced options that are enabled for this attribute. Editable — the attribute can be edited. **Group Factory** — the attribute can be used to create groups that are used for analytical purpose throughout IdentityIQ. **Searchable** — the attributes that are available for filtering in identity searches.

To delete identity attributes, right-click the attribute and select **Delete**.

Note: Deleting an identity attribute also deletes any group factories that reference it. Review the group factory information in the Confirm Deletion of Attribute dialog before clicking Yes.

Edit Identity Attributes Page

Use the Edit Identity Attribute page to create and edit identity attributes including the display name, advanced options and source mapping.

The maximum number of searchable attributes you can create is defined during the application installation and configuration process and controlled from the System Setup pages. The default number is ten (10). See "Create Icons to Represent Specialized Account Attributes" on page 36.

To support the governance of bots, IdentityIQ has three new standard attributes in the identity object that enable you to do things like run a focused certification on just bots.

The attributes are:

- **Type**: an attribute to define the type of identity. The standard values for this attribute are:
  - Employee
  - Contractor
  - External / Partner
  - RPA / Bots
  - Service Account

  However, you can define your own types in addition to these 5, via editing XML in debug
- **Version:** an attribute to indicate what version of software the bot is using. This attribute is intended to be used only for bots.
- **Administrator:** the owner, certifier, of the bot. This is used instead of manager for bots throughout IdentityIQ.

The Edit Identity Attribute page contains the following information:

Table 13— Edit Identity Attributes Page Field Descriptions

Field	Description
Identity Attribute:	
Attribute Name	The name of the attribute as it is used throughout IdentityIQ. For example, this the name used rules.
Display Name	The IdentityIQ user assigned name.
Advanced Options:	
Attribute Type	Select from the following attribute types: **String** — creates a text-editable field. **Identity** — creates a drop-down list from which you choose an existing identity.

Table 13— Edit Identity Attributes Page Field Descriptions

Field	Description
Edit Mode	Enable editing of this attribute from the Identity pages. **Read Only** — this attribute cannot be edited from the Identities pages. **Permanent** — changes made on the identities pages are not overwritten by refresh tasks. **Temporary** — changes made on the edit identities pages are overwritten when an aggregation task brings over a new (changed) value for the attribute.
Searchable	Enable this attribute for use in searches and filtering through IdentityIQ.
Multi-Valued	Specify attributes for which multiple values might be returned during aggregation. Attributes flagged as multi-valued are stored as a list. Even objects that have a single value for a multi-value attribute are stored as a single-item list. Multi-valued attributes are used for queries throughout the product.
Group Factory	Enable this attribute for use in creating groups used for analytical purpose throughout IdentityIQ.
Value Change Rule	Specify a rule to run every time a change is detected on this attribute during the aggregation process. For example, a rule can be written to send change notifications, request change approval or launch a certification. Click the "..." icon to launch the Rule Editor to make changes to your rules if needed. See "Rule Editor" on page 17
Value Change Workflow	Specify a business process to run every time a change is detected on this attribute during the aggregation process. For example, a business process can be written to send change notifications, request change approval or launch a certification.
Note: If you set the source and target mapping to the same application/attribute pair, it creates circular references and where the values continuously change with every attribute synchronization.	
Source Mappings: The list of application/attribute pairs from which employee attributes are derived. If the required data is unavailable on the primary source, the collection process continues down the list of configured sources until the information is found.	
Target Mapping (Only available for Identity attribute types): When creating or editing an Identity attribute, use the Target Attribute options to define targets that the basis for attribute synchronization. Click **Add Target** to display the Add a target to the AttributeName attribute dialog, and complete all of the information.	

How to Add or Edit Identity Attributes

1. Click **Add New Attribute** or click an existing attribute to display the Edit Identity Attribute page.
2. Enter or change the attribute name and an intuitive display name.

   Note: You cannot define an extended attribute with the same name as any existing identity attribute.

   Note: Changing an attribute name might cause attributes that were previously aggregated to no longer be recognized.
3. **Optional:** Enable or change the Advance Options.
4. Click **Add Source** to display the Add a source dialog.

5. Specify a source for the new attribute.

   Map directly to an attribute on an application.

   For Application Attributes you have the option to also make this source a target for attribute synchronization. If there are multiple source applications on which a user might have accounts, you would likely want to push the most authoritative value to the rest of the accounts.

   a. Select **Application Attribute**.
   b. Select an application from the **Application** drop-down list.
   c. Select an attribute from the **Attribute** drop-down list.

   Map to an application rule. This rule only applies to the application specified.

   a. Select **Application Rule**.
   b. Select an application from the **Application** drop-down list.
   c. Select a rule from the **Rule** drop-down list.

   Map to a global rule. This rule applies to all applications that contain this attribute.

   a. Select **Global rule (all apps)**.
   b. Select a rule from the **Rule** drop-down list.

6. Click **Add** to add the new source.
7. Use the arrows to the right of the sources list to rearrange the search order for the attribute sources. When aggregation tasks are run they search the source at the top of the list, or the primary source, first and then work down the list.
8. For Identity attribute types only, add targets for attribute synchronization:
   a. Select **Add Target** to display the Add a target to the attribute dialog.
   b. Select the application to receive the value.
   c. Select the attribute to receive the value.
   d. **Optional:** Select a transformation rule to transform the value before it is set on the destination.
   e. **Optional:** Select Provision All Accounts to provision all of the identities accounts on the targeted application. If you disable this option you are asked to select the accounts to provision manually.
9. Click **Save** to create the new attribute and return to the Identity Attribute page.

Account Mappings

Note: Extended attribute names must be unique. Extended attributes cannot share a name with any other attribute in any other application schema.

Use the Account Mapping page to setup and map specialized accounts. Specialized accounts can be any accounts that justify special handling throughout your enterprise. For example privileged accounts such as Root, Administrator, or Super User, and service accounts that access a specific service or function on an application. Any attribute extended on this page is available for searching on the Identity Search page.

You can assign icons to extended attributes to highlight these accounts in certifications and the detailed identity pages. See "Create Icons to Represent Specialized Account Attributes" on page 36.

Specialized account attributes can be modeled to handle any concept using simple one-to-one mapping and rules. This section describes two of the most common scenarios.

Use the Account Attributes page to view the extended account attribute information for your configuration. Use this page to set up specialized account attributes such as Privileged and Service, and any other extended attributes for use in certifications and searches.

The Account Attributes page contains the following information:

Table 14— System Setup- IdentityIQ - Account Mapping - Account Attributes Descriptions

Column	Description
Attribute	The display name of an account attribute derived from the attribute and its associated application in the Primary Source Mapping column.
Primary Source Mapping	The first of the list of attribute/application pairs or rules from which account attributes are derived. If the required data is unavailable on this primary source, the collection process continues down the list of configured sources until the information is found. Set up the list of sources on the Edit Account Attribute page.

To work with the attributes and sources, see "How to Add or Edit Account Attributes" on page 34.

To delete account attributes, right-click the attribute and select **Delete**.

To edit account attributes, right-click the attribute and select **Edit**.

Edit Account Attributes Page

Use the Edit Account Attribute page to create and edit account attributes including the display name, attribute type and source mapping. You can also use this page to create specialized account attributes. See "Create Icons to Represent Specialized Account Attributes" on page 36.

The maximum number of searchable attributes that can be created is defined during the installation and configuration process. By default you can set five searchable account attributes.See "System Setup" on page 5.

The Edit Account Attribute page contains the following information:

Table 15— Edit Account Attributes Page Field Descriptions

Field	Description
Account Attribute:	
Attribute Name	The name of the attribute as it appears in the application. **Note: Changing an attribute name might cause attributes that were previously aggregated to no longer be recognized.**
Display Name	The IdentityIQ user assigned name for use throughout IdentityIQ.
Advanced Options:	
Edit Mode	Enable editing of this attribute. **Read Only** — this attribute cannot be edited. **Permanent** — changes made to this attribute manually are not overwritten by refresh tasks. **Temporary** — changes made to this attribute manually are overwritten by the first refresh task that detects a value different than the original value. For example, if the original value is A and it is manually changed to B, the value is not overwritten by a refresh task until the newly aggregated value is not A. When an aggregation detects a value that is not A, it refreshes the manually-changed value and the value is updated with each subsequent refresh.
Attribute Type	The attribute type to be linked, for example string, boolean or date.

Table 15— Edit Account Attributes Page Field Descriptions

Field	Description
Searchable	Account attributes are existing link values and are always searchable. This field is displayed as selected and read only so that identity and account attribute configuration pages are consistent in appearance.
Multi-Valued	Specify attributes for which multiple values might be returned during aggregation. Attributes flagged as multi-valued are stored as a list. Even objects that have a single value for a multi-value attribute are stored as a single-item list. Multi-valued attributes are used for queries throughout the product.
Source Mappings: The list of attribute/application pairs or rules from which account attributes are derived. If the required data is unavailable on this primary source, the collection process continues down the list of configured sources until the information is found. This feature is unlikely to be used for Account Attribute mapping.	

How to Add or Edit Account Attributes

1. Click **Add New Attribute** or click an existing attribute to display the Edit Account Attribute page.
2. Enter or change the attribute name and an intuitive display name.

   Note: You cannot define an extended attribute with the same name as any application attribute that is provided by a connector.
3. Edit the Advance Options as required.
4. Click **Add Source** to display the Add a source dialog and specify a source for the new attribute.

   Map directly to an attribute on an application.
   a. Select **Application Attribute**.
   b. Select an application from the **Application** drop-down list.
   c. Select an attribute from the **Attribute** drop-down list.

   Map to an application rule. This rule only applies to the application specified.
   a. Select **Application Rule**.
   b. Select an application from the **Application** drop-down list.
   c. Select a rule from the **Rule** drop-down list.

   Map to a global rule. This rule applies to all applications that contain this attribute.
   a. Select **Global rule (all applications)**.
   b. Select a rule from the **Rule** drop-down list.
5. Click **Add** to add the new source.
6. Use the arrows to the right of the sources list to rearrange the search order for the attribute sources. When aggregation tasks are run they search the source at the top of the list, or the primary source, first and then work down the list.
7. Click **Save** to create the new attribute and return to the Account Attribute page.

Account Attributes

Create a Service Account Using Simple Mapping

In this example, if IdentityIQ finds an attribute named Service that has a value of true on the application DB Application it is marked as a service account. For this case the database connector has already provided an attribute value to reflect the service state, so a simple mapping is all that is required.

Note: After configuring these attributes you must re-aggregate or refresh the identity cubes to set the values.

To configure the mapping:

1. Access the Account Attributes page.
   Select the System Setup tab and select **Account Mappings** from the table.
2. Click **Add New Attribute** to display the Edit Account Attribute page.
3. Specify the following values:
   - **Attribute Name** — service
   - **Display Name** — Service Account
   - **Edit Mode** — Read Only
   - **Attribute Type** — boolean
   - **Searchable** — Read Only
   - **Multi-Valued** — this is not a multi-valued attribute so do not select this field.
4. Click **Add Source Mapping** to display the Add a source to the attribute dialog.
5. Map the attribute:
   a. Select **Application Attribute**.
   b. Select **DB Application** from the Application drop-down list.
   c. Select **Service** from the Attribute drop-down list.
6. Click **Add**.

Create a Privileged Account Using a Rule

In this example, if IdentityIQ finds an account that is a member of the group **Domain Admins** on any AD application, that account should be marked as a privileged account.

1. Write the rule to define the logic.
   This rule checks each account on every AD application and looks for the Domain Admins group. If the Domain Admins group is found, the rule returns true, and the account is considered privileged.
   Example rule:

```
<Rule language="beanshell" name="Example privileged promotion rule"
type="LinkAttribute">
<Source>
<![CDATA[
Boolean privileged = null;
If ( link.getApplication().getName().contains(“AD”) ) {
privileged = new Boolean(false);
List groups = (List)link.getAttribute("memberOf");
if ( groups != null ) {
```

```
for ( String group : groups ) {
if ( ( group != null ) &&
( group.startsWith("cn=Domain Admins") ) ) {
privileged = new Boolean(true);
}
}
}
)
return privileged;
]]>
</Source>
</Rule>
```

2. Access the Account Attributes page.
   Go to the Global Settings and select **Account Mappings**.
3. Click **Add New Attribute** to display the Edit Account Attribute page.
4. Specify the following values:
   - **Attribute Name** — service
   - **Display Name** — Service Account
   - **Edit Mode** — Read Only
   - **Attribute Type** — boolean
   - **Searchable** — Read Only
   - **Multi-Valued** — this is not a multi-valued attribute so do not select this field.
5. Click **Add Source** to display the Add a source to the attribute dialog.
6. Map the attribute:
   Select **Global Rule (all applications)**.
   Select **Example privileged promotion rule** from the Application drop-down list.
7. Click **Save**.

Create Icons to Represent Specialized Account Attributes

Assign icons to extended attributes to highlight these accounts in certifications and the detailed identity pages. To assign icons you must modify the UIConfig file and add AccountIconConfig entries for any value that should be recognized.

The following example references the attributes defined in this section.

```
<ImportAction name='merge'>
<UIConfig name='UIConfig'>
<Attributes>
<Map>
<entry key='accountIconConfig'>
<value>
<List>
<!—This indicates that when we are displaying accounts and we see
 the value “true” for the extended account attribute named
privileged we should display the icon listed in the “source”
attribute. The title will be used in hover-over help.
->
<AccountIconConfig attribute="privileged"
value="true"
```

```
source="/images/icons/privilege_16.png"
title="This is a privileged account"/>
<!—This indicates that when we are displaying accounts and we see
the value “true” for the extended account attribute named
service we should display the icon listed in the “source”
attribute. The title will be used in hover-over help.
->
<AccountIconConfig attribute="service"
value="true"
source="/images/icons/service.png"
title="This is a service account"/>
</List>
</value>
</entry>
</Map>
</Attributes>
</UIConfig>
</ImportAction>
```

Use the IdentityIQ console to import the modifications.

Application Attributes

Use the Edit Application Configuration page to define extended application attributes not provided by the application connectors during aggregation. These extended attributes are displayed on the Attributes tab of the Application Configuration page below the connection attributes provided by the connector. Because the additional attributes are stored with those provided by the connector, if you define an extended attribute with a name that matches any connector attribute, the values of the extended attribute overwrite the values of the connector attribute.

You can use these extended attributes inside rules and custom reports and queries.

The Edit Application Configuration page contains the following information:

Table 16— System Setup- IdentityIQ -Configure Account Mapping Descriptions

Column	Description
Name	The display name of the application attribute assigned when it was added.
Category	The category defined when the attribute was created. If no category was defined this column is blank.
Description	A short description of the extended application attribute.

Click **New Attribute** to add additional attributes to the applications.

To edit or delete an existing attribute from the list, right-click the attribute and select the corresponding option from the menu. If you are deleting an attribute you must confirm the deletion in the pop-up dialog.

Edit Application Attributes

Use the Edit Extended Attribute page to create and edit additional application attributes including the display name, attribute type and description.

The fields displayed on the Edit Extended Attribute page are dependent on the attribute type selected.

The Edit Extended Attribute page contains the following information:

Table 17— Edit Extended Attribute Page Field Descriptions

Field	Description
Attribute Name	The name of the attribute as it appears in the application. Changing an attribute name might cause attributes that were previously aggregated to no longer be recognized.
Display Name	The IdentityIQ user assigned name for use throughout IdentityIQ.
Type	The attribute type to be linked, for example string, boolean, date, rule, or identity.
Description	A brief description of the application attribute.
Category Name	An optional category used to separate the attributes into categories on the Application Configuration page. Enter a category name or select an existing one from the drop-down list.
Searchable	Enable this application attribute for use in searches throughout the product.
Editable	Enable editing of this attribute from other pages in the product.
Required	For String type attributes only. Required attributes must have a value before you can save an application.
Allowed Values	For String type attributes only. Enter the values that are allowed for this attribute. The values entered in this list are used to populate the drop-down value list on the Application Configuration page.
Default Value	Enter a default value for the attribute or select a value from the drop-down list, depending on the attribute type you are working with.

How to Add or Edit Extended Attributes

1. Click **New Attribute** or click an existing attribute to display the Edit Extended Attribute page.
2. Enter or change the attribute name and an intuitive display name.

   Note: You cannot define an extended attribute with the same name as any application attribute that is provided by a connector.

   Note: If you define an extended attribute with the same name as an application attribute, the value of the extended attribute overwrites the value of the connector attribute.
3. Select the attribute type from the drop-down list, String, Integer, Boolean, Date, Rule, or Identity.
4. **Optional:** Enter a description of the additional attribute.
5. **Optional:** Select a category for the attribute.
6. **Optional:** Activate the Searchable option to enable this attribute for searching throughout the product.
7. **Optional:** Activate the Editable option to enable this attribute for editing from other pages within the product.
8. **Optional:** Mark the attribute as required. For string type attributes only.
9. **Optional:** Enter allowed values for the attribute. For string type attributes only.
10. **Optional:** Specify a default value.
11. Click **Save** to save your changes and return to the Edit Application Configuration page.

Entitlement Catalog Attributes

Use the Edit Entitlement Catalog Configuration page to define custom extended entitlement attributes. The extended attributes are displayed with the rest of the entitlement information throughout the product. An example of a extended entitlement attribute might be Time Zone.

The Edit Entitlement Catalog Configuration page contains the following information:

Table 18—System Setup- IdentityIQ - Entitlement Catalog Attributes - Column Descriptions

Column	Description
Name	The display name of the extended entitlement attribute assigned when it was added.
Category Name	The category defined when the attribute was created. If no category was defined this column is blank.
Description	A short description of the extended entitlement attribute.

Click **New Attribute** to add additional extended entitlement attributes.

To edit or delete an existing attribute or type from the list, right-click the item and select the corresponding option from the menu. If you are deleting, you must confirm the deletion in the pop-up dialog.

Edit Extended Entitlement Attributes

Use the Edit Extended Attribute page to create and edit additional role attributes including the display name, attribute type and description.

The Edit Extended Attribute page contains the following information:

Table 19— Edit Extended Entitlement Attribute Page Field Descriptions

Field	Description
Attribute Name	The name of the attribute as it appears in the application. Changing an attribute name might cause attributes that were previously aggregated to no longer be recognized.
Display Name	The name for use throughout the product.
Type	The attribute type to be linked, for example string, boolean, date, rule, or identity.
Description	A brief description of the entitlement attribute.
Category Name	An optional category used to separate the attributes into categories on the Application Configuration page. Enter a category name or select an existing one from the drop-down list.
Searchable	Enable this entitlement attribute for use in queries.
Editable	Enable editing of this attribute from other pages in the product.
Required	For String type attributes only. Required attributes must have a value before you can save an entitlement.
Allowed Values	For String type attributes only. Enter the values that are allowed for this attribute. The values entered in this list are used to populate the drop-down value list on the Roles page.

Table 19— Edit Extended Entitlement Attribute Page Field Descriptions

Field	Description
Default Value	Enter a default value for the attribute or select a value from the drop-down list, depending on the attribute type you are working with.

How to Add or Edit Extended Entitlement Attributes

1. Click **New Attribute** or click an existing attribute to display the Edit Extended Attribute page.
2. Enter or change the attribute name and an intuitive display name.

   Note: You cannot define an extended attribute with the same name as any application attribute that is provided by a connector.
3. Select the attribute type from the drop-down list, String, Integer, Boolean, Date, Rule, or Identity.
4. **Optional:** Enter a description of the additional attribute.
5. **Optional:** Select a category for the attribute.
6. **Optional:** Activate the Searchable option to enable this attribute for searching throughout the product.
7. **Optional:** Activate the Editable option to enable this attribute for editing from other pages within the product.
8. **Optional:** Mark the attribute as required. For string type attributes only.
9. **Optional:** Enter allowed values for the attribute. For string type attributes only.
10. **Optional:** Specify a default value.
11. Click **Save** to save your changes and return to the Edit Entitlement Catalog Configuration page.

Quicklink Populations

Quicklinks are tasked-based links to frequently-used areas of IdentityIQ. Quicklinks are displayed as cards on the IdentityIQ Home page and as links in the Quicklink Menu, which is available throughout the product.

Use the Quicklinks Populations page to associate quicklinks, that are created and imported into IdentityIQ by your administrators, with quicklink populations, sometimes referred to as dynamic scopes.

Quicklink populations grant access to specific areas of IdentityIQ to predetermined populations of users. These populations can be defined based on capabilities, identity attributes, work groups, or by selecting individual identities.

One predefined population, Everyone, is in the list by default. If you have purchased IdentityIQ Lifecycle Manager you also see Help Desk, Manager, and Self Service in the Populations list.

Select a population from the list or click **New** to open the Configuration and Quicklinks tabs.

The Configuration tab contains the following:

Table 20—Quicklink Populations page Configuration tab field descriptions

Field	Description
Details	
Name	Name of the population.
Description	Description of the population.

Table 20—Quicklink Populations page Configuration tab field descriptions

Field	Description
Membership	
Membership Rule	Select a membership rule to define the population. **None** — only the identities specified in the **Included Identities** list are in the population. **All** — include all identities in the population. **Match List** — only identities whose criteria match that specified in the list. Add identity attributes, application attributes and application permissions. Customize further by creating attribute groups to which this assignment rule applies. If Is Null is selected, the associated value text box is disabled. When the is null match is processed, the term matches users on the chosen application who have a null value for that attribute/permission. **Filter** — a custom database query. **Script** — a custom script. **Rule** — select an existing rule from the drop-down list. Click **Edit Rule** to launch the Rule Editor. See “Rule Editor” on page 17. **Population** — select an existing population.
Included Identities	Manually select identities to include in the population.
Excluded Identities	Manually select identities that should not be included in the population. For example, Administrator.
Who can members request for? Identities for whom the members of this population can make access requests.	
Everyone	Can create access request for anyone.
Specific Users	Can only create access requests for identities based on the selected criteria. Use the drop down list to specify if they must match all of the criteria or just any of the criteria.
Share attributes with the requester	Can make requests for identities that share the attributes specified.
Report to the requester	Enable managers to make requests for their subordinates. Specify if this applies to direct reports or all subordinates. If all subordinates, specify a **Maximum Hierarchical Depth**.
Match custom criteria	The filter is the context of the identity object and is parsed as a Velocity template with a parameter called requester. For example for an identity whose manager’s name is the same as the manager’s name for the requester: `manger.name == "$requester.manager.name"`
Ignore scoping	Disregard IdentityIQ scopes when determining for whom request can be made.
What can members request? **Note: Click Edit Rule to launch the Rule Editor for any of the following. See “Rule Editor” on page 17.**	
Roles	Select a rule that defines the set of roles that this population can request.
Applications	Select a rule that defines the set of applications from which this population can request entitlements.

Table 20—Quicklink Populations page Configuration tab field descriptions

Field	Description
Entitlements	Select a rule that defines the set of entitlements that this population can request.

Note: The Quicklinks tab contains all of the quicklinks that are available in your environment. You cannot add quicklinks within IdentityIQ. The New button opens the Configuration tab to create a new population.

Table 21—Quicklink Populations page Quicklinks tab description

Column	Description
Enabled	Specify which quicklinks to associate with this population.
Name	Name of the quicklink as it appears on cards on the IdentityIQ Home page and as links in the Quicklink Menu.
Description	Description of the quicklink.
Category	The category in which this quicklink displays in the Quicklink Menu.
Options	When available, use **Configure** to specify quicklink settings.

Forms

Form Editors are used for configuring forms for Workflows, Role Provisioning Policies, and Application Provisioning Policies in IdentityIQ. Forms are referred in two ways, Centralized Forms and Reference Forms.

Centralized Forms

Centralized Form location is a single location within the system, where an Administrator would be able to view all the forms. All of the forms can be created, edited, managed, and maintained as one object.

The Forms page displays all the standalone forms. The form grid displays Workflow, Role and Application types of forms.

Create and Edit Forms

To create a new form, click the **Create Form** button. On the **Create New Form** window, select one of the following type of the form to be created:

- Application Provisioning Policy Form
- Role Provisioning Policy Form
- Workflow Form

On saving the newly created forms, the form grid would be updated.

To edit an existing form, click the **Edit** button provided next to each form in the grid to display the Form Editor page.

Search Forms

Forms are searched by their names. To search the required form, enter any name word of the Form Name. The search results is displayed by refreshing the list to display the searched form.

Reference Forms

Reference Forms provide a means to create a single form that can be reused and referenced as standalone forms for Application Provisioning Policy Form, Role Provisioning Policy Form, and Workflow Form.

Create Forms

To create a new policy form, click **Add Policy** under the Provisioning Policies tab to display the Forms window. Click **Create Policy Form** to display the Provisioning Policy Editor page.

On saving the newly created policy, the provisioning policies are updated.

Referencing Forms

Note: The Existing Forms lists only contain the reference forms of the type associated with the policy type you are viewing, application, role, or workflow. If you are not seeing forms in the list, ensure that the forms you are looking for have the correct type set on the Global Settings -> Forms page.

Form referencing is done by using the form name. To reference a form, click **Add Policy** under the Account Provisioning Policies tab to display the Forms window. Click **Reference Policy Form** to display the list of application provisioning policy forms from the central location on the Existing Forms page.

Edit Reference Forms

Click the policy name to display an option to **Create a Form** or **Change Reference Policy Form** button to edit an existing reference form.

- Click **Create Policy Form** to display the Provisioning Policy Editor page. On saving the newly created policy, the reference provisioning policies are removed and policy is updated with newly created form.
- Click **Change Reference Policy Form** to display a list of application provisioning forms.

Role Configuration

Use the Edit Role Configuration page to define custom extended role attributes and role types. The extended attributes are displayed with the rest of the role information throughout the product. An example of a extended role attribute might be role status. Role type is used to configure roles to perform different functions within your business model. For example, type might be used to control inheritance or automatic assignment of roles.

The Edit Role Configuration page contains the following information:

Table 22—System Setup- IdentityIQ -Configure Role Attribute Column Descriptions

Column	Description
Role Attributes:	
Name	The display name of the role attribute assigned when it was added.
Category	The category defined when the attribute was created. If no category was defined this column is blank.
Description	A short description of the role attribute.
Role Types:	
Name	The display name of the role type.
Description	A short description of the role type.

Click **New Attribute** to add additional role attributes. See "Edit Extended Role Attributes" on page 44.

Click **New Type** to add or edit a role type. See "Edit Role Types" on page 45.

To edit or delete an existing attribute or type from the list, right-click the item and select the corresponding option from the menu. If you are deleting, you must confirm the deletion in the pop-up dialog.

Edit Extended Role Attributes

Use the Edit Extended Attribute page to create and edit additional role attributes including the display name, attribute type and description.

The Edit Extended Attribute page contains the following information:

Table 23— Edit Extended Role Attribute Page Field Descriptions

Field	Description
Attribute Name	The name of the attribute as it appears in the application. **Note: Changing an attribute name might cause attributes that were previously aggregated to no longer be recognized.**
Display Name	The name for use throughout the product.
Type	The attribute type to be linked, for example string, boolean, date, rule, or identity.
Description	A brief description of the role attribute.
Category Name	An optional category used to separate the attributes into categories on the Application Configuration page. Enter a category name or select an existing one from the drop-down list.
Searchable	Enable this role attribute for use in queries.
Editable	Enable editing of this attribute from other pages in the product.
Required	For String type attributes only. Required attributes must have a value before you can save a role.

Table 23— Edit Extended Role Attribute Page Field Descriptions

Field	Description
Allowed Values	For String type attributes only. Enter the values that are allowed for this attribute. The values entered in this list are used to populate the drop-down value list on the Roles page.
Default Value	Enter a default value for the attribute or select a value from the drop-down list, depending on the attribute type you are working with.

How to Add or Edit Extended Attributes

1. Click **New Attribute** or click an existing attribute to display the Edit Extended Attribute page.
2. Enter or change the attribute name and an intuitive display name.

   Note: You cannot define an extended attribute with the same name as any application attribute that is provided by a connector.
3. Select the attribute type from the drop-down list, String, Integer, Boolean, Date, Rule, or Identity.
4. **Optional:** Enter a description of the additional attribute.
5. **Optional:** Select a category for the attribute.
6. **Optional:** Activate the Searchable option to enable this attribute for searching throughout the product.
7. **Optional:** Activate the Editable option to enable this attribute for editing from other pages within the product.
8. **Optional:** Mark the attribute as required. For string type attributes only.
9. **Optional:** Enter allowed values for the attribute. For string type attributes only.
10. **Optional:** Specify a default value.
11. Click **Save** to save your changes and return to the Edit Role Configuration page.

Edit Role Types

Use the Edit Role Type Definition page to create and edit types to use with roles. Role type is used to configure roles to perform different functions within your business model. For example, type might be used to control inheritance or automatic assignment of roles.

Role modeling also uses the concept of permission to enable you to grant users permission to specific roles without assigning them the role or incorporating it in their role hierarchy. For example, while a non-IT user with a business-type role might need access to the entitlements contained within an IT-type role, they probably do not need to have that role assigned to them or included as part of their hierarchal role structure.

The Edit Role Type Definition page contains the following information:

Table 24— Edit Role Type Definition Page Field Descriptions

Field	Description
Type Name	The name of the role type.
Display Name	The display name of the role type used throughout the product.
Description	A brief description of the role type.

Table 24— Edit Role Type Definition Page Field Descriptions

Field	Description
Icon Path	The path to the iconic representation of this role type. See "Edit Role Types" on page 45
Disallow inheritance of other roles	Do not allow roles of this type to inherit other defined roles.
Disallow other roles from inheriting this role	Do no allow roles of this type to be inherited.
No automatic detection with profiles	Do not automatically detect and assign this role to identities during aggregation and correlation.
No automatic detection with profiles unless assigned	Do not automatically detect and assign a role during aggregation and correlation unless it is required or permitted by an identity's assigned roles.
No entitlement profiles	Do not enable the direct assignment of profiles to this role type. For example, a roles used to create hierarchy in your business model might only gain access to entitlement profiles through permitted IT roles.
No automatic assignment with rule	Do not allow a rule to automatically assign roles of this type to identities.
No assignment rule	Do not display the Assignment Rule panel in the Role Modeler for rules of this type.
No manual assignment	Do not allow roles of this type to be assigned manually from the Identities User Rights tab.
No permitted roles list	Do not display the Permitted Roles panel in the Role Modeler for rules of this type.
Disallow this role from being on a permitted roles list	Do not display roles of this type on the select list of the Permitted Roles panel of any other role.
No required roles list	Do not display the Required Roles panel in the Role Modeler for rules of this type.
Disallow this role from being on a required roles list	Do not display roles of this type on the select list of the Required Roles panel of any other role.
Disallow Granting of IdentityIQ User Rights	Do not allow the granting of IdentityIQ capabilities or scopes based on role assignment. If this option is selected, the Granted IdentityIQ User Rights table is not displayed on the Role Editor page.

How to Add or Edit Role Types

1. Click **New Type** or click an existing type to display the Edit Role Type Definition page.

2. Enter or change the name and display name.
3. Enter an icon path to link to the iconic image associated with roles of this type in the Role Modeler.
   To assign an icon to a role type, do the following:
   a. Add two icon images to iiq_home/images/icons folder of your IdentityIQ installation, one for the role and one for the role as it is undergoing analysis or approval. For example,
   ```
   .itIcon {
   background-image: url("../images/icons/modeler_application_16.png")
    !important;
   background-repeat: no-repeat;

   .itIconPendingbusiness process {
   background-image: url("../images/icons/
   modeler_application_approval_16.png") !important;
   background-repeat: no-repeat;
   ```
   b. Reference the images from the iiq-custom.css file in the iiq_home/css directory.
4. **Optional:** Select configuration options for the role type.
5. Click **Save** to save your changes and return to the Edit Role Configuration page.

Scopes

Scope is used to determine the objects to which a user has access. If scoping is active, identities can only see objects that they created or that are within the scopes they control. IdentityIQ capabilities control the components within the product to which a user has access. Scope controls access to the individual objects within those components. For example, a user might be able to access the Identity Search page, however, the Application and Role drop-down lists only display application and roles that are contained within a scope they control.

Scope is referred to in two ways, Controlled Scope and Assigned Scope. Assigned scope is the scope assigned to an identity or object manually, automatically, or through aggregation and correlation. Controlled scopes refer to the scopes to which an identity has access. You can only see objects that are within your controlled scopes, that you created, or possibly that have no scope assigned. Controlled scope is hierarchical. If you control a parent scope, you control any child scopes contained within.

Use the Configure Scoping page to create new scopes, edit existing scopes, and configure scoping for your enterprise.

Note: If you manually create scopes they should be associated with existing identity attributes or be defined in a scope correlation rule.

Create and Edit Scopes

To create a new scope, right-click **Scopes** and select **New** to display the Create Scope page. Enter the scope name and click **Create** to return to the Scope page. Use the Scope Correlation Rule to correlate identities with the correct scopes.

To edit an existing scope, right-click the scope and select **Edit** to display the Edit Scope page. You can only edit the display name.

Drag and drop existing scopes to create a scope hierarchy.

Delete Scopes

To delete a scope, right-click the scope and select **Delete** to display the Delete Scope page. The Delete Scope page contains the following:

Table 25—Scope Configuration - Delete Scope Page Field Descriptions

Field	Description
Assigned Scope Replacement	Reassign objects to a different scope upon deletion.
Authorized Scope Replacement	Assign an authorized scope to replace the one to be deleted.
Delete Child Scopes	Delete all child scopes in the scope hierarchy.

Configure Scoping

Use the Configure Scoping page to configure scope assignment and correlation. The Configure Scoping page contains the following:

Note: You must run an identity refresh task with the refresh scope option enabled before scope configuration changes are visible.

Table 26—Scope Configuration - Configure Scoping Page Field Descriptions

Field	Description
Enable Scoping	When checked, scoping mechanisms are enabled. Scopes do not take effect until this is enabled, even if the scopes are already defined and assigned. Note: De-selecting this option is useful in troubleshooting performance issues.
Scope Identity Attribute	Select an identity attribute from the drop-down list to use for scoping. A scope is created for each value of the selected attribute aggregated during the identity refresh task. This attribute is used to correlate identities to assigned scope.
Scope Correlation Rule	Select a rule to use to correlated scopes and identities during aggregation and refresh task. If a scope is not found that correlates to the value returned by an attribute, one is created. Scope correlation rules enable more flexibility in scope assignment than specifying a single identity attribute. Note: Click the "..." icon to launch the Rule Editor to make changes to your rules if needed. ... See "Rule Editor" on page 17

Table 26—Scope Configuration - Configure Scoping Page Field Descriptions

Field	Description
Scope Selection Rule	Select a selection rule to use if the identity attribute or scope correlation rule return more then one value for the assigned scope of an identity. For example, if department is specified as the scope identity attribute and the identity aggregation task returns more then on value for department for an identity, this rule determines which value to use as the assigned scope. Note: Click the "..." icon to launch the Rule Editor to make changes to your rules if needed. See "Rule Editor" on page 17
Unsloped Objects Globally Accessible	When selected, all objects that do not have an assigned scope are available to all users. When cleared, all objects that do not have an assigned scope are only available to system administrators.
Identity Controls Assigned Scope	When selected, identities automatically control the scope to which they are assigned.

Time Periods

Use the Configure Time Periods page to specify the time periods used for activity searching. Setting time periods for your enterprise enables you to track who is accessing your sensitive applications and when they are accessing it. Access at unusual times can indicate a security issue that requires investigation. Time periods include things such as office hours, holidays, and weekends. Because each time period is set individually, you can customize the setting to meet the needs of your enterprise.

The following are the available time periods:

- Date ranges — a range of specific dates that define things such as fiscal quarters.
- Time ranges — a range of hours, or times, that define office hours and non-office hours.
- Date lists — a list of dates that define enterprise holidays.
- Day lists — a list of days that define week days and weekends.

To edit a time period, click a time period in the Time Periods column to access the Configure Time Period page and make the required changes. See "Configure Time Period" on page 49.

Configure Time Period

The configuration options on the Configure Time Period page are based on the type of time period to be edited.

Date Ranges:

For date ranges, specify the Begins on and Ends on dates for the time period to be defined. For these date ranges you can enter dates manually of click the... icon and select a date from the calendar.

Time Ranges:

For time ranges, specify the starting at and ending at times. Time ranges are used to define working and non-working hours.

Date Lists:

Use the date list to specify a list of holidays, or regularly schedule dates on which your enterprise business would not normally conduct business. This list does not include weekends. Weekends are defined separately from a day list. You can enter dates manually of click the "..." icon and select a date from the calendar. Use the Add and Delete buttons to add or remove dates from the list.

Day Lists:

Use the day lists to specify week days from weekend days. Select the correct days using the selection boxes on the right of the table and deselect, or DE-activate, the days that should not be included.

Audit Configuration

Use the Audit Configuration page to specify the actions that are collected for audit logs. Since collecting event information and storing it in the audit logs affects performance, a system administrator must specify the actions that are audited. Before any data is collected by the audit logs for use in an audit search, IdentityIQ must be configured for auditing.

The Audit Configuration page contains the following types of actions:

- General Actions — typical action performed while using IdentityIQ. For example, running a task and signing off on a certification are general actions.
- Link Attribute Changes — changes made to any assigned link attributes.
- Identity Attribute Changes — changes to assigned roles, capabilities, authorized scopes, and controlled scopes, and changes to the password. This list might also include extended identity attributes.
- Class Actions — action taken on the underlying classes used to configure the way in which IdentityIQ operates. For example, editing a role, creating a policy, specifying the default email template, and adding attachments are class actions.
- SCIM Resource Actions — action taken on any SCIM resource, for example, create, read, update, and delete.

Electronic Signatures

Electronic signatures provide proof of a decision. Use the Electronic Signatures page to set the meaning or text that is displayed to the end user when they perform an electronic signature.

Note: Electronic Signatures have legal implications. They need to explicitly state each action and consequence to ensure end users understand what they are signing.

Electronic Signature Meanings

The Electronic Signature Meanings page displays a table of configured meanings. Click **New Meaning** to open the New Electronic Signature Meaning window. Input the Name, Display Name, and Meaning then click **Save**.

API Authentication

IdentityIQ supports the use of OAuth 2.0 (client credentials) as a token-based protocol for API authentication. Use this feature to create and manage OAuth clients that you use with the IdentityIQ API.

Note: **You set up a proxy user that connects on behalf of the user to avoid exposing sensitive user data. In order for the proxy user to have correct rights to make API calls, you must assign capabilities to that proxy user.**

OAuth Client Management Page

The OAuth Client Management page has the following tabs and options:

- OAuth Client Management tab — displays a list of the current OAuth clients.
  - Create Button — creates an OAuth client that has a proxy user with an associated secret.
  - Secret Details icon — displays the secret for the an OAuth client.
  - Actions icons — Edit, Delete, Regenerate Secret
- General Settings tab
  - Access Token Expiration In Seconds

How To Create An OAuth Client

1. From the top menu, navigate to the **Gear icon > Global Settings > API Authentication**.
2. On the OAuth Client Management tab, click **Create**.
3. In the OAuth Client dialog enter a unique name for **Client Name** and then enter a user name or select a user from the drop-down list for the **Proxy User**.
4. Click **Save** to save your new OAuth client.

After your create an OAuth client, you can use it with the associated secret to log in and access the token for that proxy user.

IdentityAI Configuration

Note: **This link is only present if you have purchased the IdentityAI product. Refer to the SailPoint *IdentityAI Implementation Guide*.**

Use the IdentityAI Configuration page to connect IdentityIQ to the IdentityAI product.

Table 27— IdentityAI Congiguration Page Field Descriptions

Field	Description
Connection Information for IdentityAI:	
IdentityAI Hostname	The host name of the IdentityAI recommendation API
Client ID	OAuth client ID for the IdentityAI recommendation API
Client Secret	OAuth client secret for the IdentityAI recommendation API
Advanced:	
Read Timeout	The number of seconds IdentityIQ attempts to read recommendations from IdentityIQA before reporting a failure
Connect Timeout	The number of seconds IdentityIQ attempts to connect to IdentityIQA before reporting a failure

Use **Test Connection** to ensure the connection information is accurate and operating.

Import From File

Use the Import From File page to import files into IdentityIQ. For example, use this page to import custom rules or scoring pages.

Note: Imported files might not be immediately available. Some file contents might require an application restart.

Note: Any include directives in the import file include files from the application server file system and not that of the client browser.

Use the Browse button to navigate to a file or enter the path manually and click **Import** to begin the download process.

Select **No role events generated for role propagation** to avoid event generation during role propagation.

When the import is complete the results are displayed on the Import from File Results page. Click **Import Another File** to go back to the Import from File page, or **Done** to return to the System Setup page.

Compliance Manager

From the Navigation bar, click the gear icon and then select **Compliance Manager**. Use the Compliance Manager page to configure control and default settings for certifications. The following table displays the available options.

Note: Most of the fields on this page enable you to configure default settings that a user can change on certifications they are reviewing or scheduling. Those fields that behave differently are described as such.

Table 28—Compliance Manager - Certification Configuration Descriptions

Field	Description
Presentation:	
Initial Access Review View	Select the view displayed when access review reports are initially accessed. **List** — open the grid view, either the worksheet or list view. **Detailed** — open the Access Review Decisions tab associated with the first item in the access review.
Default Access Review Grid View	Select the grid view to display for all identity-type access review report list pages. **Worksheet** — the individual line items that are assigned to the identities within identity-type access reviews. **Identity** — the top-level items that make up an access review, such as identities, groups, or roles.
Default Entitlement Display Mode	Select the default display mode for entitlements. Entitlement Value - display only the assigned value of the entitlement. Entitlement Description - display the longer description of the entitlement.
Subordinate Access Review Page Size	Specify the number of subordinate access reviews to show per page on the certification page before paging. If set to zero, paging is disabled.
Lifecycle:	

Table 28—Compliance Manager - Certification Configuration Descriptions

Field	Description
Notify Users of Revocations	Select to enable email notifications to users that have items revoked.
Certification Escalation Rule	Select a rule from the drop-down list as the default rule that the system uses when an access review is escalated.
When Exceptions Expire	Select the action performed on a mitigation when it expires
Active Period Duration	Input the number of units and unit type (hours, days, weeks or months) to use as the default active period duration.
Enable Challenge Period	Select to enable default challenge period and its default duration. The challenge period enables users to challenge requests from certifiers to remove access privileges.
Enable Revocation Period	**Note: If the revocation period is disabled, the certification is not scanned for completed revocations and revocation status might not be accurately reflected throughout the product.** Select to enable the default revocation period and its default duration. The revocation period places a limit on the amount of time a revoker has to act on a revocation request before that request work item is escalated.
Default Revoker	Select the user to whom all bulk remediation requests are to be sent. Bulk revocation requests are made during the certification process. You can select an item from the **Select Bulk Action** drop-down list on the Certification Report worksheet view or click **Revoke All** on the Certifications Decision tab. If this field is left blank, the remediator is specified as part of the request process.
Enable Automatic Closing	Specifies that the remediation period should be enabled, during which IdentityIQ periodically scans users to determine whether the requested remediations have been carried out. Use the following options to configure the details of this process. **Time After Certification Expiration** — Select the amount of time following this access review expiration date that IdentityIQ should wait before attempting to automatically close it. **Closing Rule** — Select the rule that IdentityIQ runs at the beginning of the automatic closing process. **Action Taken On Undecided Items** — The action that IdentityIQ assigns to any undecided items when automatically closing this access review. Choose from Approve, Revoke, or Allow Exception. **Comments** — Input the comments that IdentityIQ adds to any undecided items when automatically closing this access review. **Signer** — Select the identity who signs off on automatically closed access reviews. This setting is only configurable at the system setup level. Individuals who are scheduling certifications cannot define the signer.
Behavior:	

Table 28—Compliance Manager - Certification Configuration Descriptions

Field	Description
Require Bulk Certification Confirmation	Select to prompt user for confirmation on bulk access reviews.
Selection Count Requiring Bulk Revoke Confirmation	Input the number of selected items which require additional confirmation for bulk revocations.
Prompt for Sign Off	Select to display a pop-up window when an access review is complete and ready for sign off.
Require Electronic Signature	Select to require that, by default, all certifications require an electronic signature.
Require Subordinate Completion	Require that, by default, all subordinate access reviews be completed before the parent access review can be completed.
Auto Sign Off When Nothing to Certify	Automatically sign off the certification when assignee has nothing to certify.
Suppress Notification When Nothing to Certify	Suppress notification of certification when assignee has nothing to certify.
Require Reassignment Completion	Require that, by default, all reassigned access review items be completed before the parent access review can be completed.
Return Reassignments to Original Access Review	Specify that, by default, the content of reassigned access reviews be returned to the parent access review upon sign off. Use this option to ensure that the original content of an access review request is preserved for tracking and reporting purposes.
Automatically Sign Off When All Items Are Reassigned	Specify that an access review be automatically signed off on when all items in that access review are reassigned. **Note: This item is not available if the Required Reassignment Completion or the Return Reassignments to Original Access Review options are selected.**
Require Comments for Approval	Require that all certifiers enter comments for each item they approve in an access review request.
Require Comments When Allowing Exceptions	Require certifier to include comments when a certification decision is made.
Require a review on delegated certification items	Select to require that all access review approvers review the decision made on any user, role, entitlement, or policy violation that they delegated to another approver before they can complete the access review containing that delegation.
Require delegated certification items to be completed	Select to require that all items in a delegation work item have a decision associated with them before the work item can be marked as complete. This setting is only configurable at the system setup level. Individuals cannot change the value of this setting for a single certification.

Table 28—Compliance Manager - Certification Configuration Descriptions

Field	Description
Disable Delegation Forwarding	Select to disallow the forwarding of a work item that a different user delegated.
Allow Self Certification For	Choose which users may self-certify - that is, be the certifier for their own access, either by forwarding or reassigning an access review: All certifiers, Certification and System Administrators, System Administrators only
Self Certification Violation Owner	For users that are not allowed to self-certify, this is the identity or workgroup that will receive any items that would require a self-certification - that is, when the reviewer and the user whose access is under review are the same person. If a Self Certification Violation Owner is not specified, any items that require self-certification will be read-only to the reviewer.
Limit Reassignments	The limit reassignment feature allows you to limit the number of times the users within the certification campaign can reassign a certificate item.
Reassignment Limit	Set the number of reassignments allowed. **Note: Certification is not forwarded or reassigned when the reassignment limit is reached.**
Decisions:	
Enable Provisioning Missing Role Requirements	Enable the certifier to provision missing role requirements from within an access review.
Enable Line Item Delegation	Enables certifiers to delegate individual access review items, such as a single role or entitlement, rather than the entire identity to be reviewed. This option also enables the delegation of policy violations, either from inside an access certification or from the Manage -> Policy Violations page.
Enable Account Approval	Enable certifiers to bulk approve all accounts for a given entitlement.
Enable Account Revocation	Allow users to bulk revoke all entitlements for a given account.
Enable Account Reassignment	Enables a certifier to reassign an account and all of its associated entitlements.
Enable Overriding Violation Remediator	Enable certifiers to specify a remediator from the policy violation pop-up dialog even if there is a default remediator specified.
Enable Role Creation Requests from Certifications	Activate this field to enable certifiers to request that new roles be created from the certification pages.This setting is only configurable at the system setup level. Individuals cannot change the value of this setting for a single certification. Roles requested from the certification pages must be approved before they are available for use by the system. The assignment of role approval requests is controlled by a rule specified on the Configure System Settings Rules tab.

Table 28—Compliance Manager - Certification Configuration Descriptions

Field	Description
Enable Identity Delegation	Enable certifiers to delegate entire identities from a certification request.
Enable Allow Exceptions (applies only to non-policy violation items)	Enables certifiers to allow exceptions on access review items such as roles or entitlements, that are not policy violations. Allowing an exception means the user should not have access indefinitely, but can retain access for a specified period of time.
Deprovision Items When Exception Expires (applies only to non-policy violation items)	Enables automatic deprovisioning of access when the allowed exception period has expired. This setting applies only to items such as roles or entitlements, that are not policy violations.
Enable Allow Exception Popup	Enables certifiers to view the Allow Exception pop-up and manually set expiration dates.
Default Duration for Exceptions	Set the time period during which exceptions should be allowed. Input the number of units and unit type (hours, days, weeks or months) to use as the exception duration.
Default Operation for Remediation Modifiable Attributes	Set the default operation shown on the revocation dialog for remediation-modifiable attributes.
Show Recommendations	**Note: This option is only visible if you have purchased and activated the SailPoint IdentityAI product.** Enable recommendations from IdentityAI to display in access reviews.
Bulk Actions:	
Select the actions to enable from the Worksheet/Identity view and the Detail view. The actions include the following: Enable Bulk Approve Enable Bulk Revocation Enable Bulk Allow Exceptions Enable Bulk Reassignment Enable Bulk Account Revocation Enable Bulk Clear Decisions	
Certification Contents:	
Additional Entitlement Granularity	The default granularity at which additional entitlements are listed in the access review. For example, if you select **Attribute/Permission**, each permission associated with each attribute is listed, and must be acted upon, separately.
Exclude Logical Tier Entitlements	Exclude entitlements on tier application accounts from the access review. This only applies to logical applications. Tier applications are those application that make up a logical application.

Table 28—Compliance Manager - Certification Configuration Descriptions

Field	Description
Generate Certification(s)	Specify whether, by default, access review requests should generate an access review request for the specified managers, or for the specified managers and all employees below them in the reporting hierarchy. If you select **For the specified manager(s) only**, the **Flatten Hierarchy** option is displayed. Select the **Flatten Hierarchy** option to include all of the employees that report directory to the selected managers and the employees that report to their subordinate managers on the access review request.
Email Templates:	
Much of the communication performed during the access review process is done through email notifications sent automatically by IdentityIQ as an access review proceeds through its life cycle. Use this section to specify the template to use for each certification-related notice.	

Chapter 2: Lifecycle Manager Setup

The Lifecycle Manager portion of IdentityIQ Setup includes of the following:

Lifecycle Manager Configuration

Use Lifecycle Configuration to customize the availability of tools and functionality based on end user needs. Lifecycle Manager configuration is divided into the following sections:

Note: IdentityIQ System Administrators can make any request regardless of the Lifecycle Manager Configuration settings.

- See “Configure Tab” on page 59
- See “Business Processes Tab” on page 63
- See “Identity Provisioning Policies Tab” on page 63

Configure Tab

Use the Configure tab to customize your Lifecycle Manager configuration. The Configure tab includes the following.

Table 29—Lifecycle Manager Configuration — Configure Tab Options Descriptions

Field	Description
General Options	
Allow requesters to set request priorities	Use this option to enable requesters to set the priority level of their request. If this option is not selected, all requests have a default “Normal” priority level.
Enable Account Group Management	Use this option to enable provisioning of account groups through Lifecycle Manager requests.
Enable Full Text Search	Use this option to enable full text searching on the Lifecycle Manager request pages. Enabling full text searching might have some affect on the performance of those pages. For detailed information, see "Configuring Full Text Searching" on page 65. You must run the Full Text Index Refresh task before full-text search is available. See “Predefined Tasks” on page 284.
Base directory path used to store full text index files	The directory on the server in which full text index searches are stored.

Table 29—Lifecycle Manager Configuration — Configure Tab Options Descriptions

Field	Description
Enable automatic index refresh	Enables the automatic refreshing of the full text index at the interval specified.
Allow Searching by Population when requesting access	Enable the use of populations as a search filter.
Allow Searching by Identity when requesting access	Enable the use of identities as a search filter.
Allow opt-in to viewing request access search result details	Use this option to limit the amount of information displayed for each item on the Access Request, Review and Submit panel and add a **View Details** button on each item to show the complete information. This feature enables more items to display on each table.
Show external service request details	Use this option to display the information such as request numbers and ID from external ticketing systems throughout IdentityIQ.
Maximum number of results returned in a Request Access search	Limit the number of items returned by an access request. Large lists are hard to scan and the search should be narrowed or refined.
Applications that support additional account requests	Use the drop-down list to specify the applications on which multiple accounts can exist or be created. Select **All Applications** to include all applications in your environment.
Request Role Options	
Request Role Options	Select the role types that are available for role requests. Any options not selected are unavailable to any user attempting to make that type of request.
When searching for roles based on population, only return roles contained by at least the following percentage of the population	Specify the minimum percentage of a population whose roles must match any given search criteria.
Request Entitlement Options	

Table 29—Lifecycle Manager Configuration — Configure Tab Options Descriptions

Field	Description
When searching for entitlements based on population, only return entitlements contained by at least the following percentage of the population	Specify the minimum percentage of a population whose entitlements must match any given search criteria.
Entitlement Search Results must return less than this number of identities when searching by identity	Indicate the maximum amount of identities an entitlement search result can yield.
Create Identity Options	
Require password on all identity creation requests.	Require a password on all identity creation requests.
Enable self-service registration	Enables new user self-registration and creates a link for registration on the IdentityIQ login page. Use the **Advanced View** option to view or configure all available variables. The securityOfficerName variable must be configured within the LCM Registration process variable before the self-service registration functionality is fully enabled. This is done using the "Compliance Manager" on page 52. The default securityOfficername is the IdentityIQ system administrator. Follow these steps to setup self-service registration: 1. From the navigation menu bar, go to **Setup** -> **Business Processes**. 2. In the Edit An Existing Process panel, select **LCM Registration**. 3. Click the **Process Variables** tab. 4. **Security Officer** is the default setting for the **Approvers** field. - To delete the **Security Officer** setting, click the **x** icon next to it. - To add another setting, click the down-arrow next to the **Approvers** field and select another entry. 5. The default entry for the **Fallback Approver** is the IdentityIQ system administrator. If desired, you can change the **Fallback Approver**. 6. When you are satisfied with all of the entries, click **Save** at the bottom of the screen.

Table 29—Lifecycle Manager Configuration — Configure Tab Options Descriptions

Field	Description
URL of action button after successful registration	Enter a URL to redirect the browser to the specified page after successful user registration. If this field is blank, the user is redirected to the login page.
Prevent pruning of new identities for this many days	Select the number of days that must pass after the creation of an identity before it can be pruned. Default is 30 days.
Manage Account Options	
Show Enable/Unlock decision buttons regardless of whether the account is disabled or unlocked.	Display the decision buttons on account management page for disabled or unlocked accounts.
Manage Account Actions	Choose which actions are enabled for Manage Accounts requests for yourself and subordinates. Options include the following: Delete Disable Enable Unlock Deselected options are unavailable to a user attempting to make that type of request. Select one or more applications from the Applications that support account only requests to specify which applications allow Account Only requests. Select All Applications to enable this feature for all applications.
Disable auto refresh account status	The status is automatically refreshed only for the accounts from applications that are not listed in the Disable auto refresh account status list AND accounts that support the Enable or Unlock feature AND accounts without the NO_RANDOM_ACESS feature. Deactivate auto refresh for account status. By default, accounts from all applications support this feature.
Applications that do not support auto refresh account status	Select one or more applications to deactivate auto refresh.
Applications that support account only requests	Select applications from the drop-down list that support request for accounts that are not associated with a role or entitlement. Select All Applications if un-associated accounts can be request for all applications.
Manage Password Options	

Table 29—Lifecycle Manager Configuration — Configure Tab Options Descriptions

Field	Description
Choose Enable password auto-generation when requesting for others to enable passwords to be auto-generated when requests are made on behalf of another user by an authorized user.	
Password Validation Rule	
Select a rule from the drop-down list to used when validating password creations.	
IdentityAI Approval Recommendations	
Enable generation of IdentityAI recommendations for approvals.	
Batch Request Approver	
Require an approval before granting batch requests.	

Business Processes Tab

Use the Business Process tab on Lifecycle Manager Configuration to determine which business process is used when performing specified Lifecycle Manager actions.

Identity Provisioning Policies Tab

Identity Provisioning Policies are used to define identity attributes that must be set when creating an identity from a Lifecycle Manager request.

The following types of Identity Provisioning Policies are available:

- Create Identity
- Update identity
- Self-service Registration

Note: If an Update provisioning policy is defined, that policy overwrites the Create policy.

You must include the criteria required by the provisioning policy in the generated form before the request can be completed. Use the Provisioning Policy Editor to customize the look and function of the form fields generated from the provisioning policy.

Table 30—Provisioning Policy Editor: Identity Provisioning Policy Field Descriptions

Field Name	Description
Name	The name of your provisioning policy.
Description	A brief description of the provisioning policy.
Provisioning Policy Editor Use the Edit Provisioning Policy Fields panel to customize the look and function of the form fields generated from the provisioning policy.	
Attribute	Select the attribute field from the drop-down list to display on the form generated from the provisioning policy.
Display Name	The name displayed for the field in the form generated by the provisioning policy.

Table 30—Provisioning Policy Editor: Identity Provisioning Policy Field Descriptions

Field Name	Description
Help Text	The text you wish to appear when hovering the mouse over the help icon.
Type	Select the type of field from the drop-down list. Choose from the following: Boolean — true or false values field Date — calendar date field Integer — only numerical values field Long — similar to integer but is used for large numerical values Identity — specific identity in IdentityIQ field Secret — hidden text field String — text field
Multi Valued	Choose this to have more than one selectable value in this field of the generated form. Click the plus sign to add another value.
Read Only	Determine how the read only value is derived: Value — value based on the selection from the drop-down list Rule — value is based on a specified rule Script — value is determined by the execution of a script
Hidden	Determine how the hidden value is derived: Value — value based on the selection from the drop-down list Rule — value is based on a specified rule Script — value is determined by the execution of a script
Owner	The owner of the provisioning policy. This is determined by selecting from the following: None — no owner is assigned to this provisioning policy. Application Owner — identity assigned as owner of the application in which the provisioning policy resides. Role Owner — identity assigned as owner of the role in which the provisioning policy resides. Rule — use a rule to determine the owner of this provisioning policy. Script — use a script to determine the owner of this provisioning policy
Required	Choose whether or not to have the completion of this field a requirement for submitting the form.
Refresh Form on Change	Select this option to have the form associated with this policy refresh to reflex changes to this policy.
Display Only	Set this field as display only.
Authoritative	Boolean that specifies whether the field value should completely replace the current value rather than be merged with it; applicable only for multi-valued attributes
Value	Determine how the value is derived. Select from the following: Literal — value is based on the information you provide Rule — value is based on a specified rule Script — value is determined by the execution of a script

Table 30—Provisioning Policy Editor: Identity Provisioning Policy Field Descriptions

Field Name	Description
Allowed Values	The value(s) which can be displayed in the field of the generated form. Choose from the following: None — the field is blank Literal — value is based on the information you provide Rule — value is based on a specified rule Script — value is determined by the execution of a script
Validation	Gives the ability to specify a script or rule for validating the user's value. For example, a script that validates that a password is 8 characters or longer.

Configuring Full Text Searching

When full text searching is enable, users can use the following types of searches to find the correct access to request:

- Keyword search — Users can search based on keywords that relate to role, entitlements and descriptions.
- Affinity search — Users can search for access based on what other users who are similar to them currently have.

Feature / Enhancement	Description	Benefit
Keyword search	Search that finds results based on role and entitlement names, descriptions and extended attributes using relevance-based search and predictive analytics.	Provides a familiar shopping experience for end users. The keyword search makes it possible for end users and managers to find the right access to request.
Affinity Search	Guides users to the right access to request by enabling them to find roles or entitlements assigned to specific users or a population of users.	Enables users to locate roles and entitlements by reviewing access that others in the organization have. The affinity search provides a controlled, governance-based approach that enables you to compare similar access and view any areas of risk, such as high identity risk scores or open policy violations.

Enabling Full Text Searching

To enable the most basic full text searching:

1. From the navigation menu bar, go to **gear icon->Lifecycle Manager Configuration** page. Select the **Enable Full Text Search** option on the **Addition Options** Tab.
2. Select the **Enable Full Text Search**.
3. Run the Full Text Index Refresh task. See “Predefined Tasks” on page 284.

Note: The Full Text Index Refresh must run every time you make a change to roles, managed attributes, or the FullTextIndex objects in your enterprise. The index files are only updated when this task is run. If you do not select this option, you will have to schedule the Full Text Index Refresh to run periodically or you will have to remember to run it manually.

When you run the **Full Text Index Refresh** task the first time, files for each FullTextIndex object in your IdentityIQ configuration are created.

Setting the Location of Index Files

To set the location of the index files, edit the FullTextIndex objects and add an indexPath key.

For example, `<entry key=”indexPath” value=”indexFileLocation”>` where **IndexFileLocation** is a fully qualified path name. By default the index files for roles, BundleIndex, managed attributes, ManagedAttributesIndex, and unstructured targets TargetAssociation are added to the `WEB-INF` folder of the directory where you installed IdentityIQ.

By default, after completing both steps above, you can do full text searches on the following fields:

- **Managed Attributes:** displayableName, description, and application.name
- **Roles:** name, displayableName, and description
- **Targets:** name and description

Adding Additional Fields

To add additional fields, edit the FullTextIndex objects and add a field with analyzed=”true” set: <FullTextField analyzed="true" name="*myAttribute*"/>.

The following example illustrates how to add a new full text searchable field (division) and indicate a location for the index files (/tmp/indexlocation). This example is for the roles index file.

Note: Roles are also referred to as bundles in the product code.

Field options:

- **Analyzed** – used to index the field and for full text searching. Add analyze fields to include custom attributes in full text search.
- **Indexed** – enables the field to be used in the advanced filters on the access request pages.
- **Stored** – enables the field to return in the search results and display on the access request pages, if the user interface is designed to support this use.
- **Ignored** – sets the field to not be used in full text searching nor filtering. This field does appear in the filter passed down from the user interface.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE FullTextIndex PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<FullTextIndex created="1346076712810" id="4028818239686c4f0139686c9f6900e7"
name="Bundle">
 <Attributes>
    <Map>
      <entry key="fields">
        <value>
          <List>
            <FullTextField analyzed="true" indexed="true" name="name"/>
            <FullTextField analyzed="true" indexed="true" name="displayableName"/>
            <FullTextField analyzed="true" name="description"/>
            <FullTextField indexed="true" name="assignedScope.path"/>
            <FullTextField indexed="true" name="type"/>
```

```
            <FullTextField name="defaultDescription" stored="true"/>
            <FullTextField ignored="true" name="disabled"/>
            <FullTextField name="riskScoreWeight" stored="true"/>
            <FullTextField name="owner.id"/>
            <FullTextField name="owner.name"/>
            <FullTextField name="owner.displayName" stored="true"/>
            <FullTextField name="division" analyzed="true" indexed="true">
          </List>
        </value>
      </entry>
      <entry key="indexPath" value="/tmp/indexlocation"/>
    </Map>
 </Attributes>
</FullTextIndex>
```

Special Considerations

When FullTextSearch is enabled, Bundle / Role references within filter objects in Request Object Authority rules should include only the following indexed attributes:

- name
- displayableName
- id
- description
- owner.name
- owner.id
- assignedScopePath (id of the associated scope).

Note: The only attributes that are indexed in the FullTextSearch index are listed above. If you use attributes that are not in this list, extra Bundles are returned during search, which can result in errors in the log.

Creating Direct Links to IdentityIQ

Lifecycle Manager enables you to create direct links into IdentityIQ pages from outside of the product from places such as emails, forms, or portal. These direct links can either use your single-sign on solution or require users to login to IdentityIQ as an intermediate step. Direct links can also use a number of filtering options enabling users to go directly to specific pages using specific filtering criteria.

IdentityIQ supports the following types of direct links:

- "Desktop Direct Links" on page 67
- "Mobile Interface Direct Links" on page 68

Desktop Direct Links

Direct links provide a method to link directly to IdentityIQ Desktop pages. For Example, use the following direct links to go to the Manage Accounts, or Manage Passwords, or Manage Identity pages for a user that is logged in

to IdentityIQ, where *<hostName>* is the name of the host on which IdentityIQ is installed. The following direct links can be used:

- Manage Accounts
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/identities/identities.jsf&rp2=quickLinks/Manage+Account`
- Manage Specific Account
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/identities/identities.jsf&rp2=identities/<identityId>/accounts`
- Manage Password
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/identities/identities.jsf&rp2=quickLinks/Manage%20Passwords/identities`
- Manage Specific Password
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/identities/identities.jsf&rp2=identities/<identityId>/password`
- Create Identity
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/identities/identities.jsf&rp2=quickLinks/Create+Identity/createIdentity`
- Edit Identity
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/identities/identities.jsf&rp2=quickLinks/Edit+Identity`
- Edit Specific Identity
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/identities/identities.jsf&rp2=identities/<identityId>/edit`
- View Identity
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/identities/identities.jsf&rp2=quickLinks/View%20Identity/identities`
- View Specific Identity
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/identities/identities.jsf&rp2=identities/<identityId>/attributes`
- Access Request Details (previously named Track My Requests)
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/identityRequest/identityRequest.jsf&rp2=requests`
- Track My Requests
  `https://<hostname>/identityiq/identityRequest/identityRequest.jsf`
- Manage Certifications
  `https://<hostname>/identityiq/certification/certifications.jsf#/certifications`
- Policy Violation List Page
  `https://<hostname>/identityiq/policyViolation/policyViolation.jsf#/policyViolations`

Mobile Interface Direct Links

Direct links provide a method to link directly to IdentityIQ Mobile pages. The following direct links can be used:

- “Direct Link to Passwords (Mobile)” on page 69
- “Direct Link to Manage Accounts (Mobile)” on page 69.
- “Direct Link to Manage Certifications (Mobile)” on page 69.
- “Direct Link to Policy Violations (Mobile)” on page 69.

- "Direct Link to Access Management Page (Mobile)" on page 69.
- "Direct Link to IdentityIQ Manage Access Review Page (Mobile)" on page 71.
- "Direct Link to Pending Work Items (Mobile)" on page 73.

Direct Link to Passwords (Mobile)

- Manage Password
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/ui/index.jsf&rp2=quickLinks/Manage%20Passwords/identities`
- Manage Specific Password
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/ui/index.jsf&rp2=identities/<identityId>/passwords`

Direct Link to Manage Accounts (Mobile)

- Manage Accounts
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/ui/index.jsf&rp2=quickLinks/Manage%20Accounts/identities`
- Manage Specific Account
  `https://<hostname>/identityiq/ui/rest/redirect?rp1=/ui/index.jsf&rp2=identities/<identityId>/accounts`

Direct Link to Manage Certifications (Mobile)

- Manage Certifications
  `https://<hostname>/identityiq/ui/index.jsf#/certifications`

Direct Link to Policy Violations (Mobile)

- Policy Violations List Page
  `https://<hostname>/identityiq/ui/index.jsf#/listViolations`

Direct Link to Access Management Page (Mobile)

Specific access request pages can be accessed through direct links using parameters. Query parameters can be appended to the Access Review Management tab URL:

Note: Your browser may require Special characters in the parameter values to be URL encoded. For example, spaces must be replaced with %20, & must be replaced with %26, and ? must be replaced with %3F.

`https://<hostname>/identityiq/ui/rest/redirect?rp1=/accessRequest/accessRequest.jsf&rp2=accessRequest/manageAccess/add?identityName=<identity1>&filterRoleType=<roleType1>&filterRoleStringAttr=<roleAttrib1>`

The following parameters allow you to create direct links to the page with a variety of filters already selected:

Table 31—Access Request Management Deep Link Parameters

Type	Parameters
Identity	**identityName** (required) - name of identity the deep link is targeting.

Table 31—Access Request Management Deep Link Parameters

Type	Parameters
Role Filters	**filterRoleType** **filterRole<attribute>** **Note: Only role type and extended attributes are supported. Attributes from the bundle object are not supported.**
Entitlement Filters	**filterEntitlementApplication (multi)** **filterEntitlementAttribute (multi)** **filterEntitlementEntitlement (multi)** **filterEntitlementOwner** **filterEntitlement<attribute>** The (multi) params can be specified multiple times in a single URL. However, filterEntitlementOwner is NOT multi. If an entitlement application has only one attribute defined, the direct link can omit the entitlement attribute on the URL and the defined attribute is used by default. **Note: With the exception of Application, Attribute, and Value, only extended attributes are supported.**
Keyword Filters	**filterKeyword** **Note: If full text search indexing is enable, description is also searched for the keyword.**

Access Request for Single User Pre-Selected

In the following example,
<hostName> is the name of the host on which IdentityIQ is installed
<identity1> is the name of the identity

```
https://<hostname>/identityiq/ui/rest/redirect?rp1=/accessRequest/accessRequest.jsf
&rp2=accessRequest/manageAccess/add?identityName=<identity1>
```

Access Request for Single User Pre-Selected — Filtering on Role Type

In the following example,
<hostName> is the name of the host on which IdentityIQ is installed
<identity1> is the name of the user
`<roleType1>` is the requested role

```
https://<hostname>/identityiq/ui/rest/redirect?rp1=/accessRequest/accessRequest.jsf
&rp2=accessRequest/manageAccess/add?identityName=<identity1>&filterRoleType=<roleTy
pe1>
```

Access Request for Single User Pre-Selected — Filtering on Role Type and Role Extended Attribute

In the following example,
<hostName> is the name of the host on which IdentityIQ is installed
<identity1> is the name of the user
<roleType1> is the type of role
<roleAttrib1> is the role attribute

```
https://<hostname>/identityiq/ui/rest/redirect?rp1=/accessRequest/accessRequest.jsf
&rp2=accessRequest/manageAccess/add?identityName=<identity1>&filterRoleType=<roleTy
pe1>&filterRoleStringAttr=<roleAttrib1>
```

Access Request for Single User Pre-Selected — Filtering on a Single Entitlement Application/Attribute/Value

In the following example,
<hostName> is the name of the host on which IdentityIQ is installed
<identity1> is the name of the user
<entApp1> is the entitlement application
<entAttrib1> is the entitlement attribute (such as memberOf or groupmbr)
<entValue1> is the entitlement value

```
https://<hostname>/identityiq/ui/rest/redirect?rp1=/accessRequest/accessRequest.jsf
&rp2=accessRequest/manageAccess/add?identityName=<identity1>&filterEntitlementAppli
cation=<entApp1>&filterEntitlementAttribute=<entAttrib1>&filterEntitlementEntitleme
nt=<entValue1>
```

Access Request Logged In User Selected with Filtering on Multiple Applications

In the following example,
<hostName> is the name of the host on which IdentityIQ is installed
<identity1> is the name of the user
<entApp1> and <entApp2> are the entitlement applications
<entAttrib1> and <entAttrib2> are the entitlement attributes (such as memberOf or groupmbr)
<entValue1> and <entValue2> are the entitlement values

Note: In the following example, two entitlements are requested.

```
https://<hostname>/identityiq/ui/rest/redirect?rp1=/accessRequest/accessRequest.jsf
&rp2=accessRequest/manageAccess/add?FidentityName=<identity1>
&filterEntitlementApplication=<entApp1>&filterEntitlementAttribute=<entAttrib1>
&filterEntitlementEntitlement=<entValue1>&filterEntitlementApplication=<entApp2>
&filterEntitlementAttribute=<entAttrib2>&filterEntitlementEntitlement=<entValue2>
```

Access Request Logged In User Selected with Filtering on a Keyword Search

In the following example,
<hostName> is the name of the host on which IdentityIQ is installed
<keyword1> is the specific keyword you want to find

```
https://<hostname>/identityiq/ui/rest/redirect?rp1=/accessRequest/accessRequest.jsf
&rp2=accessRequest/manageAccess/add?filterKeyword=<keyword1>
```

Direct Link to IdentityIQ Manage Access Review Page (Mobile)

Specific access request review pages can be accessed through direct links using parameters. Query parameters can be appended to the Access Request Review tab URL:

Note: Your browser may require Special characters in the parameter values to be URl encoded. For example, spaces must be replaced with %20, & must be replaced with %26, and ? must be replaced with %3F.

```
https://<hostname>:<port>/ui/rest/redirect?rp1=/ui/index.jsf&rp2=certification/<id>
```

The following parameters allow you to create direct links to the page with a variety of filters already selected:

Table 32—Access Request Review Deep Link Parameters

Type	Parameters
Identity	**filterKeyword** — search term If no identityName parameter is specified, the loggedInUser is used.
Role	To specify a role or entitlement using name or id: **role (multi) — name of id of role** **entitlement (multi) — entitlement id** The (multi) params can be specified multiple times in a single URL.
Entitlements	To specify an entitlement without an id, use a combo: entitlementApplication<X> entitlementAttribute<X> entitlementValue<X> <X> corresponds to a matching integer, such as entitlementApplication1, entitilementAttribute1, entitlementValue1.

Access Request for Logged In User for a Single Role

In the following example,
<hostName> is the name of the host on which IdentityIQ is installed
<role1> is the name of the role

```
https://<hostName>/identityiq/ui/rest/redirect?rp1=/ui/index.jsf&rp2=accessReque
st/review?role=<role1>
```

Access Request for a Specified User for Multiple Roles

In the following example,
<hostName> is the name of the host on which IdentityIQ is installed
<identity1> is the name of the user
`<role1> and <role2> are requested roles`

```
https://<hostName>/identityiq/ui/rest/redirect?rp1=/ui/index.jsf&rp2=accessRequest/
review?identityName=<identity1>&role=<role1>&role=<role2>
```

Access Request for Logged In User for Single Entitlement Using Entitlement ID

In the following example,
<hostName> is the name of the host on which IdentityIQ is installed
<entitlementId> is the identifying number for the entitlement

```
https://<hostName>/identityiq/ui/rest/redirect?rp1=/ui/index.jsf&rp2=accessRequest/
review?entitlement=<entitlementId>
```

Multiple Entitlements for Specified User Using Entitlement Application/Attribute/Value

Note: If you define only one attribute defined for an application, the entitlementAttribute can be omitted and it will be filled in automatically. In all other cases, the attribute is required. In all cases, entitlementApplication and entitlementValue are required for each entitlement combination.

In the following example,
<hostName> is the name of the host on which IdentityIQ is installed
<identity1> is the name of the user
<entApp1> and <entApp2> are the entitlement applications
<entAttrib1> and <entAttrib2> are the entitlement attributes (such as memberOf or groupmbr)
<entValue1> and <entValue2> are the entitlement values

Note: In the following example, two entitlements are requested.

```
https://<hostname>/identityiq/ui/rest/redirect?rp1=/ui/index.jsf&rp2=accessRequest/
manageAccess/add&identityName=<identity1>&filterEntitlementApplication=<entApp1>&fi
lterEntitlementAttribute=<entAttrib1>&filterEntitlementEntitlement=<entValue1>&filt
erEntitlementApplication=<entApp2>&filterEntitlementAttribute=<entAttrib2>&filterEn
titlementEntitlement=<entValue2>
```

Direct Link to Pending Work Items (Mobile)

IdentityIQ supports the following mobile work items:

- Forms
- Approvals
- Request Violations

For all other types of work items, go to the desktop version of IdentityIQ and access the page associated with the work item.

You can link directly to any open work item such as a form or a violations. To access a direct link, a user must be logged in, have visibility to the work item and have authorization to access the item.

Note: Some work items, such as manager access reviews, are not supported as direct links. If a direct link contains a work item id that is not supported, a warning message displays that indicates the work item does not exist.

In the following example,
<hostname> is the name of the host on which IdentityIQ is installed
<workItemid> is the identifying number for the work item

```
https://<hostname>/identityiq/ui/rest/redirect?rp1=/ui/index.jsf&rp2=commonWorkItem
/<workItemid>
```

Using Direct Work Item Links in Email Templates

When you send an email with a direct link to a pending work item to a user, the email system variable must be configured to match server name and path of the currently deployed instance of IdentityIQ. Click the **Gear** icon in the navigation menu bar and go to **Global Settings -> Mail tab -> Email Templates -> Server Root Path**. For example, the default is set to `https://localhost:8080/`IdentityIQ. However, if you deploy from `/`spt on port 80, you should change the setting to `https://localhost/`spt.

Note: The $spTools.formatURL() is a velocity template function that formats the url correctly in the actual email sent to the user.

```
$spTools.formatURL('/ui/index.jsf#/commonWorkItem')/$item.id
```

Chapter 3: Using the Administrator Console

Use the Administrator Console link, under the gear icon, to access the Administrator Console and view the Task, Provisioning, and Environment monitoring tables.

Access to the Administrator Console is controlled with IdentityIQ rights.

Manage Task Results

Use the Tasks table to view host affinity check run time data. From this page you can also postpone a scheduled task, terminate a running task, or dump a stack trace of a running task. The stack trace is typically used when the task is running long and to diagnostics is desired.

Use the tabs at the top of the table to limit your view by task status: Active, Scheduled, or Completed. Use the **Filter** options or search field to further limit the tasks displayed.

Active Tab

This tab displays all of the tasks that are currently running.

Use the Actions column to terminate a running task or request a stack trace, if a task is running long and you would like to see diagnostics.

Table 33—Manage Tasks - Active Tasks Tab

Column	Description
Name	Name of the task
Type	Task type
Start Date	Name of the task
Owner	The task owner, not necessarily the identity who requested the task be run
Host	Host on which the task is currently running
Current Runtime	How long the task has been actively running
Average Runtime	The time that this task has historically taken to complete

Scheduled Tab

This tab displays all of the tasks that are scheduled to run in the future, including those that are scheduled to run periodically, for example Perform Maintenance.

Use the Actions column to postpone a scheduled task or delete a schedule. No instance of a postponed task will be performed until after the selected date.

Table 34—Manage Tasks - Scheduled Tasks Tab

Column	Description
Name	Name of the task
Type	Task type
Task	Name of the task
Host	Host on which the task is scheduled to run
Next Execution	The next time this task is scheduled to run
Last Execution	The last time this task was executed
Last Result	The result of the last run, for example Success or Failed
Owner	The task owner, not necessarily the identity who requested the task be run

Complete Tab

This tab displays all of the tasks that have completed, regardless of the result.

Table 35—Manage Tasks - Completed Tasks Tab

Column	Description
Name	Name of the task
Type	Task type
Result	The result of the last test run
Start Date	The date and time at which this task began
Date Complete	The date and time at which this task stop running
Owner	The task owner, not necessarily the identity who requested the task be run
Host	Host on which the task was run
Average Runtime	The time that this task has historically taken to complete
Runtime	The actual runtime
Diff from Average	The difference between the actual and average runtimes

Manage Provisioning Transaction Results

Note: This feature can be disabled and might not appear in your instance of IdentityIQ. Contact your system administrator for details.

Use the Provisioning Transactions table to view the status of all provisioning transactions in your implementation of IdentityIQ; connectors, manual work items, and IdentityIQ operations.

Use the tabs at the top of the table to limit your view by transaction status: All, Failure, Success, or Pending. Use the **Filter** options or search field to further limit the transactions displayed. The logging level is controlled by a system setting. If you are not seeing all of your transactions, contact your system administrator.

Use the report/download button to launch a Provisioning Transaction Object report in the background. From the Report Launched window, use **Get Email Notification** to receive an email when the report is complete, or **View Report** to display the Report Results page.

Click the information icon for any transaction to view detailed information. The Transaction Details window provides very detailed information, including the reasons for a Failed or Pending status. After viewing take the appropriate actions to correct the reasons for the failure or delay, you can use the **Override** or **Retry** options to proceed with the provisioning process.

Use **Override** to manually create a work item to take action on failed transactions.

Use **Retry** to manually retry the provisioning transaction. The retry option is only available on transactions that are in the Pending state, and only for transactions that were created with the retry options enabled. This button overrides the reset counter configured in the transaction.

Failed transaction cannot be retried, you must use the override option to create a new work item for those transactions.

The information contained on this page is also available in two reports, the Provisioning Transaction Object Report and the Detailed Provisioning Object Report.

Manage the information displayed on this page from the Miscellaneous tab of the Configure IdentityIQ Settings page of Global Setting, found under the gear icon.

Monitoring Your Environment

The Environment Monitoring page provides insight into each defined Application's health and the status of your Modules and Extensions. This helps diagnose issues with connectivity within the environment.

The Application view provides a view from an Application up perspective. Each Application is listed, along with a summary of all statuses reported by all configured Hosts.

Click **Columns** to select and arrange the information displayed on the pages. Use the search field to locate specific information.

Use the gear icon in the title bar to define global settings for all hosts in IdentityIQ. These settings are used for all hosts that have not explicitly over-ridden the defaults.

Hosts

This tab displays all of the hosts associated with an IdentityIQ instance.

Click on a name in the **Host Name** column to show all ServerStatistics captured for the selected host, grouped by Snapshots. The snapshots can be cycled using the previous/next arrows, or selected by name using the drop-down list.

Use the action buttons in the **Host Action** column to configure or delete hosts. The Host Setting dialog enables you to specify the services running on each host and configure host monitoring.

Deleting a host will remove all associated server statistics, as well as the Server object. The host will no longer appear in the list of hosts after deletion. However, if the underlying server is still running, the host will reappear the nest time its heartbeat service runs. All configuration settings for a re-generated host will use defaults for the its list of services, and for the monitoring service configuration.

Services

This enables a specific host to enable/disable services. The one exception is the Request Service. The Request Service cannot be fully shut down. The Host, if service shutdown, still processes requests that have been specifically targeted to that host, but will not pick up generic requests (un-targeted requests).

Configuration

Note: The Application Monitoring does not adhere to the restore defaults.

The Configuration tab enables host specific monitoring configuration. This enables you to override the global defaults for Polling Interval and Statistics Retention, and to enable and disable given retained statistics.

Click **Use Default Settings** to clear all host specific overrides revert back to the global defaults.

The Configuration tab also allows selecting Applications in which to monitor health. When selected, the Application is contacted each time the monitoring service runs, and the health check status is recorded.

Applications

Note: An application must be monitored by at least one host before it will report statistics.

The Application tab provides a view from an application up perspective. Each application is listed, along with a summary of all statuses reported by all configured hosts. Monitoring can be run from any number of servers, on any subset of applications.

Click an application name to display a panel containing more detailed information about a the application's statuses. This shows each host that has reported a status for the application, as well as the status, and time of ping.

If you have full access rights, click the refresh icon to schedule a request on the host to perform a health check for the application. The refresh icon is disabled until the request is fulfilled.

SailPoint Modules and Extensions

The SailPointModules and Extensions tab provides a list of all installed modules and extensions and a summary of all statuses reported. Click a module or extension name to see a list of reported statuses.

Chapter 4: Working with Plugins

The SailPoint Plugin Framework is an extension framework model for IdentityIQ. It enables third parties to develop rich application and service-level enhancements to the core SailPoint platform. It enables plugins to extend the standard user interface, deliver custom REST endpoints, and to deliver custom background services.

A plugin can be a simple REST service or a full page application on top of IdentityIQ. A plugin can consist of one or all of the following components.

- A client side front end
- REST web services
- ServiceDefinition, PolicyDefinitions, and TaskDefinition implementations
- Java classes available for scripting
- Custom plugin configuration
- Database tables

During your initial installation, IdentityIQ is set up to work with plugins. A separate plugin table, identityiq`Plugin`, is created as part of the database schema creation scripts and the `plugins.runSqlScripts`, `plugins.importObjects`, and `plugins.enabled` properties are set to `true` in the iiq`.properties` file.

To disable plugins completely in IdentityIQ, set the plugin property values to `false`.

Plugin Framework

The plugin framework manages the installation and loading of plugins. It provides:

- Class path isolation on the server side
  - Implementers are free to use any 3rd party libraries or technology they choose. As long as it can be served from a REST end point, a background service, or a Java class called from scripts.
- JavaScript isolation on the client side
  - Implementers are free to use any 3rd party client side libraries.
- Core code protection
  - The framework insures and certifies no plugin overrides or changes backend product code behavior. Essential for security and upgrading.
- Web service extensions
  - Implementers can define custom REST end points to push and pull data between their plugin and the SailPoint data model.
- Plugin installation and removal
  - Plugins can be dynamically loaded to provide drag and drop installation and removal, or you can choose to require installation prior to application startup.

A plugin's user interface can be as simple as a piece of JavaScript or text injected on an existing page or a full page plugin. The behavior is defined by the `manifest.xml` in the plugin's root directory.

Working with Plugins in IdentityIQ

Note: The plugin feature must be enabled in IdentityIQ and you must have the proper access, for example System Administrator or Plugin Administrator, before this page can be displayed.

The Installed Plugins page displays and enables you to manage your plugins from within IdentityIQ. Open the page by selecting Plugins from the list under the gear icon.

From the Installed Plugins page you can install, uninstall, enable, disable, and configure your plugins.

Install — click New and either drag and drop a zip file onto the page, or navigate to the directory containing the plugins.

Enable/Disable — click the power button icon to enable or disable plugins. You will be asked to confirm your decision.

Uninstall — click the x icon to uninstall a plugin. See “Plugin Installation and Removal” on page 86 for additional information on removing a plugin.

Configure Plugins Page

The Configuration page enables you to view detailed information about the plugin, including its version, installation date, and certification level.

This page also enables you to change the values of the Settings objects for the plugin, based on the object type. If no Setting objects were defined when the plugin was created, none will display on the Configuration page.

Working with Plugins from the IdentityIQ Console

The IdentityIQ console has a number of commands that enable you to manage plugins from there, as well as perform scripted installation of multiple plugins.

The `iiq console` contains the following commands:

- **plugin install** — Installs a single plugin or multiple plugins in a directory, see “Install” on page 80
- **plugin upgrade** — Upgrades a plugin, “Upgrade” on page 81
- **plugin uninstall** — Uninstalls a plugin, “Uninstall” on page 81
- **plugin enable** — Enables a plugin, “Enable” on page 81
- **plugin disable** — Disables a plugin, “Disable” on page 82
- **plugin export** — Exports a plugin and all of its current configuration to a zip file in a specified directory, “Export” on page 82
- **plugin status** — View the enabled status of a plugin, all plugins or whether or not plugins are enabled globally as defined in the iiq.properties file, “Status” on page 82
- **plugin list** — View a list of all installed plugins, “List” on page 82
- **plugin classes** — “Classes” on page 82

Install

Install a single plugin or multiple plugins. Either a path to the zip file of the plugin or a directory containing multiple plugin zip files can be specified. If a directory is specified, any zip file in that directory is installed.

Flags:

- file — path to a plugin file
- dir — the directory containing the plugin zip file to install
- no-cache — the plugin should not be cached after install

Upgrade

Note: Refer to the "Plugin Versioning Requirements" on page 75

Upgrade to a newer version of a plugin.

Upgrading a plugin to the same version or a previous version is not supported. While developing a plugin, this behavior can be disabled for easier testing. To do so, include a "-dev" suffix on the version, for example, 2.0-dev.

The version of a plugin can either be official or development.

Development versions end with the suffix '-dev,' for example, 2.0-dev, and bypass most version checks so that the plugin can be recompiled, upgraded and tested easily.

Official versions drop the '-dev' suffix and can only be installed over a development version or an earlier official version. The minimum upgradeable version must also be valid.

Valid upgrade paths:

- 1.0 -> 2.0-dev
- 2.0-dev -> 2.0-dev
- 2.0-dev -> 2.0
- 1.0 -> 2.0

Invalid upgrade paths:

- 2.0 -> 2.0
- 2.0 -> 1.0

Flags:

- file — path to a plugin file
- no-cache — the plugin should not be cached after the upgrade

Uninstall

Uninstall a plugin.

Flags:

- id — plugin id
- name — plugin name

Enable

Enable the plugin.

Flags:

- id — plugin id
- name — plugin name
- no-cache — the plugin should not be cached after being enabled

Disable

Disable a plugin.

Flags:

- id — plugin id
- name — plugin name

Export

Export a single or all installed plugins to their respective zip files and, optionally, a specified directory.

Flags:

- id — plugin id
- name — plugin name
- * — export all installed plugins
- dir — the directory in which to save the zip files. If the directory does not exist, the command will attempt to create one. If none is specified, the files are save to the current working directory.

Status

The enabled status of a single plugin, all installed plugins or the system-wide enabled status of plugins.

Flags:

- id — plugin id
- name — plugin name
- * — view the status of all plugins
- no flag — system-wide status of plugins as defined in iiq`.properties`

List

The list of all install plugins.

Classes

The list of the classes available (from a plugin or all plugins), and the intended use for each class.

Flags:

- id — plugin id
- name — plugin name
- * — the list of all available classes from all plugins

Developing Plugins

IdentityIQ stores the .zip archive file of the Plugin in the IdentityIQ database in a data LONGBLOB in the spt_file_bucket table. The data in the spt_file_bucket table is referenced ID to an entry in the spt_persisted_file table.

Plugins are loaded from this .zip file after installation or after an application server restart. The .zip file is extracted, and all important files are cached for later use. There are several accessor methods to reference the cached files, but they can also be referenced by the url prefix `/identityiq/plugin/`*`pluginName`* followed by the path found in the build structure. Compiled java classes are loaded and cached from the .zip archive using the PluginClassLoader class.

Plugin Versioning Requirements

Note: The single exception to these requirements are version numbers with -dev appended to the end. This suffix causes version number validation to be bypassed.

To provide better support for upgrading plugins, we have set new requirements for plugin version number formats. Plugin version numbers must be numeric, contain no alphabetic or other characters, and separate the elements of the version number with decimal points. Within each segment of the version number, the values between the decimal points, the values are cast as integers, and leading zeroes are trimmed.

For example:

- `04` and `00004` are both interpreted as `4`
- A segment containing any non-numeric values is interpreted as `0`
- `1.004.alpha` is parsed as `1.4.0`
- `2.3.4a` will be parsed as `2.3.0`

Plugin Object Model

A plugin is defined in IdentityIQ by the Plugin XML object. This object defines the parameters of the plugin, for example items such as REST Resources, Snippets, Widgets, and Settings. This Plugin object is defined in the `manifest.xml` file. The Plugin Object is an XML object that defines the features of the plugin. This object tells IdentityIQ what features are in your plugin by defining them as attributes of a Plugin Object. In the Plugin Object you also define items such as the name of the plugin, the rights required for using the plugin, version, snippets, and REST resources.

The following attributes are included in the plugin model:

Table 36—Plugin Model Attributes

Attribute Name	Description
name	Unique Name of the Plugin
installDate	Date that plugin is installed

Table 36—Plugin Model Attributes

Attribute Name	Description
displayName	Display Name for the plugin
disabled	Status of the plugin
rightRequired	What SPRIGHT is required for this plugin
version	The version of the plugin
minSystemVersion	The minimum version of IdentityIQ that the plugin will run on
maxSystemVersion	The maximum version of IdentityIQ that the plugin will run on
attributes	List of configurable attributes
file	Reference to the persisted file in the database

Plugin Structure

A plugin will consist of the following components:

- Manifest file
- Build file(s)
- Database Scripts
- UI Elements
- XML Artifacts
- Java Classes
- Java JAR libraries

Not all of these components are required for a plugin - it can be as basic as the manifest, and some javascript/xhtml pages. In order to understand how a plugin operates, and how best to create one, it is important to understand what each of these components does, and how they interact.

Plugin Manifest File

A plugin is defined in IdentityIQ by the Plugin XML object that defines the parameters of the plugin. For example, features such as REST resources, Snippets, Settings. The Plugin object is defined in the `manifest.xml` file. This is a required artifact.

For more complex plugins that require support for other field types, and more dynamic behavior, such as drop down lists or password fields, use the advanced plugin settings to define a form or reference a custom plugin configuration file.

Dynamic behavior might include showing or hiding additional fields depending on previous selections. For example, if a user chooses basic authentication, a username and password field would appear, but, if oauth authentication is chosen, it might be more appropriate to show an access token field.

Plugin Settings

Note: If your plugin requires more than simple input fields, string, boolean or int values, you must use the plugin advanced settings.

Plugin settings are attributes that are available for modification as part of the installation. Click **Configure** to display the configuration settings page. Settings are displayed as a form. If the plugin does not use the plugin advanced settings, the form is created automatically.

Settings from the manifest file are listed, in order, on the plugin settings page.

The Plugin setting object can be used to represent a single setting on the configuration settings page for a Plugin. Each object is used to represent a single configurable setting on the settings page.

Table 37—Plugin Settings

Attribute Name	Description
allowedValues	List of allowed values for population of a dropdown
dataType	The type of the setting, for example, "string" or "int" or "boolean"
defaultValue	The default value for the setting
helpText	Associated help text for the setting
label	Label to be displayed for the setting
name	Name of the current setting
value	Value for the setting

Plugin Advanced Settings

Plugin advanced settings are used to define forms or reference a custom plugin configuration file to define more complex plugins.

- settingsForm — define a plugin using a form
- settingsPage — reference a custom HTML/JS file

settingsForm

To use a form for plugin configuration, you can build a form using the form builder then copy and paste that form into the manifest file, or you can build the form directly into the manifest.

Values entered into a form can be accessed using the FormData.values. FormService has functions to assist with validating required fields and displaying errors. Additional validations are built into the HTML and AngularJS code based on the form design, which means Angular will set a field to undefined if it is not valid. These validations can be used to prevent a form from being submitted and show error messages if necessary.

settingsPage

Develop your own configuration settings page by providing the required HTML and javascript. You can use whatever frameworks you prefer for your settings, but they need to fit in with whatever IdentityIQ has loaded. For example, angular is not required, but you can use it.

To use angular frameworks, see “SailPoint Angular Components” on page 93.

Use the settingsPage setting to specify the name of your custom configuration settings page, for example, `config.xhtml`.

Snippets

Snippets are small, configurable snippets of code that can be injected into the rendering of normal IdentityIQ user interface pages. For example, you can insert a menu option, a button, or even a larger set of interface components into an IdentityIQ page.

Snippets must be specified in the plugin's manifest file. They use a regular expression pattern to identify the IdentityIQ pages where the snippet should execute, and therefore appear.

You can have multiple snippet components in the same plugin, some that apply globally, and some that apply to specific targeted pages. You need to define a separate `js` file for each location a snippet applies, and then specify a separate snippet in the manifest file with the right regexPattern to run it on the appropriate pages.

The details for the user interface component's contents, and its placement within the page, are specified in the JavaScript file.

A snippet contains four equally important components:

Table 38— Snippet Components

Component Name	Description
regexPattern	This is a regular expression pattern that is run against the current URL in the browser - if the URL matches the pattern, the Snippet will attempt to displayed
rightRequired	This determines the scope of users allowed to view the Snippet element - should reference an IdentityIQ SPRight object
scripts	This is a list of the scripts to run when a particular URL matches the regexPattern. Normally this will consist of injecting an element into the DOM of the page. The example `hearder.js` file uses JQuery
styleSheets	List of any css files that are required by Snippet Scripts

Widgets

A Widget is a targeted snippet – one that inserts a block of user interface components into a fixed area of the Home page that can be added selectively for different users, as a unit.

Widgets can be configured to appear on the Home page for any or all IdentityIQ users.

The first thing you need, to implement a plugin Widget, is the Widget object itself. When you import that object into IdentityIQ during plugin installation, it defines the existence of the Widget making it available for any user.

Widget objects are simple, the only details about the user interface component that get defined in the object are its name and title.

Widgets require a snippet definition in the manifest file for this plugin. This snippet defines the home page hook for the widget. The regular expression pattern for the widget snippet must specify the IdentityIQ home page.

The rest of the snippet definition has IdentityIQ execute the contents of the specified JavaScript file when it loads pages that meet the regex pattern.

The contents of the JavaScript file then define both the user interface layout, in the form of a directive, and the controller for the Widget. Because the home page is an Angular page, this JavaScript must specify an Angular controller for the widget.

The name given to this directive must follow a fixed naming convention. It must be specified in relation to the name given to the widget object. Specifically, the widget object name must be prefixed with sp and suffixed with Widget. So the Search widget object requires a directive called sp*WidgetName*Widget.

The directive references the controller for the widget. That controller is also defined within the widget's JavaScript file and defines the variables that serve as the model for the view elements and performs the required operations to set their data values.

Plugin Build File

Note: Complications can arise when the plugin is built using a different version of java than the version deployed on the application servers hosting IdentityIQ. Parametrize the javac argument in the build.xml file with the most compatible Java version available. To do this, add the property target to the javac directive, and set equal to whatever version is being targeted.

For example:
<javac srcdir="${pluginSrc}" destdir="${pluginClasses}"
includeantruntime="false" target="1.7">

Apache Ant is a readily available tool that can be used to package plugins prior to deployment and distribution. To provide build specific values, the standard is to also include a `build.properties` file with a simple key-value pair for all build specific tokens.

The following example illustrates how a properties file can be leveraged to enable multiple developers to use the same build process, despite having dissimilar build environments. The actual `build.xml` file is responsible for creating the build directory, compiling any java classes, packaging those compiled classes into a .jar archive, and archiving in .zip format the complete plugin.

```
jdk.home.1.7=/Library/Java/JavaVirtualMachines/jdk1.8.0_66.jdk
iiq.home=/usr/local/apache-tomcat-8.0.30/webapps/identityiq/
pluginName=TodoPlugin
version=2.0.0
```

Plugin Database Scripts

Plugins that require persistence of data outside of that allowed by the IdentityIQ object model require at minimum the creation, updating, and deletion of unique tablespace. The plugin framework creates a database named identityiqPlugin. The creation of this database is handled by the installation scripts packaged with every download of IdentityIQ, in the `WEB-INF/database` folder. Additionally, a default user identityiqPlugin is created to perform operations, installation and deletion of plugins, on this new database. Similar to the base IdentityIQ username and password, these can be modified and updated in the `IdentityIQ iiq.properties` file located in `WEB-INF/classes/iiq.properties`.

When creating a plugin, you must create a folder named db in your project directory. This folder should be further subdivided into three operation specific folders: install, uninstall, and upgrade.

The scripts placed in these folders are automatically run when a plugin is installed or deleted. It is recommended that you include scripts for the four major database types supported by IdentityIQ, MySQL, SQLServer, DB2, and Oracle, or note in your documentation which databases are supported. Database specific scripts must include the database type as the file extension, for example `.mysql`. The `upgrade` folder should contain any deltas in table definitions from prior versions of the plugin.

Plugin User Interface Elements

Most plugins have some additional user interface component that appears in IdentityIQ. Images, CSS files, HTML templates, and JavaScript can all be used to provide the interactions and views required by the plugin. Plugins that use a fullPage element look for a file called `page.xhtml` in the build.

To extend the classes loaded with your plugin to the rest of IdentityIQ, you must specifically declare those classes in the manifest file.

Plugin Authorization

To prevent unauthorized access to your new endpoints, each should be guarded with an authorization mechanism. You can constrain which users can see and access the user interface components, and you can secure the REST endpoints you build into your plugin.

When you define snippets, including widget plugin components, in the manifest file for a plugin, you can specify a rightRequired attribute to constrain access. This attribute names a SailPoint SPRight which users must be assigned for the component to appear in their IdentityIQ instance.

You can also specify a rightRequired at the Plugin object level, in the manifest file, which will specify the required SPRight for a user to be able to access the full-page component of the plugin.

If you leave these rightRequired attributes off, all IdentityIQ users will be able to access those plugin components.

SPRights are the most granular permission object in IdentityIQ. In most cases, users are assigned SPRights in IdentityIQ by attaching those rights to one or more Capability objects and then granting the Capability to the appropriate users.

If the plugin contains a full-page component that users can access through a quicklink, the Quicklink access will be governed by Quicklink Populations, like any other IdentityIQ Quicklink.

User must also be authorized to the full page itself, with the rightRequired specified in the plugin manifest, to be able to view the page.

Other authorization points of note. First, whether or not you explicitly authorize system administrators to these components, they will have full visibility and access to them. Second, when you include a widget in your plugin, the widget will appear in the list of available widgets for all users when they are editing their home page and deciding which content to include there. However, if they are not authorized to the widget's snippet, and they add that widget to their Home page, IdentityIQ will add an empty widget and they will neither be able to see nor interact with any of its functional elements.

To secure the endpoints the plugin framework use Annotations. In Java, an annotation is a syntactic metadata that is added, often before a method signature, that describes the parameters used in that method.

An annotation should have at least three parts

- The HTTP method (GET, POST, PUT, DELETE, etc)
- The path or endpoint - this can be parameterized which is useful for pulling back a single record. The above example uses parameterization by adding the variable within {} tags to the end of the URL, and also declaring the @PathParam appName in the input arguments of the method signature
- The authorization of the method - the allowed values are:
  - **@AllowAll** - this allows anyone to interrogate the endpoint
  - **@RequiredRight("<SPRight>")** - allows users who possess the named SPRight to access the endpoint
  - **@SystemAdmin** - system administrator access only
  - **@Deferred** - Authorization is deferred to the method. When this option is selected, you must also create an Authorizer class that implements the sailpoint.authorization.Authorizer interface. The Authorizer class should overwrite the authorize(UserContext) method of the base Authorizer interface. Inside of the REST resource method, the author would then call authorize().

Plugin XML Artifacts

Any IdentityIQ objects that are required as part of a plugin need to be represented in XML artifacts. This could be something as small as a single new SPRight object or a complex workflow or rule. The mechanism that is used for importing these artifacts during installation is the same as any IdentityIQ object import, so the normal import actions are also available, merge, include, execute, logConfig.

Development of these XML artifacts can be done directly in the build folder, or in the IdentityIQ user interface and either exported using the console or copy and pasted from debug into the build.

When developing in the user interface and then migrating to your build folder using cut and paste, you must remove the id attribute assigned by Hibernate and any other hibernate ID value references. For this reason, it is preferable to export the artifacts using the IdentityIQ console command **./iiq export -clean**.

Everything in the `import` folder is imported The objects can be separated into individual files, or combined into a single file. When a plugin is uninstalled, the XML artifacts that were imported remain in the IdentityIQ database, but the `.zip` archive from which the plugin files was loaded, is removed from the spt_file_bucket and spt_persisted_file tables.

Plugin Java Classes

Plugins are a powerful productivity-enabler, that give users the ability to extend both the IdentityIQ user interface and server in a well-defined manner.

Plugin Java Classes - REST Classes

The plugin framework relies heavily on REST web services integration for the majority of CRUD (create, read, update, and delete) operations. To create a custom REST Resource:

- Extend the BasePluginResource class
- Secure the new endpoints

Extend BasePluginResource

The BasePluginResource class should be used as the base class for all resources. It provides access to utility methods for accessing plugin settings, getting database connections, and more.

- **getConnection** - gets connection to the datasource specified in the `iiq.properties` file for the plugins
- **getPluginName** - this method should be overwritten to return the correct name of the plugin
- **getSettingBool** - gets value of boolean plugin setting for plugin name returned by getPluginName()
- **getSettingInt** - gets value of int plugin setting for plugin name returned by getPluginName()
- **getSettingString** - gets value of String plugin setting for plugin name returned by getPluginName()
- **prepareStatement** - convenient security method for getting Java PreparedStatement object for any database queries that are required
  - signature is prepareStatement(Connection, String, Object...) where the String would be the SQL statement you want to execute
  - Object... would be a list of the parameters values, if any, to be used
- **authorize** - should be overwritten by implementers, but by default only ensures that SystemAdministrator can see everything

Introduce additional methods to handle the various endpoints required by the plugin.

Plugin Java Classes - Plugin Executors

The plugin framework enables you to include custom task implementations or services with your plugin. These items rely on executor classes that contain the business logic for these services. The following executors are currently available:

- Service Executors
- Task Executors
- Policy Executors

You must specifically declare the classes to be exported for each of the executors. Only classes specifically declared are accessible from the rest of IdentityIQ. If a class is not declared, it will fail to instantiate when you open the plugin. Classes are declared in the `manifest.xml` file.

Use the following attributes to declare classes:

- serviceExecutor
- taskExecutor
- policyExecutor

For example:

```
<entry key="taskExecutors">
  <value>
    <List>
      <String>com.acme.TaskExecutor1</String>
      <String>com.acme.TaskExecutor2</String>
    </List>
  </value>
</entry>
```

Plugin Object Properties

When defining your plugin object you must provide the list of service executors that are included. The list lives inside an attributes map under the key serviceExecutors.

- Plugin Helper methods
- All inherited Service methods
- BasePluginTaskExecutor
- Plugin Helper methods
- All inherited TaskExecutor methods
- BasePluginPolicyExecutor
- Plugin Helper methods
- All inherited PolicyExecutor methods.

Plugin Helper Methods

The list of methods that are included with the BasePlugin classes are as follows:

- getPluginName() - returns a string value of the name of the plugin
- getConnection() - returns a Connection object used to query the database
- getSettingString(String settingName) - returns a String setting value from the Plugin Settings
- getSettingBool(String settingName) - returns a boolean value from the Plugin Settings
- getSettingInt(String settingName) - returns a integer value from the Plugin Settings

You can think of the BasePlugin classes as foundation for creating your specific objects. By using them you gain access to the Plugin Helper Methods, but you are not required to use the BasePlugin classes. You can extend directly from the parent class object.

Implementing a Plugin Service Definition

To implement a Plugin Service there are two parts. The service class, containing the business logic that you want the service to actually do, and the service definition xml, that is loaded into IdentityIQ.

- BasePluginService Class — an abstract class that extends the Service class, as well as implements the PluginContext interface. You can use this class as the foundation for your custom Plugin Service
- Service Definition — specify a pluginName attribute. This tells IdentityIQ to use the plugin class loader for this executor. If this attribute is not specified the executor class will not be found

Implementing a Plugin Task Executor

To implement a Plugin Task Executor there are have two parts. The Task Executor class, which handles the business logic for your task, and the TaskDefinition xml object, which gets loaded into IdentityIQ.

- BasePluginTaskExecutor Class — an abstract class that extends the AbstractTaskExecutor class, as well as implements the PluginContext interface. You can use this class as the foundation for your custom Plugin Executor task
- TaskDefinition — include the pluginName attribute, as this attribute tells IdentityIQ to use the plugin class loader instead of default class loader. If the attribute is not specified the executor class will not be found

Implementing a Policy Executor

To implement a Policy Executor there are have two parts. The Policy Executor class, which handles the business logic for your policy, and the Policy xml object, which gets loaded into IdentityIQ.

- BasePluginPolicyExecutor Class — an abstract class that extends the AbstractPolicyExecutor class, as well as implements the PluginContext interface. You can use this class as the foundation for your custom Plugin Policy Executor
- Policy XML — include the pluginName attribute, as this attribute tells IdentityIQ to use the plugin class loader instead of default class loader. If the attribute is not specified the executor class will not be found.

Plugin Java Classes - Script Classes

Note: Beanshell executions are referred to as scripting in this document.

The classes installed for plugins can be made available to all beanshell executions. Beanshell (rules, scriptlets, workflows) scripting invocations are able to use all Java classes, from all plugins, that are declared as exported for scripting. The scripts should fail to load classes which are in plugins, but which are not explicitly exported.

Beanshell executions include:

- rules
- workflow steps (rules and scripts)
- scriptlets

Use the scriptPackages attribute to declare the Java packages exported for use by scripts as follows:

```
<entry key="scriptPackages">
  <value>
    <List>
      <String>com.acme.classy.util</String>
    </List>
  </value>
</entry>
```

Example: Using Plugin Classes From a Rule

This is a simple example of how to call the plugin classes from a rule.

The example Java class named `com.acme.classy.util.ClassyUtil` can be used from a rule if declared properly in the `manifest.xml` using the `sriptPackages` entry shown above.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Rule PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<Rule language="beanshell" name="ClassyRule" >
  <Description>Returns the current timestamp by calling class from
                      ClassyPlugin</Description>
```

```
  <Signature returnType="string"/>
  <Source>
import com.acme.classy.util.ClassyUtil;
long now = ClassyUtil.now();
return "The current timestamp is: " + now;
 </Source>
</Rule>
```

SailPoint Angular Components

To implement the SailPoint-styled angular components, your project needs to include the SailPointBundleLibrary JavaScript file. There are specific directive dependencies on this library when you use the SailPoint-styled components.

Widgets and snippets donot require this library in the plugin project, because for those, the plugin architecture automatically references this library from your IdentityIQ instance. But if you are going to use these in a full page plugin implementation, your plugin needs to include a copy of this JavaScript library, copied from your IdentityIQ version. Plus, your `page.xhtml` component needs to explicitly reference the JavaScript file in a script element so IdentityIQ can resolve the SailPoint Angular directives you are be using. This library needs to match the version of IdentityIQ where the plugin will be deployed.

Your angular module definition needs to specify any other modules that your module has dependencies on. This list will vary, depending on the elements you include. For example, if you are using modal boxes, you need to include `sailpoint.modal`, tree components rely on `sailpoint.tree`, and so on. If your user interface only contains some of the more basic of the field types, you need fewer modules in your dependency list.

An example plugin that demonstrates the different components that are part of the SailPoint Angular bundle, called the Kitchen Sink plugin is installed with IdentityIQ. The plugin is available for download from Compass. Once the plugin is downloaded, you can install and view the plugin through IdentityIQ to see how the different field types behave. Then you can examine the HTML and JavaScript files to see the directives involved and how they are populated.

The library contains a wide array of field types, just like you might see throughout core product pages. For example, selection lists, plain text entry fields or fields where you can enter multiple separate text values as a list, dropdown list boxes where you can select from a set of choices, and a special directive for when the choices should be Boolean true-false values, date pickers, checkboxes, and radio buttons.

The library offers SailPoint styled buttons, and the demo plugin shows how you can use them to attach logic to do things like open modal boxes of various types, post an alert notification to the page or navigate the user to another page in IdentityIQ while preserving navigation history.

Beyond the simpler field types and buttons, there are a few components that create more complex elements. For example, a tree directive for showing nested relationships in data, a directive for displaying a list of cards that is configured for paging when result sets are large, and a directive for a data table that matches the tables in the Angular pages of IdentityIQ.

Internationalization

Message catalog files are specified per language with the two character abbreviation for the corresponding language, or the four-character options when you have locale-specific message catalogs, for example, `TrainingPlugin_fr_ca.properties` for Canadian French. These files should be recorded in the messages folder of your plugin project.

To guard against collisions with IdentityIQ base product message key names, or message keys from other plugins, the best practice is to name your plugin's message keys with a prefix that makes them unique to your plugin. For example, consider using your plugin's name as a prefix.

Both the user interface and server side code can access the provided catalogs.

In the user interface, full page plugins and widget plugins use two different mechanisms, because of differences in library support that are included in the pages.

In a Full page plugin component, the HTML uses the `msgs` function to translate the text.

In Widgets or other snippets that display text, the `spTranslate` function in the SailPoint Bundle Library does the translation, by piping the message key through it.

It is possible to use this syntax in a full page plugin, but only if you have included the SailPoint Angular Bundle Library and have declared a dependency for your angular module on the `sailpoint.i18n module`. The `spTranslate` function is part of that module.

In server-side code, localization is done with the Message object's localize method, passing it the message catalog key to look up and translate.

Plugin Installation and Removal

To install a plugin, click the gear icon and select **Plugins** to navigate to the Installed Plugins page. Click **New** and drag and drop or upload the plugin `.zip` file.

If you downloaded a plugin, the `.zip` file should be included with the download. If you developed the plugin yourself, the `.zip` file is in your project directory under `build/`*`your plugin name`*`/dist`.

When a plugin is installed, the database scripts from the `db/install` folder run, which creates any tables necessary for the plugin, the XML configuration files are imported into the IdentityIQ database from the `import/install` folder, any compiled classes are loaded into the unique plugin classloader, and the manifest file is imported creating the Plugin object.

Remove a plugin by clicking **X** on the appropriate Plugin card on the Installed Plugins page. Database scripts in charge of cleaning up data run from the `db/uninstall` folder and the manifest file (the Plugin object) is removed.

Addition steps might need to be taken to edit the System Configuration file to remove objects created by the plugin, such as quicklinks, or to remove tables created in the plugin database.

Chapter 5: Configure Applications

For each application in your enterprise, you must define each application in your enterprise and specify the following items:

- Connection properties
- Relevant attributes,
- Target
- Aggregation rules

Application List Page

The Application List page displays all of the applications currently configured. To access this page, from the menu bar, go to **Applications -> Application Definition**. The Application List contains the following information:

Table 39—Application List Column Descriptions

Column	Description
Name	The name of the application.
Host	Host where the application resides.
Type	The application type, for example LDAP or JDBC.
Modified	The date when the application was last modified.

Use the Configure Application page to add or edit applications. Click on an existing application or click **New Application** to open the “Edit Application Page” on page 95.

Edit Application Page

Note: Do not open multiple tabs or browsers. Opening multiple tabs might overwrite changes made in the other.

Use the Edit Application page to define the applications in your enterprise. Specify the connection properties, relevant attributes, aggregation rules, and activity information for each application.

The information contained on the Configuration, Correlation, Risk, Activity Data Sources, and Unstructured Targets, Rules, Password Policy, and Tiers tabs is determined by the type of application specified on the **Application Type** drop-down list. Use these tabs to define how each application interacts with IdentityIQ.

Note: The Tiers tab is only available for Logical application types. See detailed information about configuring logical applications in the SailPoint IdentityIQ *Direct Connectors Administration and Configuration Guide*.

The Edit Application page opens to the Details page and contains the following tabs:

- "Configuration Tab" on page 98
- "Correlation Tab" on page 104
- "Accounts Tab" on page 104
- "Risk Tab" on page 104
- "Activity Data Sources Tab" on page 105
- "Unstructured Targets Tab" on page 106
- "Rules Tab" on page 106
- "Password Policy Tab" on page 107

For each application enter or edit the following information:

Note: This screen also contains any extended attributes that were configured for your deployment of IdentityIQ. The extended attributes are displayed at the bottom of the tab.

Table 40—Edit Application Page — Details Field Descriptions

Field	Description
Name	The name of the application. This is the named used to identify the application throughout the IdentityIQ application.
Owner	The owner of the application. The owner specified here is responsible for certifications and account group certifications requested on this application if no revoker is specified. Application ownership can be assigned to an individual identity or a workgroup. If the application ownership is assigned to a workgroup, all members share certification responsibilities, are assigned certification request associated with the application and all can take action on those requests.
Application Type	The application type, for example LDAP or JDBC. The **Application Type** drop-down list contains the types of application to which IdentityIQ can connect. This list will grow and change to meet the needs of IdentityIQ users.
Description	A brief description of the application. **Note: You must Save the description before changing languages to enter another description.** Use the language selector to enter description in multiple languages. The drop-down list displays any languages supported by your instance of IdentityIQ. The description displayed throughout the product is dependent on the language associated with the user's browser. If only one description is entered, that is the description used by default.
Revoker	The default IdentityIQ user or workgroup to be assigned revocation requests associated with entitlements on this application. **Note: If no user is specified in this field, all revocation requests are assigned to the to application owner by default.**

Table 40—Edit Application Page — Details Field Descriptions

Field	Description
Proxy Application	Optional: Specify an application to manage accounts and provide a connector and schema settings for this application. A proxy application is an application that handles the processing (aggregation and provisioning) on behalf of your application. Here are three examples of proxy applications: —Multiplex applications: In this case you define an application and, most often, a build map rule that sorts the data out in multiple sub-applications. In that case, the sub-applications have the main application as the proxy. — Similar to the multiplex applications are the connectors for legacy identity management systems such as, BMC, Novell/NetIQ, IBM Tivoli, and Sun/Oracle Waveset. — The Cloud Gateway connector tunnels all aggregation and provisioning requests to the gateway in another network. The gateway then acts on behalf of IdentityIQ. All applications that live in the remote network need to have the cloud gateway connector set as the proxy.
Profile Class	An optional class used to associate this application with a larger set of applications for role modeling purposes. For example, you might set a profile class of XYZ on all of the applications where any user that has read account privileges should be assigned the role XYZ Account Reader. You can then create a single profile for that role instead of a separate profile for each instance of the applications. During the correlation process any user with read account privileges on any of the applications with the profile class XYZ is assigned the role XYZ Account Reader.
Scope	**Note: This field is only visible if scope is enabled.** The scope for this application. If scope is assigned, only the owner of the application or users that control the designated scope can work with this application. Objects associated with this application, for example entitlements in a certification request, are visible to a user with any or no controlled scope, but if a new object is being created, for example a certification schedule, this application does not appear on the select list unless the creator controls the scope assigned. Depending on configuration settings, objects with no scope assigned might be visible to all users with the correct capabilities.
Authoritative Application	Select if this application in an authoritative application. You can specify multiple authoritative applications. An authoritative application is a repository for employee information for your enterprise, for example a human resources application. These might not be at risk applications, but they are the data source from which the majority of the IdentityIQ Identity Cubes are built.
Case Insensitive	Use to cause case insensitive comparisons of account attribute values when evaluating provisioning policy.

Table 40—Edit Application Page — Details Field Descriptions

Field	Description
Native Change Detection	Select this option if this application should be included when IdentityIQ performs native change detection during aggregation.
Native Change Definitions:	
Native Change Operations	Select which operations are included when detecting native change. If no operations are selected, native change detection is disabled.
Attributes to Detected	Indicates which attributes are compared when accounts are modified. If the **Entitlement** option is selected, all entitlement attributes are included. If you select **User Defined**, enter the name of the attributes to compare in the **Attribute Names** box.
Maintenance: Take an application off line for maintenance	
Maintenance Enabled	This application is excluded from provisioning and aggregation during the defined maintenance period.
Maintenance Expiration	The date at which the maintenance will end. If no date is defined, this application will be in maintenance indefinitely.

After adding the application information, click **Save** to save your changes and return to the Application List page.

Configuration Tab

The information displayed on the Configuration tab changes depending on the application type specified. The tab is haste following tabs:

- “Settings Tab” on page 98
- “Schema Tab” on page 99
- “Provisioning Policies Tab” on page 101

Settings Tab

Note: The terms account group and application object are use interchangeably in this document but have the same meaning. Some application can have multiple application objects. An account group can be the name of one of those objects.

A note is displayed at the top of this tab for application configured to use credential cycling. For those applications, the credentials are stored and maintained on a Privileged Access Management (PAM) module, and verification is performed using existing hook points that support the retrieval of passwords from application credential management solutions such as, CyberArk Application Identity Manager (AIM) or BeyondTrust PowerBroker Password Safe. Refer to the SailPoint IdentityIQ *Privileged Assess Management Module Guide* for more information.

The Settings tab contains the information that IdentityIQ uses to connect and interact with the application. Each application type requires different connection information and the fields on this tab are changed accordingly.

For most application types you see account and group object types, certain application types, however, enable you to create multiple application object types, each with their own schema. These application object types are sometimes referred to as account groups and those term might be used interchangeably in discussion around this feature.

Click at the top of the page to add a new object type. This function is only available for if the application type is associate with a connector that is enable to handle multiple application object types, or multiple schema.

This button is also displayed if you recently upgraded your instance of IdentityIQ and the application type now supports multiple schemas. In that case you must add the supported application object type here and then run the Account Group Aggregation task to import the new information.

Multiple application object types can be directly correlated, for example an application object type is also an attribute in the schema of another, or they can be indirectly associated, for example they are both objects (schemas) in the same application. These objects and their associations are tracked throughout IdentityIQ and appear in place such as reports, policy violations, searches, and certifications.

If your enterprise is going to use partitioning for account aggregations, identity refreshes, and manager certification, you must enable that function here. Each application type requires different partitioning information.

This is also where you enable an application for data merging and delta aggregation.

Refer to the connector documentation for detailed information on each of the connectors.

Enter the information on this tab as required by the application type being configured. Click **Test Connection** to verify the information is correct.

Schema Tab

The Schema tab is used to define the attributes for each object type in the application being configured. Use the following fields to define attributes for use with the IdentityIQ application. The field content is dependent on the application being configured.

Refer to the connector documentation for detailed information on each of the connectors.

When initially configuring applications, click **Add New Schema Attribute** to define the attributes for each object. Most application types include a default set of schema attributes. For more dynamic application types (JBDC or DelimitedFile), schemas should be defined manually. Click **Edit** to display the Advanced Properties dialog.

The connectors for some application types enable the automatic discovery of the base schema attributes for those applications. For those application types, click **Discover Schema Attributes** to automatically populate your schema tables. After using the automatic discovery function you must designate the Identity Attribute and Display Attribute for the application.

Click **Preview** to test the respective schema configuration. A pop-up sample table displays to indicate a successful configuration. These tables automatically update when you make changes so that you can use this feature before committing your changes. Only one table can be open at one time. Failures result in an error message specifying the point of failure, for example, a file path and name.

Note: The Preview function does not apply to applications which do not support aggregation.

Table 41—Application Configuration - Schema Tab Field Descriptions

Fields	Descriptions
Native Object Type	**Note: LDAP default types are iNetOrgPerson and groupOfUniqeNames for groups.** **Note: This is a required field.** The type of object with which the attributes are associated. For example, User and Group for Active Directory LDAP or DBA_USER and DBA_ROLES for Oracle.

Table 41—Application Configuration - Schema Tab Field Descriptions

Fields	Descriptions
Identity Attribute	**Note: This is a required field.** The attribute that is used by the IdentityIQ application to identify the object.
Include Permissions	Select this function to automatically add directPermissions to the schema. This option is available for any application that has `DIRECT_PERMISSIONS` in the `featureString`, for example, Oracle, DelimitedFile, and sybase HR. With this option activated, IdentityIQ correctly pulls in permission data for identities.
Display Attribute	**Note: This is a required field.** The attribute that is used as the object name as it appears throughout the IdentityIQ application.
Instance Attribute	The attribute that uniquely identifies a specific instance of an application. **Note: Instance Attributes are not supported for Managed Attributes.**
Remediation Modifiable	Accounts that are remediation modifiable can have their values and permissions modified from the Certification Report page for the identity being certified. Specify the method of modification for this attribute: **Select** — display a select list of all possible values or permissions for this attribute. **Free text** — display a text field in which a certifier can enter any value.
Attributes:	
Name	**Note: Attribute names cannot begin with IIQ_. Attributes with names that begin with IIQ_ are considered internal, reserved attributes and are not displayed in the product.** The name of the attribute.
Description	A brief description of the attribute.
Type	The type of attribute being defined. For example, string or boolean. Select from the drop-down list.
Properties: Click Edit to open the Advanced Properties dialog to edit the attribute properties.	
Managed	Specify attributes to promoted to a first-class object in the IdentityIQ database so that they can be associated with other objects with that value, for example a description or an owner. Any attribute can become managed: department, location, title, but the most common attribute to be managed is the one holding group memberships. Managed attributes can be viewed and managed from the Entitlement Catalog page.
Entitlement	Specify attributes to be used as entitlements on this application. Attributes specified as entitlements are used by IdentityIQ as follows: — as additional entitlements during certification. — when creating profiles based on exiting users on this application. Profiles are created on the IdentityIQ Modeler and are used to create roles. — in account group certifications. — in Lifecycle Manager.

Table 41—Application Configuration - Schema Tab Field Descriptions

Fields	Descriptions
Multi-Valued	Specify attributes for which multiple values might be returned during aggregation. Attributes flagged as multi-valued are stored as a list. Even objects that have a single value for a multi-value attribute are stored as a single-item list. Multi-valued attributes are used for queries throughout the product. Before multi-valued attributes are available for use in searches, they must be mapped on the Edit Account Attribute page. "How to Add or Edit Account Attributes" on page 34
Correlation Key	Specify attributes that IdentityIQ can use to correlate activity discovered in the activity logs for this application with information stored in identity cubes. For example, activity logs might contain the full name of users instead of unique account ids. Therefore, correlation between the activity discovered by an activity scan and the identity cube of the user that performed the action must key off of the user's full name. **Note: Correlation Key is only used during activity aggregation. If activity aggregation is not being used, Correlation Key should not be selected.**
Minable	Specify attributes for use during role and profile creation. When creating roles and profiles it is possible to mine applications for attributes and permissions to use in those objects rather than manually entering the values. Only attributes designated as minable are returned by those searches.
Remediation Modifiable	Attributes that are remediation modifiable can have their values and permissions modified from the Certification Report page for the identity being certified. Specify the method of modification for this attribute: **Select** — display a select list of all possible values or permissions for this attribute. **Free text** — display a text field in which a certifier can enter any value.

Provisioning Policies Tab

Provisioning Policies are used to define application object attributes that must be managed due to a Lifecycle Manager request. With a provisioning policy in place, when a role or entitlement is requested the user must input specified criteria into a generated form before the request can be completed. A policy can be attached to an IdentityIQ application object or role and is used as part of the provisioning process.

For applications that support multiple application objects, each object is displayed in a separate table containing the provisioning policies those objects support. Not all application objects support all of the provisioning policies listed below.

In order to be able to provision to a DN with a backslash (\) to an Active Directory application through the Cloud Gateway you will need to set the following properties in catalina.sh or catalina.bat on the Cloud Gateway instance:

```
    set CATALINA_OPTS=%CATALINA_OPTS%
-Dorg.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=true
    set CATALINA_OPTS=%CATALINA_OPTS%
-Dorg.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH=true
```

Setting the dependencies between applications and accounts implies ordering in provisioning.

IdentityIQ includes the following types of provisioning policies:

- Create
- Update
- Delete
- Enable Account
- Disable Account
- Unlock Account
- Change Password
- CreateGroup
- UpdateGroup

Click an existing provisioning policy or click **Add Policy** to create a new one using the Provisioning Policy Editor or to reference an existing policy. Only one of each policy types is supported.

Use the **Application Dependencies** drop-down list to create the list of applications where this application is dependent for provisioning. If no account is detected on an application where this application is dependent, an account request is added to the provisioning plan and the provision policy for this application is processed as expected.

The Provisioning Policy Editor panel contains the following information:

Table 42—Application Configuration - Provisioning Policy Editor Field Descriptions

Field Name	Description
Name	The name of your provisioning policy.
Description	A brief description of the provisioning policy.
Owner	The owner of the provisioning policy. This is determined by selecting from the following: None — no owner is assigned to this provisioning policy. Application Owner — identity assigned as owner of the application in which the provisioning policy resides. Role Owner — identity assigned as owner of the role in which the provisioning policy resides. Rule — use a rule to determine the owner of this provisioning policy. Script — use a script to determine the owner of this provisioning policy
Edit Provisioning Policy Fields Panel Use the Edit Provisioning Policy Fields panel to customize the look and function of the form fields generated from the provisioning policy.	
Name	The name of the field.
Display Name	The name displayed for the field in the form generated by the provisioning policy.
Help Text	The text you wish to appear when hovering the mouse over the help icon.

Table 42—Application Configuration - Provisioning Policy Editor Field Descriptions

Field Name	Description
Type	Select the type of field from the drop-down list. Choose from the following: Boolean — true or false values field Date — calendar date field Integer — only numerical values field Long — similar to integer but is used for large numerical values Identity — specific identity in IdentityIQ field Secret — hidden text field String — text field
Multi Valued	Choose this to have more than one selectable value in this field of the generated form. Click the plus sign to add another value.
Read Only	Determine how the read only value is derived: Value — value based on the selection from the drop-down list Rule — value is based on a specified rule Script — value is determined by the execution of a script
Hidden	Determine how the hidden value is derived: Value — value based on the selection from the drop-down list Rule — value is based on a specified rule Script — value is determined by the execution of a script
Owner	The owner of this provisioning policy field. This is determined by selecting from the following: None — no owner is assigned to this provisioning policy. Application Owner — identity assigned as owner of the application in which the provisioning policy resides. Role Owner — identity assigned as owner of the role in which the provisioning policy resides. Rule — use a rule to determine the owner of this provisioning policy. Script — use a script to determine the owner of this provisioning policy
Required	Choose whether or not to have the completion of this field a requirement for submitting the form.
Review Required	Choose whether or not to require the person who is approving the workflow item to approve this field.
Refresh Form on Change	Select this option to have the form associated with this policy refresh to reflex changes to this policy.
Display Only	Set this field as display only.
Authoritative	Boolean that specifies whether the field value should completely replace the current value rather than be merged with it; applicable only for multi-valued attributes
Value	Determine how the value is derived. Select from the following: Literal — value is based on the information you provide Rule — value is based on a specified rule Script — value is determined by the execution of a script
Validation	Gives the ability to specify a script or rule for validating the user's value. For example, a script that validates that a password is 8 characters or longer.

Correlation Tab

Use the correlation tab to configure how application accounts are assigned to identities within IdentityIQ using account and identity information.

To configure Account Correlation you can select an existing correlation configuration from the list or create a new configuration using the correlation wizard. The correlation wizard walks you through both attribute and condition based correlation.

In the manager correlation section, configure how assigned managers should be resolved to identities using existing information.

- **Attribute Based Correlation** — use attributes of the application's account to find identities based on attribute values stored on Identity objects. This is how accounts are typically correlated to Identities. For example, you can correlate the application's account attribute “mail” with an identity's attribute “email”.
- **Condition Based Correlation** — assigns application accounts to existing identities by defining attribute conditions. Service and Administrator accounts might be handled using condition based correlation. For example, the root account on Unix typically does not have any identifying attributes that can help when trying correlate it to an existing identity. In cases where the account owner is known because they are the application owner, a direct mapping can be used.

To configure Manager Correlation you must select two attributes, the Application Attribute and the Identity Attribute.

- **Application Attribute** — the name of the applications account attribute that holds the reference to the manager.
- **Identity Attribute** — the name of the identity attribute to use when searching for managers.

For example, if the application has an attribute `managerEmail` with the value set as the email address of the manager of every user with an account on the application, and you have an identity attribute `email` configured within IdentityIQ with the value set as the email address for every identity cube, you would correlate the application attribute `managerEmail` with the identity attribute `email` to perform manager correlation.

Accounts Tab

The Accounts Tab list the following information for the selected account:

- Account ID
- Account Name
- Status
- Last Refresh
- Identity Name

Click the down -arrow next to an account name to view detail about the account, such as the last login time and date.

Risk Tab

The application Risk tab provides a current application risk score and a detailed view of the raw and compensated risk score for each category used to derive that score. This page also provides a list of the top composite score contributors providing further information on how the score was derived and providing clues on the areas of highest risk. These scores are based on the latest information discovered by IdentityIQ.

IdentityIQ uses a combination of base access risk and compensated scoring to determine the overall application risk scores, or composite risk score, used throughout the application.

All scores are calculated by first determining the percentage of accounts that have the qualities tested by the component score. For example, if 10 out of 100 accounts are flagged as service accounts, then the raw percentage is ten percent (.10). This number is then multiplied by a sensitivity value which can be used to increase or decrease the impact of the original percentage. The default sensitivity value is 5 making the adjusted percentage fifty percent (.50). This final percentage is then applied to the score range of 1000 resulting in a component score of 500.

After the component score is calculated a weight, or compensating factor, is applied to each component score to determine the amount each contributes to the overall risk score for the application. For example, a few violator accounts might increase risk more than many inactive accounts.

Service, Inactive, and Privileged component scores look for links that have a configured attribute. For example, the component `service` with a configured value `true`.

The Dormant Account score looks for a configured attribute that is expected to have a date value, for example `lastLogin`. This algorithm has an argument, `daysTillDormant`, that defaults to thirty (30). If the last login date is more than thirty (30) days prior to the current date, the account is considered dormant and is factored into the risk score.

The Risky Account score looks for links whose owning identity has a composite risk score greater than a configured threshold. The default threshold is five hundred (500).

The Violator Account score looks for links whose owning identity has a number of policy violations greater than a configured threshold. The default threshold is ten (10).

Activity Data Sources Tab

The Activity Data Source tab is used to configure the data sources from which activity information is collected. The information collected from these sources is normalized and then stored by IdentityIQ and used to monitor activity information for users and applications. Activity information is collected and correlated using the Activity Aggregation task. Activity information displayed on the or returned by activity searches is based on the information stored by IdentityIQ since the last aggregation and correlation tasks were run.

The Activity Data Sources table contains the following information:

Table 43—Application Configuration - Activity Data Source Tab Table Descriptions

Column	Description
Name	A descriptive name for the activity data source from which the activity data is collected.
Type	The activity data source type, for example JDBC Collector, Log File, CEF Log File or Windows Event Log Collector.
Modified	The date when the activity data source was last modified.

Right-click a data source and select **Edit** or click **New Activity Data Source** to access the Activity Data Source Configuration page.

Right-click a data source and select **Delete** to remove an activity data source.

See “Activity Data Source Configuration” on page 112.

Unstructured Targets Tab

Unstructured target information is used to define unstructured data sources from which the connector is to extract data. Unstructured data is any data that is stored in a format that is not easily readable by a machine. For example, information contained in an Excel spread sheet, the body of an email, a Microsoft Word document, or an HTML file is considered unstructured data. Unstructured targets pose a number of challenges for IdentityIQ connectors, because not only is the data stored in a format that is hard to extract from, the systems and directory structures in which the files reside are often difficult to access.

The most common unstructured data type supported by IdentityIQ is an operating system's file system permissions.

This target collector requires a the IdentityIQ Service to be installed on a machine that has visibility to the directory or share to include in the target scan. Refer to the IdentityIQ *Installation Guide* for information on installing and registering the IQService.

The unstructured targets defined on this tab are used by the Target Aggregation task to correlate targets with permissions assigned to identities and account groups for use in certifications.

Each of the Target Source Types require unique details or attributes for their configuration, but share some basic information.

The Unstructured Targets tab contains the following basic information for each of the Target Source Types:

Table 44—Application Configuration - Unstructured Targets Tab Field Descriptions

Field	Description
Name	The name of this unstructured target configuration.
Description	A brief description of this configuration.
Rules: Specify the rules used to transform and correlate the targets. **Note: Click the "..." icon to launch the Rule Editor to make changes to your rules if needed.**	
Creation Rule	The rule used to determine how the unstructured data extracted from data source is transformed into data that can be read by IdentityIQ.
Correlation Rule	The rule used to determine how to correlate account information from the application with identity cubes in IdentityIQ.
Provisioning Action	The provisioning action used to pass information to the affected applications.

Rules Tab

These are the rules that can be customized to handle the complexity of the data being extracted. Rules are specific to connectors and are used throughout the product. You can write more than one of each type and select the rule to use from drop-down lists.

A file containing an example of each rule type is included in the IdentityIQ installation package. The examplerules.xml file is located in the IdentityIQ_*HOME*/WEB-INF/config directory.

The rules in this table apply to all applications and are called by the aggregation process. A correlation rule is only required if there is more than one application and a correlation configuration has not been defined.

The delimited file connector has rules that are specific to its implementation; buildMapRule, mergeMapsRule, and mapToResourceObject Rule.

Password Policy Tab

Use the password policy tab to select and create password policies which apply to specified applications.

The password policy panel contains the following:

Table 45—Application Configuration - Password Policy Field Descriptions

Field Name	Description
Name	The name of your password policy.
Description	A brief description of the password policy.

Click an existing password policy to edit it or click **Create New Policy** to configure one from scratch.

Table 46—Application Configuration - Password Policy Editor Field Descriptions

Field Name	Description
Password Policy Name	The name of your password policy.
Password Policy Description	A brief description of the password policy.
Minimum number of characters	Input the minimum number of characters required for a valid password.
Maximum number of characters	Input the maximum number of characters required for a valid password.
Minimum number of letters	Input the minimum number of letters required for a valid password.
Minimum number of character type constraints to meet	The minimum number of character types (digits, upper case, lower case, special) required for a valid password.
Minimum number of digits	Input the maximum number of numerical digits required for a valid password.
Minimum number of uppercase letters	Input the minimum number of uppercase letters required for a valid password.
Minimum number of lowercase letters	Input the minimum number of lowercase letters required for a valid password.
Minimum number of special characters	Input the minimum number of special characters required for a valid password.

Table 46—Application Configuration - Password Policy Editor Field Descriptions

Field Name	Description
Number of repeated characters allowed	The maximum number of consecutive repeated characters allowed in a valid password. For example, if this option is set to 2, “cloudd” and “cclooud” is valid, but “clouddd”, “cloooud” and “cccloud” are invalid. For “cclooudd” invalid password, the following error message is displayed: Password should not contain more than 2 occurrence(s) of the repeated characters. For “Clouddd” invalid password, the following error message is displayed: Password should not contain more than 2 consecutive repeated characters.
	The maximum number of occurrences of repeat characters allowed in a valid password. For example, if this option is set to 1, “happy123” is valid, but “happy123dd” and “happpy123” are not.
Password history length	The number of past passwords that cannot be used again.
Triviality check against old password	Ensure that the shorter of the old and new password is not a substring of the other. Both passwords are changed to upper case prior to the check.
Minimum number of characters by position	The minimum number of unique characters by position the new password. Can be used to ensure that not just the first or last character is being changed. Select **Case sensitive check** to ensure that more than just the case is changing in the new password.
Validate passwords against the password dictionary	Select do disallow the use of any password defined in the password dictionary. The password dictionary is a configurable list of terms unavailable for use as passwords. The `passwordDictionary.xml` file located in IdentityIQ`/WEB-INF/config/`.
Validate passwords against the identity's list of attributes	Select to disallow the use of Identity attribute values as passwords.
Validate password against the account’s display name	Select to disallow the use of the account’s display name as the password (exact match by default). Enter a **Minimum word length** to define the minimum length of a substring of the account’s display name allowed in the password.
Validate password against account ID	Select to disallow the use of the account’s ID as the password (exact match by default). Enter a **Minimum word length** to define the minimum length of a substring of the display name of the account allowed in the password.
Validate passwords against the identity's account attributes	Select to disallow the use of Identity link attribute values as passwords. Enter a **Minimum word length** to define the minimum length of a substring of the account’s ID allowed in the password.

Table 46—Application Configuration - Password Policy Editor Field Descriptions

Field Name	Description
Configure Password Filter	Select a filter that selects the identities to which this password policy applies. Select from the following filters: All — all identities have this password policy applied Match List — only identities whose criteria match that specified in the list. The criteria is configured using the tools provided. Add identity attributes, application attributes and application permissions. Customize further by creating attribute groups to which this password policy applies. **Note: If Is Null is selected, the associated value text box is disabled. When the is null match is processed, the term matches users on the chosen application who have a null value for that attribute/permission.** Filter — use an XML filter or compound filter to determine the identities to which this password policy applies. Script — use a BeanShell script to determine the identities to which this password policy applies. Rule — use a rule to determine the identities to which this password policy applies. Population — select a population to which this password policy applies.

SecurityIQ Type Application

SecurityIQ enables enterprises to discover and govern access to sensitive data and better address the security threat to unstructured data. As a key component of SailPoint's Identity Governance strategy organizations can take a comprehensive approach to govern access across all users, applications and data with enhanced visibility while reducing risk.

Use SecurityIQ to:

- Identify and govern access to exposed sensitive data found within cloud and on-premises file stores.
- Enable business users to manage access to data they know best; alleviating IT burden and over-permissioned access.
- Leverage a comprehensive identity governance solution that extends to unstructured data throughout the enterprise.

Create a new application of type SecurityIQ. By default, this application will start with an alert schema, as well as unstructured and associations schemas. The unstructured and associations schemas are used to define the makeup the Target and TargetAssociation respectively.

Attributes/Configuration:

Defined schemas on the configuration tab. If an alert schema is defined, this will include the configuration needed to set up the Alerts. If the Unstructured schema is defined, this will include the configuration needed to set up the Target/Target Permissions.

Connection Settings:

Database URL - The jdbc connection URL for the SecurityIQ database.

Driver Class - The JDBC Driver class to use for the connection.

UserName - The database user name

Password - Password for the configured user name

Schema - Schema used for the SIQ DB.

General Settings:

Referenced Applications - This is a list of applications to which the given permission are correlated. The target permissions are correlated to either a Link or ManagedAttribute belonging to one of the applications in the list.

Aggregate Inherited - True to aggregate inherited permissions. If set to true, the dataset will be much larger. If false, only the top level permissions are aggregated, and inheritance is assumed as defined on the native source.

TargetHosts - The List of SIQ Business Application Monitors from which to aggregate permissions.

Target Host Paths - This is a CSV of Paths from which to aggregate. This aggregation starts at the given root paths, and discovers all permissions under these paths. If not specified, all target/target permissions for the specified BAM are aggregated.

Rules:

The rules tab within the Application Definition user interface enables the defining of rules for given object types. The Application level rules and schema level rules, for schemas that allow them, are shown with the ability to select/edit (based off of correct capabilities) the given rules. The unstructured schema support Correlation/Creation/Customization/Refresh rules on the schema level.

Creation rules for unstructured schema will be of **Rule Type** TargetCreation

Refresh rules for unstructured schema will be of **Rule Type** TargetRefresh

Correlation rules for unstructured schema will be of **Rule Type** TargetCorrelation

Customization rules for unstructured schema will be of **Rule Type** ResourceObjectCustomization

The unstructured/assocations schema AttributeDefinitions are used to define the columns to include in the query.

Application Re-configuration

The application re-configuration option enables you to change the application type without losing history associated with the application or having to create a new application. For example, if you first deployed your instance of IdentityIQ using a flat file connection, but now want to use some of the more advance features, such as provisioning. The type defines the way in which IdentityIQ connects to the application.

Application types that have the same value format for identity and group attributes in the original and re-configure target are best suited for re-configuration.

The following application types can be re-configured:

Note: Even in the following scenarios, there might be some connectors that do not re-configure correctly. See "Application Re-configuration Considerations" on page 111.

- Delimited file to the corresponding direct connector (Delimited File to Active Directory - Direct)
- JDBC connector to corresponding direct connector (JDBC to Oracle Applications - Direct)
- Agent based connector to direct connector (Active Directory - Full to Active Directory - Direct)
- To a rewritten connector for better performance or more functional support (Google Apps - Direct)

Application Re-configuration Considerations

Take the following points into consideration before deciding to re-configure an application.

- Do the identity attribute of account and group in the original application match the identity attribute of account and group in the re-configured application?

For example:

  - Two application types (Delimited File and Active Directory – Direct) use distinguishedName as the identity attribute of account, and use the same identity attribute for group. Since both of these applications refer to the same identity attribute of both account and group, they would be good candidates for re-configuration.
  - Two application types (Oracle Application – FULL and Oracle Application – Direct) use different identity attributes for account, USER_ID in one and USER_NAME in the other and, USER_ID and USER_NAME differ in format. These are not good candidates for re-configuration.

- If there are special attributes (native identity, managed attribute, entitlement) that split into multiple attributes in the new application type, re-configuration is not recommended.
  - Profiles in SAP –Full refer to both profiles and groups in the managed system, where as in SAP-Direct, profile refers to profiles and group refers to the groups in the managed system. These are not good candidates for re-configuration.

Before Application Re-configuration

Perform the following actions before you begin the re-configuration process:

- Backup the application xml and application type specific customizations such as rules and business processes.
- Plan the attribute mapping of the original and new applications for accounts and group schema. If there are attributes in the original application type that are not in the re-configured application type, you might lose some configuration and historical data.
- Check the provisioning policies of the target application and decide which policy to use, the policy from the original application type or the policy from the re-configured application type.

How to Re-configure an Application

Note: While re-configuring an application the target application must have a static schema and not a dynamic schema like, for example, JDBC or DelimitedFile connectors. There is the button named Discover Schemas to generate the schema.

1. Go to **Applications -> Application Definition** and select an application.
2. On the Application Configuration page, click the **Reconfigure** button to display the Select New Application Type dialog.

3. Select an application type from the **New Application Type** drop-down list and click **Save**.
4. Confirm your selection to go to the Application Configuration page in edit mode.
   The tabs that contain information requiring attention are marked with an red asterisk.
5. Go to the Attributes tab and enter the valid configuration attribute settings and test the connection.
6. Go to the Schema Mapping tab and map the Previous Schema Attributes to the New Application Type Schema Attributes for the Account and Group.
   Use the **Add Missing Attributes** and **Keep Extra Attributes** options to select to add missing attributes from the old (original) application type to the new application type, and to keep attributes that are on the new application type but were not on the original application type.
7. Go to the Provisioning Policy tab and select the provisioning policy to use for the re-configured application. It is recommended that you use the policy that corresponds to the application type of the newly re-configured application. You can use a different policy, but you must manually edit that policy to match the changes made during the re-configuration process.
8. Save the re-configured application.

After Application Re-configuration

Check the re-configured application for the following when the process is complete:

- Attributes that were not mapped might become not work and the values not get populated.
  - Unmapped attributes affect configurations, for example, policy or business roles, and context based historical data, for example viewing certification history, that is based on a population that relies on the attribute.
  - Related populations might not be populated with identities.
  - Pending provisioning operations that contain that attribute might fail.
  - Verify other place that use the attribute, such as identity mapping, account mapping, roles, policies, and policy violations.
- Verify the application definition for unwanted entries like build map rules or provisioning rules still exist.
- Perform account and account group aggregation.
- Perform refresh identity cube.
- Perform prune identity cube.

Activity Data Source Configuration

Use the Activity Data Source Configuration page to add or edit activity data sources. Activity collectors access activity data sources such as event or audit logs, collect the activity information that is to be monitored, and transform that data into a format that can be read by IdentityIQ. These Activity Data Sources are use for all activity aggregation and reporting.

Changes made on this page are not committed until a save is performed on the application with which they are associated. For example, if you add or delete a data source on this page and click **Save**, you do not see that change reflected on the application until you click **Save** on the application page and commit the change.

For each activity data source enter or edit the following:

- The general data source information in Table 47, “Configure Application Page Field Descriptions.
- Activity target information found on the Activity Target tab for each source type, see “Activity Targets” on page 113.
- The unique connection and query setting for each activity data source type.
  - “JDBC Collector Settings” on page 114
  - “Windows Event Log Collector Settings” on page 114
  - “Log File Collector Settings” on page 115
  - “RACF Audit Log Collector” on page 116
  - “CEF Log File” on page 117

Table 47—Configure Application Page Field Descriptions

Field	Description
Name	A short, descriptive name for the activity data source.
Description	A brief description of the activity data source.
Transformation Rule	The transformation rule required to convert the data collected from the data source into a format that can be used by IdentityIQ. **Note: Click the “...” icon to launch the Rule Editor to make changes to your rules if needed.**
Correlation Rule	The correlation rule that should be used to correlate the activity data collected with identities. **Note: Click the “...” icon to launch the Rule Editor to make changes to your rules if needed.**
Activity Data Source Type	The type of data source from which the activity is being collected. The **Activity Data Source Type** drop-down list contains the types of data source from which activity information can be collected. This list will grow and change to meet the needs of IdentityIQ users. **Note: When “CEF Log File” is selected from the drop down list, the “Transformation Rule” and “Correlation Rule” fields are displayed with the following respective values:** **- Transformation Rule: CEFTransformRule** **- Correlation Rule: CEFActivityCorrelation**

Activity Targets

The Activity Targets tab is used to specify targets within this data source for use in activity searches. A target is a specific object within a data source that is acted upon. For example, a target might be a machine name for a login action, or a file name for a create action.

The targets specified here are used to populate lists on the Activity Search page. These targets can be grouped with targets specified on other applications to create categories of targets. For example, if you have inventory applications at three different locations and a procurement database on each, you can set each procurement database as a target, create a Procurement category, and then collect activity for all three procurement databases using a single activity search.

See “Activity Targets” on page 113.

On the Activity Targets tab you can add activity targets for the data source with which you are working. Type the name of the activity target in the field at the bottom of the list and click **Add Activity Target**. To remove activity targets, use the selection boxes on the left of the table and click **Delete**.

JDBC Collector Settings

JDBC Connection Settings

IdentityIQ uses the connection settings to access the activity data source.

Table 48—Application Configuration - JDBC Collector Connection Settings

Field	Description
Connection User	A valid JDBC user with access to the data source being accessed by this collector.
Connection Password	The password associated with the Connection User if a password is required. The password is encrypted and is not displayed with the activity data source information.
Database URL	The full url to the activity data source. For example, `jdbc:mysql://localhost/db`
JDBC Driver	The driver class of the activity data source. For example, `com.mysql.jdbc.Driver`

Query Settings

The query settings are used to control the activity information that is collected when an Activity Aggregation task is run.

Table 49—Application Configuration - JDBC Collector Query Settings

Field	Description
SQL Statement	The SQL statement used to query activity from the database.
Condition Builder	Transforms the data mapped in the rule selected as the Position Builder into a SQL statement used by subsequent queries to determine start position.
Position Builder	Rule that converts the last row in the result set returned by the query into a configuration map that is persisted into the IdentityIQ database. The data that is mapped in this rule is used by the condition builder to create a SQL statement used in future queries to determine the start location. This enables IdentityIQ to perform scheduled activity aggregations without having to scan entire data sets with each subsequent aggregation.

Windows Event Log Collector Settings

Note: Before you can use the Windows Event Log Collector, the IQService must be installed and registered. Refer to your Installation Guide for information on installing and registering the IQService.

Event Log Settings

IdentityIQ uses the connection settings to access the activity data source and the query settings to control the activity information that is collected when an Activity Aggregation task is run.

Table 50—Application Configuration - Windows Event Log Collector Settings

Field	Description
User	Valid Windows user name with access to the event log containing the activity data.
Password	The password associated with the user specified.
IQ Service Host	The host name where the IQ service is running.
IQ Service Port	The listening port of the IQ service.
Event Log Server	The server where the activity data source resides.
Query String	The MQL query use to specify the activity data to collect during the activity aggregation.
Block Size	The number of events to retrieve with each activity aggregation performed on this activity data source.

Log File Collector Settings

Transport Settings

The transportation settings are used to access the server where the log file containing the activity data resides.

Table 51—Application Configuration - Log File Collector Transportation Settings

Field	Description
Transport Type — depending on the transport type selected you will see the following:	
local	If the log file containing the activity data is on the same server as IdentityIQ, no further connection-type information is required.
ftp	FTP User — a valid user name with authentication access to the FTP host. FTP Password — the password associated with the FTP user. FTP Host — the host where the log file resides.
scp	SCP User —a valid user name with authentication access to the SCP host. SCP Password — the password associated with the SCP user. SCP Host — the host where the log file resides. SCP Private Key — the private key that is used to encrypt the collected data.

Log File Settings

The log file settings are used to define the query used to collect the activity data.

Table 52—Application Configuration - Log File Collector Log File Settings

Field	Description
File Name	The name of the log file containing the activity data.

Table 52—Application Configuration - Log File Collector Log File Settings

Field	Description
Lines to Skip	The number of lines to skip before starting the scan for activity information.
Filter Nulls	Skip lines that don't conform to the defined format.
Multi-lined Data	A single record in this file spans multiple rows.
Regular Expression	A regular expression groups that can be used to tokenize each record in the file.

Log Fields

The log field settings are used to create the log fields based on the column headings in the log file.

Table 53—Application Configuration - Log File Collector Log Fields

Field	Description
Name	The name of the log field to create based on a column name from the log file.
Trim Value	Remove white space around the column name before creating the log field.
Drop Nulls	If the column by this name is null, ignore this record. For example, if the user field is null, then the record cannot be correlated to a IdentityIQ identity and, therefore, cannot be used by IdentityIQ.

RACF Audit Log Collector

Transport Settings

The transportation settings are used to access the server where the log file containing the activity data resides.

Table 54—Application Configuration - RACF Audit Log Collector Transportation Settings

Field	Description
Transport Type — depending on the transport type selected you will see the following:	
local	If the log file containing the activity data is on the same server as IdentityIQ, no further connection-type information is required.
ftp	FTP User — a valid user name with authentication access to the FTP host. FTP Password — the password associated with the FTP user. FTP Host — the host where the log file resides.
scp	SCP User —a valid user name with authentication access to the SCP host. SCP Password — the password associated with the SCP user. SCP Host — the host where the log file resides. SCP Private Key — the private key that is used to encrypt the collected data.

Log File Settings

The log file settings are used to define the query used to collect the activity data.

Table 55—Application Configuration - RACF Audit Log Collector Log File Settings

Field	Description
File Name	The name of the log file containing the activity data.
Lines to Skip	The number of lines to skip before starting the scan for activity information.
Filter Nulls	Skip lines that don't conform to the defined format.

CEF Log File

Transport Settings

The transportation settings are used to access the server where the log file containing the activity data resides.

Table 56—Application Configuration - CEF Log File Transport Settings

Field	Description
Transport Type — depending on the transport type selected you will see the following:	
local	If the CEF log file containing the activity data is on the same server as IdentityIQ, no further connection-type information is required.
ftp	FTP User — a valid user name with authentication access to the FTP host. FTP Password — the password associated with the FTP user. FTP Host — the host where the log file resides.
scp	SCP User —a valid user name with authentication access to the SCP host. SCP Password — the password associated with the SCP user. SCP Host — the host where the log file resides. SCP Private Key — the private key that is used to encrypt the collected data.

Log File Settings

The log file settings are used to define the query used to collect the activity data.

Table 57—Application Configuration - CEF Log File Settings

Field	Description
File Name	The name of the CEF log file containing the activity data.
Lines to Skip	The number of lines to skip before starting the scan for activity information.
Filter Nulls	Skip lines that do not conform to the defined format.
Multi-lined Data	A single record in this file spans multiple rows.
Regular Expression	A regular expression groups that can be used to tokenize each record in the file. The format of CEF Log File. For example, (\w\w\w\s\d\d\s\d\d:\d\d:\d\d)\s(.*)CEF:(.*)|(.*)|(.*)|(.*)|(.*)|(.*)|(.*)|(.*)|(.*)(.*)

Log Fields

The log field settings are used to create the log fields based on the column headings in the log file.

Table 58—Application Configuration - CEF Log Fields

Field	Description
Name	The name of the CEF log field to create based on a column name from the CEF log file.
Trim Value	Remove white space around the column name before creating the CEF log field.
Drop Nulls	If the column by this name is null, ignore this record. For example, if the user field is null, then the record cannot be correlated to a IdentityIQ identity and, therefore, cannot be used by IdentityIQ.

IdentityIQ uses connectors to extract data and transform it into a format it can read. A connector is a Java class that extends the IdentityIQ `AbstractConnector` class and implements the IdentityIQ Connector interface. Connectors provide the means by which IdentityIQ communicates with targeted platforms, applications and systems. Each application type requires different information to create and maintain a connection. For detailed connector information refer to the connector documentation delivered with IdentityIQ.

Native Change Detection Configuration

IdentityIQ can be configured to detect native changes on applications with which it communicates during the aggregation process and launch business processes accordingly. Native changes are, changes made directly to an account on an application that were not processed as part of an IdentityIQ request.

Once enabled, aggregation will start detecting changes (while filtering SailPoint requested items) and storing them with other Life Cycle Events on the identity. If you make native changes you will see them being stored on the Identity object.

To configure IdentityIQ to detect native changes during aggregation, do the following:

Note: For Native Change Detection to operate you must have both a life cycle event defined and the application enabled.

1. Run an aggregation to obtain the baseline information for the application.
2. Configure a Native Change life cycle event on the Life Cycle Events page.
   There are two life cycle change events included with IdentityIQ and you can configure your own as needed:

   Lifecycle Event - Email manager for all native changes:

   Sends a formatted email to the manager describing the changes detected.

   Lifecycle Event - Manager Approval for all native changes:

   Generates an approval work item for each change detected. Any items rejected are undone/reversed and provisioned. This business process also creates an access request within IdentityIQ so that once the changes are made they will be visible from the Access Request page.
3. Go to **Applications -> Application Definition** and select an application.
4. Select Native Change Detection on the Application Configuration page.
5. Define the operations to include when detecting native changes - **Create**, **Modify**, **Delete**.

6. Define the attributes to compare when detecting native changes:
   Entitlements: All entitlement attributes
   User Defined: Manually enter the names of the attributes to compare.
7. Run or schedule aggregations to detect and store any changes.
8. Run an Identity Refresh task with the Process Events option enabled to trigger the life cycle events for any changes detected since the last time the events were processed.

Chapter 6: Define Home Page Quicklinks

Quicklinks are objects in IdentityIQ that enable you to place customized links on the IdentityIQ Home page and in the Quicklinks menu available on every page. Quicklinks are defined when IdentityIQ is deployed and are based on the needs of your enterprise. You can determine the behavior and availability of these links for different users. For example, IdentityIQ can be set up to limit access based on the user capabilities, rights, or workgroup membership.

Three objects control l inks. **QuickLink** objects define the links, the DynamicScope object controls who can view those links, and the **QuickLink**Option object references the first two to create the Quicklinks within the product.

Managing Quicklinks

QuickLinkOption

The **QuickLinkOption** object is created when a quicklink population, or DynamicScope object, is created on the Quicklink Populations page and associated with one or more **QuickLink** objects. The **QuickLinkObjects** objects do not control quicklink populations, nor the targets of the **QuickLink** objects, they are containers holding references to both.

```
<QuickLinkOptions allowSelf="true" created="1443970828183"
id="2c90900950335ce70150335e4797010e">

  <DynamicScopeRef>

    <Reference class="sailpoint.object.DynamicScope"
id="2c90900950335ce70150335e4783010c" name="Everyone"/>

  </DynamicScopeRef>

  <QuickLinkRef>

    <Reference class="sailpoint.object.QuickLink"
id="2c90900950335ce70150335e478a010d" name="Access Reviews"/>

  </QuickLinkRef>

</QuickLinkOptions>
```

DynamicScope

The DynamicScope object define groups of users, quicklink populations, based on capability, rights, indirect capabilities and rights granted by a workgroup, population, or any attribute of the identity. These objects are defined on the Quicklinks Populations page. See “Quicklink Populations” on page 40.

DynamicScope objects are referenced by name or ID in a **QuickLinkOption** object. If the quicklink population applies to an identity, the Quicklink is visible to that identity. Only System Administrators can view Quicklinks with no scopes.

Note: DynamicScope objects are used to define the population of people who can view and run the Quicklink. DynamicScope objects are not the group of identities or objects that the Quicklink interacts with after the link is clicked.

Examples

The product ships with a DynamicScope that represents the **allowAll** option. The name of the DynamicScope is named **Everyone**. You can associate this option with any Quicklinks you want to enable the entire user population to view or use. The following **QuickLinkOptions** reference this DynamicScope by default:

- Access Reviews
- Approvals
- Signoffs
- Work Items
- Policy Violations

```
<DynamicScope allowAll="true" created="1443970828163"
id="2c90900950335ce70150335e4783010c" name="Everyone"/>
```

The following XML example of a DynamicScope restricts visibility to a specified Quicklink. Visibility is enabled for users in the IT department or who have the Help Desk Personnel capability. Visibility is also enabled for identities in the Inclusion list. Because Barbara.Wilson is in the Inclusions list, she can always see the Quicklink regardless of her capabilities or department.

```
<DynamicScope created="1443973952475" id="2c90900950335ed60150338df3db000a"
name="MyDynamicScope">
  <Description></Description>
  <Inclusions>
    <Reference class="sailpoint.object.Identity"
id="2c90900950336e720150336f0797010d" name="Barbara.Wilson"/>
  </Inclusions>
  <PopulationRequestAuthority allowAll="true"/>
  <Selector>
    <IdentitySelector>
      <MatchExpression>
        <MatchTerm name="capabilities" value="Help Desk Personel"/>
        <MatchTerm name="Department" value="IT"/>
      </MatchExpression>
    </IdentitySelector>
  </Selector>
</DynamicScope>
```

By default, IdentityIQ assumes that any link defined as a top-level **QuickLink** object is for a non-Lifecycle Manager action which does not operate on a target identity, so no user selection options are presented.

Chapter 7: Configure Activity Settings

Configure the activity settings to focus activity tracking and monitoring within your enterprise. When you properly configuring your activity settings, you can narrow the focus of activity searches to obtain more specific and meaningful information to use for identifying and managing risk.

Activity Target Categories

The Activity Target Categories page displays a list of all of the categories that were defined for use with the Activity Search page. Activity Target Categories are groups of targets from one or more applications. For example, if you have inventory applications at three different locations and a procurement database on each, you can set each procurement database as a target, create a Procurement category, and then collect activity for all three procurement databases using a single activity search.

Note: Activity Data Sources and Activity Targets are defined when applications are configured to work with IdentityIQ. If no activity data source and targets were defined, you cannot create Activity Target Categories.

Use the Activity Target Categories page to add or edit activity target categories.

Click an existing category or click **New Category** to open the Add Targets to Activity Category page and create or edit a category.

To delete an active target category from the list, right-click the category and select **Delete**.

Add Targets to Activity Category

The Add Targets to Activity Category page contains a list of the targets included in the selected category and a selection box from which you can choose any target defined within IdentityIQ. Use this page to add or remove targets from the selected activity target category. The categories defined on this page are used as search criteria on the Activity Search page.

Note: Activity Data Sources and Activity Targets are defined when applications are configured to work with IdentityIQ. If no activity data sources and targets were defined, you cannot create Activity Target Categories.

To remove targets from the category, use the selection boxes on the left side of the list and click **Remove Targets**.

Click **Cancel** at any time to return to the Activity Target Categories page without saving your changes.

Add Targets on an Activity Category

To add targets to a category displayed:

1. If you are creating a new category, type a name in the **Category Name** field.
2. Select the application associated with the targets being added from the **Application** drop-down list.
   The **Application** drop-down list contains all of the applications that have at least one activity data source defined.
   After you select an application, the **Activity Data Source** drop-down list is displayed.

3. Select the activity data source that contains the targets you want to add from the **Activity Data Source** drop-down list.
   After an activity data source is selected, the **Targets** list displays.
4. Select targets from the **Targets** list.Use the **Ctrl** and **Shift** keys to select multiple targets.
5. Click **Add Targets** to add the selected targets to the **Targets For ... Category** list.
   To add targets from multiple applications or data sources repeat steps 2 through 5.
6. Click **Save** to save your changes and return to the Activity Target Categories page.

Chapter 8: IdentityIQ Email Templates

Many events in IdentityIQ generate email notifications to notify users of actions required by them or actions taken that directly affects them. These email messages are created based on email templates. Basic templates are provided with the product to construct messages corresponding to each of the email-generating events, and these messages can be customized to meet your specific needs.

To customize any of these templates, copy and rename the template using a unique name.

1. Copy and rename the template using a unique name.
2. Associate the customized template to the email-generating event through the IdentityIQ user interface or configuration XML. See “Importing Email Templates into IdentityIQ” on page 125.

Note: The default email templates should not be modified directly because they might be overwritten during the IdentityIQ release upgrade process.

Accessing the Templates

The default email templates that ship with the product are located in the following area:

- Directory — iiq_*installation_directory*/WEB-INF/config directory
  where iiq_*installation_directory* is the location where you expanded the IdentityIQ installation media
- Files — `emailtemplates.xml` and `lcmemailtemplates.xml`

A third file, named `emailtemplatesSample.xml`, contains additional example templates that are not loaded into IdentityIQ during the initial load process. These templates describe how to create HTML email messages. Any of the templates in these files can be cloned to create custom templates.

After the email templates are loaded into IdentityIQ, either during the initial load or using the console or user interface import option, the templates are stored as XML objects. The templates can be viewed or modified through the IdentityIQ Debug pages. The default templates should not be modified from here because the template are overwritten during any IdentityIQ version update. However, customized templates can be edited directly through the Debug pages, if the organization's source code control procedures allows.

To view the list of email templates from the Debug pages, select **EmailTemplate** from the object list and click **List**. Select the desired template to view or edit its XML representation.

Importing Email Templates into IdentityIQ

When email templates are edited outside of IdentityIQ, they must be imported into the system before they can be used for any notifications. Email templates can be imported through the IdentityIQ user interface or console.

To import a template through the user interface, navigate to the **Gear** icon **-> Global Settings ->** IdentityIQ **Configuration -> Import from File**. Click **Browse** to select the email template's XML file from the file system and click **Import**. IdentityIQ parses the XML during the import process and recognizes the file's contents as an EmailTemplate object. The import fails if the XML is invalid.

To import the email template through the IdentityIQ console:

1. Navigate to the IdentityIQ *Installation Directory*/WEB-INF/bin directory.
2. Start the console and use the console import command to import the file.
   The import is successful only if the XML is valid. Any errors encountered are reported to the console.

   Note: Import files can contain one or more EmailTemplate objects. However, if a file contains more than one object, the import methods expect the set of objects to be wrapped in a <sailpoint></sailpoint> block.

Associating Templates with Events

Email templates are associated to their respective email-generating events in several places in the IdentityIQ user interface and configuration XML. The table below shows the notification type, the default template name, and their association location. To use a custom template for any of these notifications, specify the custom template name in place of the default template in that notification configuration.

Note: Different email templates accept and use different arguments. The selection lists in the user interface for each notification lists all email templates, no matter what arguments they require. Because IdentityIQ provides a fixed set of arguments for each notification type, only templates whose arguments list matches the provided arguments work correctly to create a useful event notification. Refer to the default template XML to see the arguments (names and variable types) for each notification, and ensure that the selected template's argument list matches the default template's arguments.

Table 59—Email Template Associations

Notification Type (Field Label or Configuration Key)	Default Email Template	Configuration Location
For reminder notices	Work Item Reminder	**Gear icon -> Global Settings ->** IdentityIQ **Configuration -> Mail Settings -> Email Templates section**
For escalation notices	Work Item Escalation	
For work item comment notices	Work Item Comment	
For work item forwarding notices	Work Item Forward	
For policy violation notices	Policy Violation	
For task and report signoff notices	Task Result Signoff	
For work item assignment notices	Work Item Assignment	
For work item assignment removal notices	Work Item Assignment Removal	
For remediation item assignment notices	Remediation Item Assignment	
For remediation item assignment removal notices	Remediation Item Assignment Removal	

Table 59—Email Template Associations

Notification Type (Field Label or Configuration Key)	Default Email Template	Configuration Location
For task status email notice	Task status	**Gear icon -> Global Settings ->** IdentityIQ **Configuration -> Mail Settings -> Email Templates section**
Initial Notification Email Template	Certification	**Gear icon -> Global Settings ->** IdentityIQ **Configuration -> Mail Settings -> Email Templates section** **NOTE: When the challenge period is enabled, Challenge-related notification email templates can be overwritten for individual certifications on the Lifecycle page in each certification configuration.** **NOTE: Initial Notification and Bulk Reassignment notices can be overwritten for each certification on the Notifications page in each certification configuration.**
Exceptions Expiration Notices	Mitigation Expiration	
Bulk Reassignment Modification notices	Bulk Reassignment	
Challenge Period Start Notices to Challengers	Challenge Period Start	
Challenge Period End Notices to Certifiers	Challenge Period End	
Challenge Creation Notices to Challengers	Challenge Creation Notification	
Challenged Decision Notices to Certifiers	Certification Decision Challenged Notification	
Challenge Expiration Notices to Challengers	Challenge Expiration	
Challenge Decision Expiration Notices to Challengers and Certifiers	Challenge Decision Expiration	
Challenge Accepted Notices to Challengers	Challenge Accepted	
Challenge Rejected Notices to Challengers	Challenge Rejected	
Sign-off Approval Notices to Approvers	Certification Sign-off Approval	
Reminder Email Template	Work Item Reminder	This is set in the **Notifications** step in each certification configuration when the associated reminder or escalation option is enabled.
Escalation Email Template	Work Item Escalation	
(Revocation) Reminder Email Template	Work Item Reminder	
(Revocation) Escalation Email Template	Work Item Escalation	
Report signoff initial notification	Task Result Signoff	Configured in report specification i**Require Signoff** is selected when the report is defined.
Report signoff reminder notice	No default in UI but argument list matches Work Item Reminder	
Report signoff escalation notice	No default in UI but argument list matches Work Item Escalation	

Table 59—Email Template Associations

Notification Type (Field Label or Configuration Key)	Default Email Template	Configuration Location
Send PDF of report to someone	Default Report Template	Not specified in the user interface. When a report is defined, its XML can be edited to add an emailTemplateId argument to change the email message template. **NOTE: Because this is cumbersome, organizations might choose to edit the Default Report Template directly rather than cloning it. This customization must be reapplied after any** IdentityIQ **version upgrade, as it is overwritten during the upgrade process.**

Table 59—Email Template Associations

Notification Type (Field Label or Configuration Key)	Default Email Template	Configuration Location
Various: Process Variables Step Arguments Approval Work Item Configuration Email Notification Template Workflows (including sub-processes) using these templates: Identity Correlation Do Manual Actions Do Provisioning Forms Assimilate Provisioning Form Provisioning Approval Subprocess Identity Request Notify Identity Request Provision LCM Create and Update LCM Manage Passwords LCM Provisioning Lifecycle Event - Leaver Lifecycle Event Reinstate Role Modeler - Impact Analysis Role Modeler - Owner Approval	LCM Requester Notification LCM Manager Notification LCM User Notification LCM Identity Update Approval LCM Pending Manual Changes LCM Password Change Notification Pending Manual Changes Account Selection Notification Provisioning Form Notification Role Modeler - Approval Role Modeler - Impact Analysis Review	Specified within workflow definitions in **Setup -> Business Processes** as a process variable, step argument, or work item configuration email notification template. **NOTE: For templates referenced from a workflow, all workflow variables, and all steps arguments and approval arguments defined for the step that invokes the email template are also available for use in the email message (subject, body, cc, etc.)**

Table 59—Email Template Associations

Notification Type (Field Label or Configuration Key)	Default Email Template	Configuration Location
key=delegationEmailTemplate	Delegation	Can not be configured through the user interface. Can only be edited through System Config XML From IdentityIQ Debug Pages, click the **Gear icon -> Global Settings ->** IdentityIQ **Configuration** and search the XML for these email template names or key values
key=delegationRevocationEmailTemplate	Delegation Revocation	
key=delegationFinishedEmailTemplate	Delegation Finished	
key=remediationEmailTemplate	Remediation Work Item	
key=remediationNotificationEmailTemplate	Remediation Notification Sent when Notify Users of Revocations is selected in a certification configuration.	
key= certificationReminderEmailTemplate	Certification Reminder Sent on demand from certification.	
key= policyViolationDelegationEmailTemplate	Policy Violation Delegation	
key= AccountGroupPermissions. challengeGenerationEmailTemplate	Account Group Challenge Creation Notification Specialized form of the Challenge Creation Notification email	
key= accessRequestReminderEmailTemplate	Access Request Reminder Sent on demand from the Access Requests page	
key=openCertsEmailTemplate	Open Certifications	

Email Template XML

The Email Template XML consists of an <EmailTemplate> element with a set of attributes and nested elements that specify the basic components of an email message, such as sender, subject, message body, etc.

The next two sections describe those attributes and nested elements.

EmailTemplate Attributes

The following table lists the components that are generally expressed as attributes on the Email Template.

Example:

```
<EmailTemplate name="Work Item Reminder" cc="$identity.Manager.email" >
<From>administrator@XYZCorp.com</From>
<Body>
```

. . . .

Table 60—Email Template Attributes

EmailTemplate Attribute	Purpose
name	Short but descriptive name for template that uniquely identifies email template
cc, bcc	Carbon Copy and Blind Carbon Copy recipients for the email **NOTE: The to attribute is not specified in the email template because it is determined programmatically as the email is sent and would be overridden**
from	Sender email address. If this entry is not specified in template, the default sender specified on the **Global Settings ->** IdentityIQ **Configuration -> Mail Settings -> Default From Address** is used.

EmailTemplate Nested Elements

The components listed in the following table are generally expressed as nested elements due to their complexity and length.

Table 61—Email Template Nested Attributes

Nested Element	Purpose
<subject>	Subject line for the email message
<body>	Body, or main content, of the email message
<signature>	Hashmap of arguments to the email template The signature for each template cannot be changed through the XML. Arguments to each template vary based on the associated system activity to which they apply. Properties and methods belonging to any object passed as an argument are available to include in the message, but other objects that are not part of the template signature cannot be retrieved to use in the email message.
<Inputs>	Nested element within Signature, signifying the input arguments to the template
<Argument>	Nested element within Signature and Inputs. This element names and specifies the type of each input argument to the template
<Description>	Indicates descriptive information for the reader of the XML. Describes the element in which it is nested For example: <Description> within <Argument> describes the argument usage. <Description> within the <EmailTemplate> describes the purpose and usage of the template)

At the most basic level, the contents of these elements and attributes can be written as straight text values with no variable substitutions. However, the real flexibility and usefulness of these templates is found when custom text is substituted into the message body, subject, and other attributes. This substitution is managed by the Apache Velocity Engine.

Apache Velocity Engine

IdentityIQ email templates are processed through an open-source engine called Apache Velocity. Velocity is a Java-based template engine that allows web page designers to reference methods defined in Java code. IdentityIQ email templates make use of the Velocity Template Language to dynamically specify the email messages' contents and generate custom email messages specific to the recipient, work item, and action involved.

The Velocity Template Language (VTL) is a fairly simple to use. Highlights are included below, and full documentation on the syntax is available in the Apache Velocity User Guide or Reference Guide.

References

As IdentityIQ prepares to send an email notification, the appropriate email template is loaded and its argument variables are passed into the VelocityContext where they can be accessed through VTL reference syntax. The contents of different variable types can be accessed through the syntax described in the table below.

Table 62—Velocity Reference Context Syntax

Reference Type	Examples	Additional Information
Variables	$identityName${identityName} $!identityName	These three syntaxes are generally interchangeable in VTL. Shorthand notation (the first example) is the most commonly used, but each of the other two is required in special cases. Refer to the Velocity User Guide for more information.
Hash table values	$customer.Address	Returns the value corresponding to the Address key in a customer hash table
Object properties	$identity.DisplayName	Invokes the getDisplayName() method on the identity object **NOTE: Property notation resolves to the getter method corresponding to the property name, not to an instance variable.Nested object properties can also be retrieved with this notation.** **Example:** **$item.Certification.Name invokes the getName() method on the Certification object retrieved through the getCertification() method on the item object**
Object methods	$identity.getBundles($application)$identity.hasRole($role,'true')	Used for all non-getter methods and for any methods that require arguments

Directives (Commands)

These are the key commands of the Velocity Template Language that are most frequently used in IdentityIQ email templates to dynamically determine the text that is printed in each email message.

Table 63—Velocity Template Language Directives

Command/Directive	Usage/Purpose	Example
#If… #elseif… #else… #end	Conditional evaluation	#if($requester) requested by $requester.displayableName. #{end}
#foreach… #end	Loop through a list of objects	#foreach ($attrReq in $acctReq.attributeRequests) Operation: $attrReq.operation Attribute:$attrReq.name Value(s): $attrReq.value#end
#set	Establish the value of a reference	#set ($identityName = “John.Smith”)#set ($book.Title = “War and Peace”)

Refer to the Velocity User Guide for additional information on the language, including the syntax for less commonly used directives.

VTL vs. $(variableName) Notation

The VTL reference syntax must not be confused with the $(variableName) notation used for variable referencing in other IdentityIQ XML objects, such as Workflows. Velocity does not recognize this syntax and is unable to parse text that uses it. When IdentityIQ detects this syntax in any element of an email template, that portion of the message is not passed to Velocity for rendering at all. Instead, its contents are rendered by a simpler mechanism that is capable of doing the variable substitution based on the template's arguments. However, none of the Velocity directives are interpreted. Any Velocity commands included in the same element with a variable that uses the $(variableName) notation is treated as normal text and printed as-is in the final message.

Incorporating VTL in Email Template XML

All input arguments in the template signature are automatically loaded into the VelocityContext and are therefore accessible through the VTL reference notation for inclusion in the message text. Additionally, Velocity commands (conditional statements, loops, etc.) can be used in determining the text to print in the messages. Excerpts from the default email templates in IdentityIQ are used as examples throughout the rest of this section to illustrate how reference variables and various command syntaxes can be used.

Where to Use VTL

The Velocity Template Language syntax can be specified in any attribute or element that is used to build the email message. Most commonly, this means the <Subject> and <Body> elements of the message, but the cc and bcc recipients (as well as the from email address) are often dynamically specified through reference variables as well.

Reference Variables

When a variable name is referenced within the text for any of the message elements, its value is substituted into the text in its place.

Example:

```
<Body>$certifierName has accepted the challenge for '$challengeItem' and will change
the decision.
</Body>
```

Variable substitution results in the email message content:

John Smith has accepted the challenge for 'Entitlements on Financials' and will change the decision. Velocity can also access data values in fields within objects passed as arguments and replace the variable notation with those values. Consider the <subject> and <body> elements shown below. The argument list for this email template includes a Certification object (named certification) and an Identity object (named certifier).

```
<Subject>$certification.name requires approval</Subject>
<Body>$certification.name was signed by $certifier.displayableName and requires your
approval.
Login and view your work item inbox to complete this request.
</Body>
```

To resolve these variable references, Velocity calls the getName() method on the certification object and the getDisplayableName() method on the certifier's identity object. When the substitutions are made, the final email message looks like this:

> **Subject**: Manager Access Review for Catherine Simmons requires approval
>
> Manager Access Review for Catherine Simmons was signed by Catherine Simmons and requires your approval.
>
> Login and view your work item inbox to complete this request.

Any attribute or method on any of a template's input arguments can be accessed through the reference variables.

Extended attributes on IdentityIQ objects can be accessed through the attributes hash map or by providing the attribute name as an argument to the appropriate getter method. Identity extended attributes, for example, are accessible through the Identity's attributes hash map or through the getAttribute() method on the Identity object (e.g. $certifier.attributes.region and $certifier.getAttribute("region") both return the value in the "region" extended attribute).

Note: The list of available methods for IdentityIQ objects (Identity, Certification, ProvisioningPlan, etc.) can be found in the SailPoint JavaDocs that ship with the IdentityIQ product. These can be viewed through a browser at URL: [IdentityIQ base URL]/doc/javadoc/.

Conditional Statements

Conditional statements can be used to determine whether text should be included in the message or to choose alternate wording based on attribute values.

Whole paragraphs can be included or omitted based on conditional tests.

```
#if ($remindersRemaining > 0)
This work item will escalate after $remindersRemaining more reminder(s).
#end
```

Additionally, parts of a paragraph or sentence can be suppressed or altered based on conditional evaluations. In this example, if $requester is null, the portion of the text “requested by $requester.displayableName, and” is suppressed. Specifying the #if statement in-line with the rest of the text prevents extra line breaks in the middle of the sentence in the resulting email message.

```
<Body>This is your $ordinalNumReminders reminder that the work item $workItemName
#if($requester)requested by $requester.displayableName, and #{end}created on
```

…

Attribute values can also be evaluated to determine which of multiple text selections to include in a message:

```
#if ( $launcher != $identityName )
$launcher requested the following password changes be made to your account(s).
#else
The following password changes were made to your account(s) at your request.
#end
```

Method Calls

Methods within object arguments can be accessed directly through the method reference syntax.

```
#if($expiration)
#if($expiration.getTime() > $nowDate.getTime())
is due on $spTools.formatDate($expiration,3,1).
#{else}
was due on $spTools.formatDate($expiration,3,1).
#{end}
#{else}
was due on $spTools.formatDate($oldDueDate,3,1).
#{end}

#if ( $item.level )
           Priority: $item.level
#else
           Priority: Normal
#end
```

This block checks to see if $expiration is null. If it is not null, it prints “is due on…” or “was due on…” based on whether the expiration date/time is before or after the current date/time. If $expiration is null, this is an older expired work Item so the message uses the $oldDueDate field as work item due date in the message. It also checks to see if the priority was set in. If the $item.level is null, the priority is set to Normal.

Throughout this example, the printed date/time is formatted with the spTools.formatDate() method. The spTools reference variable is discussed in the next section.

Note: This example was altered from its original format in the default Work Item Reminder email template. In the template, this #if statement was specified in-line to prevent unwanted line breaks in the message. Line breaks have been inserted here for readability and should not be included in the message body unless they are desired in the resulting message text.

SPTools Function Library

Immediately before any template is submitted for evaluation by the Velocity engine, the spTools argument is added to the VelocityContext so the template can access its methods. SpTools is a function library that contains a few localization utility methods to help with message formatting -- primarily date formatting. The methods available within spTools are listed in the table below:

Table 64—spTools Methods

Method	Description
String formatDate(Object date)	Formats the passed-in date object to a string representation using the IIQ default date and time styles (both the java.util.dateformat SHORT formats), formatted per the norms of the server's default locale and timezone
String formatDate(Object date, Integer dateStyle, Integer timeStyle)	Formats the passed-in date object to a string representation using the specified date and time styles, formatted per the norms of the server's default locale and timezone **NOTE: The styles are represented by constant values:** **SHORT = 3** **MEDIUM = 2** **LONG = 1** **FULL = 0** **dateStyle and timeStyle correspond to java.text.DateFormat constants. See the Sun Javadocs for details**
String formatDate(Object date, String formatString)	Formats the date according to the specified formatString (uses the java.text.SimpleDateFormat method)
String getMessage(String key)	Returns an internationalized message from the message catalog corresponding to the provided key
String escapeHtml(String string)	Converts HTML special characters to their entity equivalents Example: escapeHtml(' This is an article ') Returns: <div class="article">This is an article</div>

In the out-of-the-box email templates, the most commonly used method from this library is the formatDate() method that takes a date object and two integers as arguments:

```
$spTools.formatDate($expiration,3,1)
```

After the reference shown above is resolved by Velocity, the date/time value in the expiration argument is printed in the email message in MM/dd/yy hh:mm:ssPM format (or the appropriate equivalent for the server's locale).

CDATA Blocks

When any component of the email message (body, subject, cc, etc.) contains characters that are illegal in XML text (e.g. characters like < and & that are interpreted by the parser as the start of an XML element or character entity, respectively), the entire component must be expressed in a CDATA block to prevent it from being parsed. For example, any message body written as HTML must be contained within a CDATA section.

```
<Body><![CDATA[

<html>

<body style="background:#FFF;margin:0;padding:0;text-align:left;">

<p style="margin:20px 0 0;padding:0;color:#333;font:bold 10pt
Arial;line-height:15pt;">$!{workItem.owner.firstname},</p>

<p style="margin:0 0 20px;padding:0;color:#333;font:normal 10pt
Arial;line-height:15pt;">As part of our periodic compliance efforts, you are
responsible for certifying the access your employees have to enterprise applications.
</p>

<p style="margin:0;padding:0;color:#333;font:normal 10pt Arial;line-height:15pt;">A
specific access certification is named <b>$!{workItemName}</b> has been created for
you, and is due on <strong>$spTools.formatDate(
$certification.expiration,3,3)</strong>. <a
href="http://localhost:8080/sailpoint/manage/certification/entityList.jsf?certifica
tionId=$!{certification.id}">Click here to get started on this task.</a></p>

</body>

</html>

    ]]> </Body>
```

The marked up text can then be passed to Velocity for variable substitution and can be rendered as an HTML email message.

Sending an Email from a Rule

Some installations may require notifications to be sent based on events that are not covered by the automated system notifications. Rules can often be used to drive these notifications. The example below shows how to send an email from a rule.

```
<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE sailpoint PUBLIC "sailpoint.dtd" "sailpoint.dtd">

<sailpoint>

<Rule language="beanshell" name="Test Email Sending" type="BuildMap">

  <Description>Debugging Tool - Sends a sample email out via the email
server.</Description>

  <Signature returnType="Map">

    <Inputs>

      <Argument name="log">

        <Description>
```

```
          The log object associated with the SailPointContext.
        </Description>
      </Argument>
      <Argument name="context">
        <Description>
       A sailpoint.api.SailPointContext object that can be used to query the database
if necessary.
        </Description>
      </Argument>
    </Inputs>
   </Signature>
   <Source>
       // Library inclusions for BeanShell
import sailpoint.api.*;
import sailpoint.object.*;
import sailpoint.tools.*;

import java.util.*;
import java.lang.*;
import java.text.*;

// Point this to the "To" email address
String emailDest = "adam.hampton@example.com";

// Specify the email template name in tplName
String tplName = "SailPoint - Test Email Sending";
EmailTemplate template = context.getObjectByName(EmailTemplate.class, tplName);
if (null == template) {
   log.error("ERROR: could not find email template [ " + tplName + "]");
   return;
}
template = (EmailTemplate) template.deepCopy(context);
if (null == template) {
   log.error("ERROR: failed to deepCopy template [ " + tplName + "]");
   return;
}
Map args = new HashMap();
// Add all args needed by the template like this
```

```
args.put("testField1", "This is a test of template parameters.");


EmailOptions ops = new EmailOptions(emailDest, args);
context.sendEmailNotification(template, ops);


return;
 </Source>
 </Rule>
</sailpoint>
```

The beanshell from this example rule can be used as a template for sending an email from any defined rule. Simply change the email template name, recipient, and template arguments to create the desired notification.

Using a Rule to Test Templates and Email Configuration

This example rule can also be used to test the email server configuration or to test any email template. Complete these steps to use the rule for testing purposes:

1. Set up an email server on **Gear icon -> Global Settings ->** IdentityIQ **Configuration -> Mail Settings**.

   Note: To test email templates independently from the email server configuration without actually sending emails through the server, choose Email Notification Type: Redirect to File. This writes the email text to the specified file.

2. Edit an email template to contain the desired message and import it.
3. Edit the “Test Email Sending” rule to include the desired “To” email address, email template, and arguments, and import the rule. (Rules are imported the same way as templates, as described in Importing Email Templates into IdentityIQ.)
4. Run the “Test Email Sending” rule from the IdentityIQ console to send the email.

   ```
   > rule “Test Email Sending”
   ```

5. Examine the resultant email (either in the recipient's inbox or the redirect email file) to verify that the message appears as expected.

Chapter 9: Data Encryption

Data encryptions is done using four basic concepts: the keystore, master password, encrypted data synchronization, and the keystore console.

- KeyStore — the location where the encryption keys used by IdentityIQ are persisted.
- Master Password — the entire keystore can be encrypted with an ascii password. This is the keystore or master password. You can change the keystore password using the keystore console command. Only one master password can exist. When the master password changes the entire keystore and master password file are re-encrypted and rewritten.
- Encryped Data Synchronization — the process of re-encrypting existing data with the newest key in the keystore.
- Keystore Console — the tool (spt keystore) used to manage the keystore and master password.

The keystore and master password are file based and secured by the file system. They are stored in two separate files. The files can be located in the IdentityIQ deployment directory or placed in an alternative directory during configuration. By default the files are stored in the following location:

```
keystorePassword = WEB-INF/classes/iiq.cfg
keystore = WEB-INF/classes/iiq.dat
```

An alternate keystore file location, password file, or just password in clear text can be specified in the spt.`properties` file under these keys:

```
keyStore.file
keyStore.passwordFile
```

spt KeyStore Console Commands

The iiq **keystore** command is the interface to update the keystore and keystore password. A master password can to be entered into the console or generated when it is being updated.

The keyStore console supports the following commands:

Table 1— KeyStore Console Commands

Command	Definition
`use KeyStoreFile masterFile`	Specify the keystore and master file to use when interacting with an alternate keystore. The *keyStoreFile* argument in position 1 specifies the path to the file to be used when creating/updating the keystore. If this argument is not specified the command uses `$SPHOME/WEB-INF/classes/iiq.dat`. The *masterFile* argument in position 2 specifies the path and filename used to store the master file. The **use** command gives you the ability to build the keystores outside your operating running environment and merge them in when scheduled. **Note: If you do not call the use command, the changes are positioned in the configured paths.**
`addKey [ -q ]`	Generate a new encryption key, the key is securely generated and random. `-q` as argument in position 1 generatesa new key without prompting for confirmation. **Note: If no argument is included, you are prompted for confirmation before the key is generated.**
`list`	List the contents of the keystore.
`master [newPassword newPasswordConfirmation]`	**Note: Passwords must be at least 8 characters.** Change the master password and re-encrypt the keystore using the new password. **Note: If no argument is included, you are prompted for confirmation.** If `newPassword` and `newPasswordConfirmation` are in argument position 1 and 2, you are not prompted for confirmation. `-g` is in argument position 1 a new password is generated without confirmation.
`about`	Specifies the two files that being modified.

Encrypted Data Synchronization

The Encrypted Data Synchronization task goes over the objects re-encrypting the values using the newest key.

Note: The Encrypted Data Synchronization task is not enabled upon installation, you must create the task from the New Task drop-down menu.

The task encrypts the following attributes/types by default:

- Application secret configuration attributes
- User passwords
- Password history
- Users challenge questions
- Activity/Target source configurations
- Integration configuration password attributes

In cases such as integration configuration and unstructured target sources the task looks for encrypted values with the password in the name. You can also add a configuration attribute, IIQ`SecretAttributes`, to either type names to define which attributes are targeted during a re-synchronization.

```
<entry key="IIQSecretAttributes">
        <value>
          <List>
            <String>mySecret1</String>
            <String>mySecret2</String>
            <String>password</String>
          </List>
        </value>
      </entry>
```

The task enables you do disable the following three categories of objects:

- Applications — which enabled application, activity and target source updates
- Identity
- Integration configuration

Using IdentityIQ KeyStore

Note: Make sure to store copies of the iiq.dat and iiq.cfg files in a safe place. When you upgrade or reinstall IdentityIQ, the files are readily available to be restored.

Note: Make sure that the file permissions are set to allow access only by the application server that runs IdentityIQ.

In a standard installation of IdentityIQ, passwords are all encrypted using the same encryption secret. Encrypted passwords used in one installation can be reused (decrypted) by any other installation ofIdentityIQ. The keystore feature enables the use of a site specific key. With the keystore feature enabled, a password used on one site cannot be decrypted on another site without having the site specific encryption keys.

Configuration

The keystore is stored in `WEB-INF/classes/iiq.dat` with an accompanying configuration file `WEB-INF/classes/iiq.cfg`.

The `iiq.properties` file provides two options to specify an alternative location for `iiq.dat` and `iiq.cfg`. In the default `iiq.properties`, these options (`keyStore.file` and `keyStore.passwordFile`) are commented out.

```
# IIQ Keystore and Master Password properties
#

# file location of the IIQ keystore
# (override of the default $SPHOME/WEB-INF/classes/iiq.dat )
#
#keyStore.file = /example/path/filename

# file location of the IIQ master password file
# (override of the default $SPHOME/WEB-INF/classes/iiq.cfg )
#
#keyStore.passwordFile = /example/path/filename
```

To put the files in an alternative location, for example `/etc/`access governance suite, enable and change these options as follows.

Note: You may need to modify your application server or Java sandbox security settings to allow access to the key files outside the application server installation directories.

```
# IIQ Keystore and Master Password properties
#

# file location of the IIQ keystore
# (override of the default $SPHOME/WEB-INF/classes/iiq.dat )
#
 keyStore.file = /etc/access governance suite/iiq.dat

# file location of the IIQ master password file
# (override of the default $SPHOME/WEB-INF/classes/iiq.cfg )
#
 keyStore.passwordFile = /etc/access governance suite/iiq.cfg
```

Key Creation

To create or manage the keystore: navigate to the `WEB-INF/bin` folder and start the IdentityIQ KeyStore console with the **keystore** command:

1. Navigate to the `WEB-INF/bin` folder and start the IdentityIQ Keystore console with the keystore command

   ```
   iiq keystore
   ```

2. The console displays a prompt similar to the IdentityIQ console. Use the **help** to list all accepted KeyStore Console commands. For example, use the **addKey** command to create a new key and the **list** command to view the contents of the keystore.

   ```
   > addKey
   Generate a new encryption key (y/n)?
   y
   ```

```
Generating a new encryption key for keystore
[/var/tomcat/webapps/access governance suite/WEB-INF/classes/spt.dat].
New encrpytion key successfully saved to keystore.
All application servers must be restarted for changes to take effect.
>
```

Note: If the keystore file does not exist, it is created and a new, randomly generated key is added.

3. The **list** command displays the newly created key:

```
> list
Listing contents for keystore
[/var/tomcat/webapps/iiq6/WEB-INF/classes/iiq.dat].
KeyAlias   Algorithm Format       Object

2          AES    RAW          javax.crypto.spec.SecretKeySpec@fffe81cd
>
```

4. Use the **exit** command to leave the console.
5. Restart your application server.
   After you restart the application server, any newly set password is encrypted using the new encryption key. Without the files iiq.dat and iiq.cfg, passwords cannot be decrypted by IdentityIQ.
   If you run more than one instance of IdentityIQ, you must place the following files in the WEB-INF/classes folder of each instance, or in the location specified in iiq.properties.

Re-Encrypt Passwords

The new encryption key is used for newly encrypted passwords. However, because existing passwords can also be decrypted using the default method on any system, you must re-encrypt existing passwords. Ro re-encrypt existing password, you must create a new Encrypted Data Synchronization Task in IdentityIQ.

1. From the Navigation menu bar, select **Intelligence -> Tasks**.
2. From the **New Task** drop-down list select **Encrypted Data Synchronization Task** from the drop-down list.
3. Enter a name for the new task.
4. OPTIONAL: If needed, you can exclude types such as applications, identities or integration configurations from processing.
5. **Save and Execute** to immediately run the task.

After the task has completed, all selected encrypted data is changed. A password encrypted with the default key is prefixed with 1. Items encrypted with the new encryption key are prefixed with 2 or another number if multiple encryption keys are stored.

For example, when you look up the Administrator's password in the console, the displays is similar to the following:

```
> search identity password where name admin
2:WpTZ2hmNaInTAJzeK9Swcw==
```

Using the Different Encryption Keys

After a new key is added to the keystore, the key is used as the default encryption key. Everything encrypted inside IdentityIQ then uses the new key. For example:

```
$ ./iiq console
> encrypt test
```

```
2:bt7YJA6iovzF5Uu6RIjueg==
>
```

There is one exception. The command iiq **encrypt**, continues to use the original default encryption key:

```
$ ./iiq encrypt test
1:8zJwAXqvK5/b92JbPXLLKw==
$
```

Although the syntax reported by the bare command does not indicates this, the command accepts an extra parameter to select the encryption key to use. For example:

```
iiq encrypt string [key]
```

Note: The encrypt command in the iiq console does NOT accept this extra parameter.

The *key* is the number that displays in the list command and used as prefix for the keys.

- To select the newly created key, use 2. If multiple keys are in the keystore, use any available higher number.
- To select the original default key use 1 or nothing.

For example:

```
$ ./iiq encrypt test 1
1:8zJwAXqvK5/b92JbPXLLKw==
$ ./iiq encrypt test 2
2:bt7YJA6iovzF5Uu6RIjueg==
```

Provisioning

This section contains the following information:

Chapter 10: Provisioning with IdentityIQ

The IdentityIQ provisioning capabilities help companies manage system access for their personnel. Provisioning requests can be created and processed in several ways in IdentityIQ, based on the needs and configuration of the installation. In many cases, modifications to access or entitlements you request in IdentityIQ can be automatically reflected in the associated native applications.

This chapter traces the flow of the provisioning plan through its evaluation and preparation for processing into the appropriate native system. Included throughout are the IdentityIQ tasks, business processes and rules that operate on the data as it moves through the process.

Note: Business processes are often referred to as workflows.

At a high level, provisioning requests are processed as follows:

- The provisioning request is made through one of several actions or activities.
- The request is created as a provisioning plan.
- The Provisioning Broker evaluates and compiles the provisioning plan, which often involves dividing the original plan into several partitioned plans. Each partitioned plan addresses a single application.
- Each partitioned provisioning plan is passed to the appropriate handler.
    - For integration configuration or read-write connectors, the change is written to the destination system.
    - For Work Items, a work item is created and assigned to an identity who must for manually process the request into the target system.
- The provisioning actions are confirmed and marked on the identity cube, based on the mechanisms involved.

Use the Administrator Console link, under the gear icon, to access the Provisioning Transactions table to view the status of all provisioning transactions in your implementation of IdentityIQ; connectors, manual work items, and IdentityIQ operations. “Using the Administrator Console” on page 75.

Access to the Provisioning Transaction table is controlled with IdentityIQ rights.

This chapter has the following sections:

Recording Provisioning Requests

You can create provisioning requests in IdentityIQ using any of the following actions or activities:

Provisioning requests create a provisioning plan that the Provision Broker can analyze and process. In all cases, except certification and policy violation-generated requests, provisioning requests create a Workflow case. The Workflow case manages the processing of the provisioning request based on a defined Workflow. See also “Processing Provisioning Requests” on page 156.

Certifications

During a Certification Access Review, certifiers review the system entitlements granted to sets of identities. Access can be approved or revoked for an identity. This certification process can result in:

- **Certificate Remediation** — When an identity’s access to a system is determined to be inappropriate for their job function, the certifier can revoke the entitlement through the Certification Access Review. This process creates a remediation provisioning request in IdentityIQ to remove that access from the source application.
- **Provisioning through Certifications** — When a business role is approved for an identity and that role includes required IT roles the identity does not have, the certifier is prompted to select whether the missing roles must be provisioned for the identity or whether the business role must be approved without provisioning the missing roles. If the certifier elects to provision the missing roles, a provisioning request is created.

  Note: **This provisioning option is only presented during the Access Review if the option Enable Provisioning of Missing Role Requirements is selected in the certification specification.**

  All revocations and provisioning requests from a specific access review are combined into a single provisioning plan and processed together except in certifications where revocations are processed immediately, such as certifications with the Process Revokes Immediately setting selected.

Policy Violations

Policies defined in IdentityIQ enable the system to evaluate an identity's access or activities and report any inconsistencies with company policies. Violations are reported to the violation owner, often the identity's manager, or the appropriate application owner. The violation owner can then permit an exception or initiate a remediation request. The following types of policy violation remediations are available:

- "Policy Violation Remediations for SOD Policy Violations" on page 151
- "Policy Violation Remediations for Non-SOD Policy Violations" on page 151

Policy Violation Remediations for SOD Policy Violations

Only remediations for role or entitlement Separation of Duties (SOD) violations generate a provisioning request to revoke the invalid access. For example, when a manager evaluates an identity's SOD violations and determines that one of the accesses for the identity must be removed, the manager can request the revocation of the invalid access.

You can create policy violation remediation requests from:

- Policy owner's Policy Violation page that you can from **Manage -> Policy Violations** page.
- Certification on which the violation is noted.

Policy Violation Remediations for Non-SOD Policy Violations

Note: By default, you cannot remediate non-SOD policy violations with a certification or in the policy violation window.

You can perform the following actions to enable certification remediate and generate a Work Item:

1. Edit the XML for any policy to include **remediated** as one of its `certificationActions` values to enable certification remediation on that policy type.
2. Select the remediation option for the violation in a certification to automatically create a Work Item that informs the appropriate party of the need to manually correct the violation.

Identity-Refresh-Driven Assignments

You can use the following options on an Identity Refresh task to generate provisioning requests for identities:

- **Refresh assigned, detected roles and promote additional entitlements** — Creates provisioning requests for IdentityIQ to add roles to identity cubes.
- **Provision assignments** — Creates provisioning requests that apply to external applications.

The following table describes these options in more detail:

Option	Description
Refresh assigned, detected roles and promote additional entitlements	Runs the defined assignment rules for roles and examines role detection profiles to update the Assigned and Detected role lists for the identity. **Note: This option does NOT provision access in external system**

Option	Description
Provision assigned roles	Generates provisioning requests to add entitlements required by the currently assigned roles, which can include: - Entitlements for newly assigned roles - Entitlements missing from previously assigned roles. **Note: If a role was previously assigned through an automatic assignment rule and the rule no longer returns true, provisioning requests are generated to remove the entitlements that the role requires. If another assigned role requires those entitlements, they are not removed.**

Note: By default, the entitlements associated with a role are de-provisioned when the role is removed from an identity. The Disable deprovisioning of deassigned roles option overrides that default and leaves the entitlements intact for the identity while the role is removed.

Lifecycle Manager Requests

Lifecycle Manager is a separately licensed portion of the IdentityIQ product that is designed to manage entitlements using provisioning requests. Based on their manager status and how the Lifecycle Manager is configured, users can make requests for themselves or for other identities.

In a typical configuration:

- Managers can make requests for their direct reports.
- Help desk users can make requests for themselves and others.
- Any user can make requests for themselves.

Lifecycle Manager Toolbar

When Lifecycle Manager is enabled, the Lifecycle Manager toolbar displays at the top of the IdentityIQ view and supports the following actions:

- “Request Access” on page 153
- "Manage Accounts" on page 153
- "Other Lifecycle Manager Options" on page 154

Note: The set of identities for which these actions can be taken is based on the individual user’s authority and the Lifecycle Manager configuration. The self-service, Request For Me, options do not include Create Identity.

Request Access

Request Access includes Role and Entitlement requests. If you are working with a single user, a third tab, **Current Access** displays that you can use to request the removal of Roles or Entitlements. Use the Lifecycle Manager Request Roles feature to generate requests that:

- Add the appropriate role to the specified identities.
- Provision the entitlements the role requires.
- Provision permitted roles, if added to the request when prompted.
- De-provision by removing roles from an identity
  This option generates a provisioning request to remove the role assignment from the identities and the entitlements the role requires if another role does not need the entitlements.

Use the Lifecycle Manager **Request Entitlements** feature to generate requests to:

- Add the entitlement to the specified identity.
- Revoke an identity's current entitlements.
  This option generates a provisioning request that removes the access from the source application or applications.

By default, when you request a new entitlement on an application and the user already has an account on that application, the entitlement is added to the existing account. If needed, you can create a separate account for specific entitlements.

To create multiple accounts for a single identity on an application or to add an entitlement to a specific existing account when several are available:

1. Navigate to the Lifecycle Manager configuration Additional Options page.
2. In the **General Options** section, select an application included in the list for **Applications that support additional account requests**.
3. For the **Account** selection, select the option to create a new account or the option to add the entitlement to an existing account that the identity already has.

Manage Accounts

Use the **Manage Accounts** feature to:

- Request accounts on additional applications — generates provisioning requests.
- Revoke or disable existing accounts — generates provisioning requests.
- Enable disabled accounts — generates provisioning requests to enable or disable accounts.
- Unlock locked accounts — generates provisioning request.

To use the Manage Accounts to request a new account:

1. Navigate to the Lifecycle Manager configuration Additional Options page.
2. In the **Manage Accounts Options section**, select an application included in the list of applications that support account-only requests.
3. For the **Account** selection, select the option to create a new account or the option to add the entitlement to an existing account held by the identity.

   Note: You can also select the Manage Accounts option on the Lifecycle Options page for any group, they can enable, disable, and delete accounts for the existing accounts. The connector must support this action and the action must not be disabled through another setting on the Additional Options page.

Other Lifecycle Manager Options

Other Lifecycle Manager options include the following items:

- **Create Identity** — Creates provisioning plans that update IdentityIQ. You can create a new IdentityIQ identity with a set of attributes that can be configured. The attributes that you can set or change are defined by a form that can be customized. New identities do not have accounts on any application.
- **Edit Identity** — Creates provisioning plans that update IdentityIQ. You can modify attributes for an existing IdentityIQ identity. The attributes that you can set or change are defined by a form that can be customized.

  Note: Life Cycle Events can cause provisioning outside of IdentityIQ or additional provisioning inside IdentityIQ. In addition. Attribute sync can also cause provisioning outside of IdentityIQ based on create or edit identity.
- **Manage Passwords** —Resets passwords on target systems which involves a provisioning plan and provisioning action.
- **View Identities** — Does not have provisioning-related functionality and is read-only.

Lifecycle Event-Driven Provisioning

With Lifecycle Manager enabled, Lifecycle Events can be configured in IdentityIQ to represent activities that occur during the normal course of a person's employment at a company. These activities include events such as joining the company, changing departments or managers, and leaving the company. The shorthand terms for these activities are Joiner, Mover and Leaver.

When Lifecycle Manager is enabled, IdentityIQ contains four pre-defined Lifecycle Events.

Lifecycle Event	Trigger	Business Process Invoked
Joiner	Identity Creation	Lifecycle Event – Joiner
Leaver	Attribute Change: **Inactive** attribute change from `false` to `true`	Lifecycle Event – Leaver
Manager Transfer	Manager Change	Lifecycle Event – Manager Transfer
Reinstate	Attribute Change: **Inactive** attribute change from `true` to `false`	Lifecycle Event – Reinstate

By default, these events are disabled and must be enabled before the events can be triggered. Lifecycle Events are triggered by specific changes to an identity. These changes can include the following actions:

- Creation
- Manager transfer
- Attribute change
- Complex changes that an `IdentityTrigger` rule detects

The triggered Lifecycle Events invoke business processes, or workflows, that can contain provisioning actions.

Note: The terms Business Process and Workflow are synonymous. The IdentityIQ user interface refers to these terms as Business Processes which is the term business managers use most often. The IdentityIQ object model and XML use the term Workflows.

Manage Lifecycle Events and Actions

The Lifecycle Events and the default actions of each of the business process that the pre-defined Lifecycle Events invoke are listed below.

- **Lifecycle Event – Joiner** — Prints the name of the identity to sysout. No actions are taken on the identity. This action is typically modified to provision birthright access for identities.
- **Lifecycle Event – Leaver** — Creates and runs a provisioning plan to disable all accounts the leaving identity has.
- **Lifecycle Event – Manager Transfer** — Prints names of the old and new manager to sysout. No actions are taken on identity or entitlements. This action is typically modified to generate a certification for the new manager to review the access an identity holds. This action can also be used to provision birthright access identified for members of new manager's group.
- **Lifecycle Event – Reinstate** — Creates and runs a provisioning plan to enable all previously disabled accounts that a returning identity had.

Lifecycle Events and Actions How-To Tasks

You can perform the following tasks for Lifecycle events and actions:

- "How To Edit Pre-defined Lifecycle Events" on page 155
- "How To Create a New a Lifecycle Event" on page 155
- "How To Delete a Lifecycle Event" on page 155

Note: Additional Lifecycle Events and workflows/business processes can be created as needed to support the business needs for each installation.

How To Edit Pre-defined Lifecycle Events

1. Navigate to **Setup -> Lifecycle Events** page.
2. Right-click an entry and click **Edit** or double click an entry.
3. Make desired changes and click **Save**.

How To Create a New a Lifecycle Event

1. Navigate to **Setup > Lifecycle Events** page.
2. Click **Add New Lifecycle Event**.
3. Enter information for **Lifecycle Event Options** and **Behavior**.
4. Click **Save**.

How To Delete a Lifecycle Event

1. Navigate to **Setup > Lifecycle Events** page.
2. Right-click an entry and select **Delete**.

How To Modify Actions for Lifecycle Events

1. Navigate to Navigate to **Setup -> Business Process** page.
2. Select the **Process Designer** tab.
3. Select a process from the **Edit An Existing Process** list

Note: Typically only administrators can edit the identity cube information. This option is available through the Identities > Identities Warehouse.

You can also access IdentityIQ Debug pages and modify actions through the XML Workflow.

See also "Business Process Management" on page 171.

Other Identity Cube Modifications

In addition to the Lifecycle Manager pages, users with the right capabilities can access an administrative interface to make additional identity modifications. Navigate to the **Identities > Identities Warehouse** page.

Most of the information is read-only, but a provisioning plan is generated that updates an identity when you:

- Edit attribute values on the **Attributes** tab.
- Delete or Move account links from the **Application Accounts** tab.
- Change capabilities or assigned scopes on the **User Rights** tab.

If the triggering attributes for the identity have not changed, deleted roles that were assigned by rules are automatically re-assigned to the identity during the next identity refresh. The re-assignment is also processed as an identity-refresh-driven provisioning request.

Processing Provisioning Requests

IdentityIQ creates a master provisioning plan for the requested actions when a provisioning request is submitted from a provisioning request source. A workflow case is also created to manage and track the progress of the provisioning activity. The workflow case contains the workflow that specifies the process to follow.

Note: Certification and policy violation based provisioning does not use workflows.

IdentityIQ ships with pre-defined workflows or business processes which can be customized for each installation as needed. The workflow case created for each provisioning request is associated with the appropriate workflow for the event that generated the request. The following table lists the Workflows that drive the provisioning process from each request source.

Provisioning Request Source	Workflow Invoked
Lifecycle Manager	Main workflows include: LCM Create and Update, LCM Manage Password, LCM Registration and LCM Provisioning
Identity Refresh	Identity Refresh
Define Identities	Identity Update
Lifecycle Events	Each event is managed by the business process listed in **Business Process** field on the Lifecycle Event definition window.

Provisioning Request Source	Workflow Invoked
Certification Remediations / Provisioning	None Managed by and RemediationManager class. **Note: If the certification specifies Process Revokes Immediately, certification starts the remediation process directly.**
Policy Violation Remediations	None Policy violations remediations that certifications create are managed the same as any other certification remediation. Policy violations remediated from **Policy Violations** page are saved directly to the violation table.

Involvement

The **Perform Maintenance** task processes all certification remediation including: roles entitlements and policy violations. This task invokes the Remediation Manager to process the remediation requests.

For certifications with specifications that include the **Process Revokes Immediately** option, the `Certificationer` object invokes the Remediation Manager directly to process the remediation requests. The basic logic of the provisioning process remains the same. The Remediation Manager uses the same mechanisms that the workflows use to complete the requests.

Remediation tasks that are performed on the **Policy Violations** page are not a part of these maintenance task processes.

Note: Requests on a certification for provision-missing-required-roles are not remediation items. These requests are added to the same provisioning plan as additional actions. The process that manages remediation items also manages these requests.

Overview of Provisioning Process

The Provisioning Process has three phases:

- "Compiling the Plan" on page 158 — Analysis and preparation of the plan for processing
- "Answering Provisioning Policy Questions" on page 161 — Request of missing required data from a user
- "Implementing the Plan" on page 162 — Submittal of the plan to the appropriate connector to provision the requested access

Compiling the Plan

The Plan Compiler is responsible for the following tasks:

- "Create the Provisioning Project" on page 158 — This task begins with the provisioning plan.
- "Evaluate and Expand Roles" on page 158 — Roles are expanded into entitlement requests.
- "Apply Provisioning Policies" on page 159 — This task a applies the policy which contains list of fields with names that correspond to an application account attribute name the role uses.
- "Identify Questions" on page 161 — This task identifies any missing information for the requests.
- "Filter and Check Dependencies" on page 161 — These tasks streamline the provisioning process and prevent unintended consequences of the requests.
- "Partition the Plan" on page 161 — This task divides the provisioning plan into smaller plans.

Create the Provisioning Project

The provisioning project begins with the provisioning plan. As roles are expanded into entitlement requests, missing information for the requests is identified and the plan is divided into smaller plans. The provisioning project serves as a container for all the smaller plans and includes the following items:

- Original (or master) provisioning plan.
- A set of partitioned plans that contain requests for a single connector.
- An unmanaged plan that contains requests that cannot be processed by any connectors. Items in this plan can be converted to manual work items.

Note: Partitioned plans contains the actions necessary to fulfill the master plan. For example, a single role assignment in the master plan can expand into many entitlement requests in the partitioned plans.

Evaluate and Expand Roles

The master plan is evaluated for any role assignments. If the plan contains role assignments, those roles must be expanded. The role expansion process:

- Identifies IT roles that an assigned business role needs.
- Determines what specific entitlements the IT role needs.
- Adds the entitlements to the lists of account/attribute/permission request for the provisioning project. Each attribute is represented as an Attribute Request or a Permission Request.

For example, Business Role X is added to an identity. Business Role X requires IT Role A which has entitlements associated with its role. The Plan Complier determines that IT Role A is required, identifies the necessary entitlements, and adds the entitlements to the project.

Note: After role expansion is complete, IT Role A does not display in the project. Only the raw entitlements that the IT role A needs are listed.

Apply Provisioning Policies

A provisioning policy is a list of fields with names that correspond to an application account attribute name the role uses. Provisioning Policies can be used to help complete an access request that has unknown data required for provisioning. When a provisioning request requires additional information to complete the access request, you can apply a provisioning policy specified for the application involved. Examples of additional or unknown data that is required for provisioning include the following items:

- Information that is not provided in the original request, such as missing information for a new account or missing information for an additional required role.
- Multiple possible values for a required field.

Types of Provisioning Policies include:

- "Role Provisioning Policies" on page 159 — Removes role uncertainty.
- "Application Provisioning Policies" on page 160 — Applied when a new account is requested.

Role Provisioning Policies

The primary purpose of provisioning policies on roles is to remove any uncertainty for the role. In some cases, examining the role profile can determine the set of entitlements to be provisioned for an IT role. Role profiles can be clear or unclear. When all the role profile terms are joined using AND statements, the profile is clear. IdentityIQ can easily analyze the role profile and provision entitlements that match the profile.

For example, A profile that includes a list of OR terms is unclear, because two or more different `memberOf` values can satisfy the role. The following table provides examples.

Role Profile Example Terms	Type of Terms	Explanation
location='Austin' and memberOf='Engineering'	Profile with a list of AND terms	To satisfy this role, the identity must have both of these account attributes. Requests for those two attributes are added to the plan.
memberOf='Engineering' OR memberOf='Sales'	Profile with a list of OR terms	The default provisioning behavior for profiles containing OR terms is to provision only the first one. In this case, memberOf='Engineering' is added to the plan but not memberOf='Sales'. **Note: If the organization wants memberOf='Sales' provisioned for new role members, a provisioning policy can be defined with one field named memberOf with the field value Sales.**

Note: Fields can also be assigned scripts or rules that enable the appropriate value to be calculated instead of using a hard-coded value.

Application Provisioning Policies

Provisioning Policies can also be specified for applications. These policies are applied when a new account is requested on an application. Application Provisioning Policies are similar to Role Provision Policies and can specify the field values as literal values or through a script or rule. The following actions trigger application provisioning policies:

- Create Account
- Update Account
- Delete Account
- Enable Account
- Disable Account
- Unlock Account
- Change Password
- Create <object type>
- Update <object type>

Application Dependency

You can specify an application dependency at the field level when you create a policy. Application dependency works with synchronous connectors and does not work with connectors that queue plans. Application dependencies are enforced during Create operations. For update and delete, the dependencies are ignored.

Note: IdentityIQ does not undo dependencies during de-provisioning.

To specify an application dependency:

1. Navigate to **Applications -> Application Definition**. On the Provisioning Policies tab of the Application Configuration page select the dependent application for the provisioning.
2. Double-click or right-click the application in the **Application List**.
3. On the Application Configuration page, select the **Provisioning Policies** tab.
4. Define the application dependency at the field level in the **Create Account** and **Create Group** policies.

Application dependency works similar to roles and entitlements. If a dependency is missing, IdentityIQ expands it and executes a Create request for the dependency. If the user has an existing link on a dependent application, IdentityIQ uses the existing link information to derive the value. When there are multiple accounts on the link, the applicable accounts are selected automatically using rules or through an interactive user interface. Selecting an account can be an option to create a new account.

The available attributes are derived from the account schema of all dependent applications. During plan compilation, IdentityIQ reads these properties and determines any new accounts that are required to satisfy the dependency.

During Plan Evaluation, IdentityIQ uses the dependency settings to determine the order that must be used to implement the plan. If a dependency plan fails, all of the dependent plans also fail. If a dependency plan requires a retry, after the retry is successfully completed, the dependent plans are executed. There is special new logic in the Provision with Retries method that loops back to the provisioning step when there are still plans to complete.

There are not transformations (rules) on dependent fields. The evaluation process copies the exact values from the dependency plan or link to the dependent plan.

Identify Questions

After the provisioning policies are applied, pieces of data can still be missing. Some provisioning policies are specifically written so the data must be obtained from a person when the role or application account is requested. These missing data elements are recorded as questions on the provisioning project. These questions are presented to a person who must provide the information necessary to complete the provision request. See “Answering Provisioning Policy Questions” on page 161.

Filter and Check Dependencies

Filter and Check Dependencies streamline the provisioning process and prevent unintended consequences of the requests. During this step of the compilation process:

- The current state of the identity is examined.
- Any entitlements requested in the plan that already exist for the identity are removed from the plan.
- Entitlements that are to be removed, based on a role removal, are examined. This step determines if the identity has another role that requires the entitlement that is scheduled to be removed.

Note: If the identity has another role that requires that entitlement, the entitlement removal request is taken out of the plan.

Partition the Plan

At the end of plan compilation, all the individual entitlement requests identified from the original master plan and the role expansion are partitioned into a set of smaller provisioning plans – one per target. The targets are designated by the connector or **integrationConfig** that IdentityIQ uses to communicate with them. Connections can include:

Note: Any requests in the plan that cannot be processed by any of the integration configurations or read-write connectors are added to the unmanaged plan and are processed manually through IdentityIQ Work Items.

See also "Implementing the Plan" on page 162.

Answering Provisioning Policy Questions

After the plan is compiled, the project can have unanswered questions that must be presented to a person to answer. The provisioning broker does not interface with the user and cannot get answers to these questions. The workflow process, the component that controls the provisioning process, is responsible for getting the questions answered.

Exceptions

Because the following processes can not present forms to users, this interactive provisioning policy phase does not apply for the associated provisioning activities. These requests are only fulfilled if they can be completed with the available information. Because remediation requests are access removal requests, these requests should not require any additional data.

- Processes that manage certification remediations
- Processes that manager provisioning activities
- Policy-violation remediations

Generally projects that have unanswered questions are only an issues if the projects have activities that require a new account to be created for a new assignment or a missing role.

Provisioning Forms

The Lifecycle Manager Provisioning, Identity Refresh, and Identity Update Workflows invoke the Do Provisioning Forms business process. This process presents questions on user-facing forms and collects the answers. The Do Provisioning Forms process separates these actions into the following steps:

- Build Provisioning Form
- Present Provisioning Form
- Assimilate Provisioning Form

Optionally, you can assign owners for individual provisioning policy fields. When an owner is assigned, any questions related to the field are sent to the field owner and not to the access requester. The controlling workflow identifies who receives the questions and then submits the forms to the correct identities.

By default, the Lifecycle Manager Provisioning Workflow contains two opportunities to present provisioning forms to a user, pre-approval and post-approval. The following named steps run the **Do Provisioning Forms** workflow:

- Identity Request Initialize
- Identity Request Provision

A Workflow can have a different number of approval steps between the steps that present provisioning forms. Each approval can modify items in the master plan that cause the project to be recompiled. For example, if an approver rejects one of the role assignments, provisioning questions for an account that role requires might not be needed.

Implementing the Plan

After the plans are partitioned and any missing fields are provided, the subdivided plans can be implemented through one of the following mechanisms:

- “Integrations” on page 163
- “Direct Read-Write Connectors” on page 163
- “Work Items” on page 164

The results are recorded in the plan and indicate if the request was implemented immediately or placed in a queue for future implementation. This status determines when the identity cube is updated to reflect the provisioned changes. See also “Updating the Identity Cube” on page 164.

The following table provides an overview of the provisioning mechanism.

Provisioning Mechanism	Plan Implementation
Integration Executors	Managed plan implementation using integration executors.Starts as an asynchronous process that might not complete immediately.
Direct Read-Write Connectors	Application objects contain the provisioning configuration.
Work Items	Unmanaged plan implementation using the controlling workflow.

Integrations

Integrations are a separately licensed components that communicate with systems within your network. The following table provides and overview of the integration modules and connectors.

System	Module	Connector
Provisioning systems, such as: OIM, ISIM, FIM	Provisioning Integration Modules (PIMs)	Read/write connectors and IntegrationConfigs/Executors
IT Service Management, such as: Remedy, Service Now, HP Service Manager	Service Integration Modules (SIMs)	IntegrationConfigs/Executors
Mobile device management systems, such as: AirWatch, MobileIron, Good Technology	Mobile Integration Modules (MIMs)	Read/write Connectors
IT Security: HP ArcSight	IT Security Integration Module	IntegrationConfigs/Executors
Enterprise Applications: Oracle EBS, SAP Portal, PeopleSoft, Siebel and NetSuite	Enterprise Resource Planning Integration Modules (ERP Integration Modules)	Read/Write Connectors
Mainframe: RACF, CA-Top Secret, CA-ACF2, RACF LDAP and Top Secret LDAP	Mainframe Integration Modules	Read/Write Connectors
Healthcare: Epic and Cerner	Healthcare Integration Modules	Read/Write Connectors
Identity Intelligence/Analytics: SAP GRC	GRC Integration Module	IntegrationConfigs/Executors
Governance platform: Amazon Web Services and SAP	IaaS Governance Modules	Read/Write Connectors

Integration Executors attempt an immediate update of the target application. If the immediate update attempt is unsuccessful, Integration Executors, place the activity in a queue.

Note: Even if the activity does not immediately commit, the Integration Executors cannot communicate back to IdentityIQ when the request is completed. Therefore, these requests are always considered to be queued.

See also "Plan Initializer Rule or Script" on page 164.

Direct Read-Write Connectors

Read-write connectors are available to manage data communication between IdentityIQ and an ever-increasing number of applications. For applications using these connectors, you manage provisioning activities through the variables in the Provisioning configuration for that application.

Provisioning using direct read-write connector with these applications is fully automated. These connectors generally:

- Run the plan immediately.
- Can report back a committed status to IdentityIQ in real time.
- Confirm that the changes can be reflected on the identity cube immediately.

See also "Plan Initializer Rule or Script" on page 164.

IdentityIQ Updates

For items that require updates to IdentityIQ, such as roles assigned to an identity or identity attribute changes, a separate plan is created. These requests are similar to direct connector updates. Although no connector is required to complete these internal updates, the requests are run immediately and are reported back as committed when updated.

Work Items

Work Items, opened in IdentityIQ that contain provisioning instructions, to provision unmanaged plans. The controlling workflow or Remediation Manager is responsible for implementing an unmanaged plan. An unmanaged plan:

- Includes provisioning requests to any application where data is aggregated using read-only connectors.
- Does not have an Integration Executor that communicates with the plan.
- Are identified and examined after the Integration Executors and direct read-write connectors are called.

If the unmanaged plan contains any requests, one or more work items are opened in IdentityIQ that contains the provisioning instructions from the plan. Each work item is assigned to a user who is responsible for implementing the changes required to complete the specified provisioning tasks. Work item assignees are often the application or entitlement owner. When the provisioning action is completed, the work item assignee must manually mark the work item as complete.

Note: Provisioning tasks managed through work items are considered queued, rather than committed. Even if the assigned user marks the work item complete, IdentityIQ cannot determine with certainty if the changes were actually made until the next aggregation from the source application is completed.

Plan Initializer Rule or Script

You can specify a Plan Initializer rule or script to run during the implementation of the provisioning plan. An installation-specific rule or script can be added to integration and provisioning configurations. When a rule or script is specified, it runs immediately prior to running the provisioning activity for the application. Provisioning is based on the provisioning plan and application associated configuration or integration executor.

See also “Implementing the Plan” on page 162.

Updating the Identity Cube

Provisioning activities that occur completely within IdentityIQ, such as assigning a business role to an identity, are the only provisioning actions that change the information on the identity cube. For example, implementing a provisioning plan does not update role detections. You must perform an Identity Refresh to update the identity based on the provisioned items. For example, to update the list of detected entitlements and roles, you must perform an Identity Refresh.

Identity Refresh

Provisioning workflows generally includes an Identity Refresh step than can be enabled or disabled as needed for the provision activity. To perform an identity refresh to update the Identity Cube, you must:

- Include an Identity Refresh step in the Workflow, or
- Run an Identity Refresh task after the Workflow completes.

To enable the Refresh step in the workflow the `doRefresh` variable must be set to `True`.

General Guidelines

Direct Read-write connectors — For Direct read-write connectors that process requests immediately, the Identity Refresh step is generally enabled. The changes to application accounts that the connectors make are usually displayed immediately in IdentityIQ.

Queued Requests — Requests that were queued are not applied to the identity cube until a re-aggregation has occurred from the application involved. As a result, the Identity Refresh step is typically disabled for provisioning workflows that are managing integration configuration-driven provisioning activities, because the refresh can not detect any changes until after an aggregation from the source system.

Items that were processed as Work Items from the unmanaged plan are treated as queued requests, because manually closing a Work Item does not necessarily indicate all the work was completed. To confirm that the request was processed, you must perform a re-aggregation from the source system. This aggregation must be followed by an identity refresh to update the identity cube with the information.

Because the **Application Accounts** tab for the Identity Cube displays account data that is recorded on the Link object for the identity, the tab lists the provisioned access immediately following the read-write connector commit or following a re-aggregation from integration configuration-managed applications. However, the entitlement data on the **Entitlement** tab and in any certification is not updated until the Identity Refresh task has run.

Special Case: Optimistic Provisioning

When the workflows are configured for Optimistic Provisioning, provisioned changes appear in IdentityIQ before the changes are confirmed through re-aggregation. Optimistic Provisioning assumes that provisioning requests are completed and then updates the identity cube to display the changes when the request is submitted, not when the request is verified.

Optimistic provisioning configuration is useful for some testing scenarios or product demonstrations, but it is not an ideal configuration for most production environments. Companies often prefer that IdentityIQ indicates a confirmed state of system access and not a desired state.

To configure the workflows for Optimistic Provisioning:

1. Verify that the workflow has the Set the `optimisticProvisioning` process variable. By default, most provisioning-related workflows are configured with this argument
2. Set the optimisticProvisioning process variable, or XML arg, option to True. The default value is false.

   Note: To modify other workflows, add the variable and then follow the steps listed above.

Summary of Workflows, Tasks, and Rules in Provisioning

The following table provides an at-a-glance list of workflows, tasks and rules for provisioning through IdentityIQ.

Type	Name	Purpose / Usage
Workflow	Lifecycle Manager: LCM Provisioning LCM Create and Update LCM Manage Passwords LCM Registration	Manages actions requested through Lifecycle Manager.
Workflow	Identity Update	Manages the provisioning actions required based on an Identity Cube update.
Workflow	Identity Refresh	Manages the provisioning actions required from an Identity Refresh.
Workflow	Lifecycle Event – Joiner Lifecycle Event – Manager Change Lifecycle Event – Leaver Lifecycle Event – Reinstate	Controls the Lifecycle Event-driven activities, which can contain provisioning actions.
Workflow (subprocess)	Do Provisioning Forms	Creates, presents and gathers data from provisioning forms. This step is the interactive provisioning policy phase of provisioning.
Workflow (subprocess)	Do Manual Actions	Presents the unmanaged portion of a provisioning project as work items to be processed manually. Update and Identity Refresh workflows use this step. Lifecycle Manager has a similar step but audits differently.
Workflow (subprocess)	Provision with Retries	Manages retries on the provisioning actions for Lifecycle Manager.
Workflow (subprocess)	Identity Request Initialize Identity Request Violation Review Identity Request Approve Identity Request Approve Identity Changes Identity Request Provision Identity Request Notify Identity Request Finalize Provisioning Approval Subprocess	These workflows subdivide Lifecycle Manager Provisioning into more manageable workflow parts. LIfecycle workflows also use some or all of these tasks.
Task	Identity Refresh	Creates provisioning requests based on application of role assignment rules or role detection.
Task	Perform Maintenance	Processes certification-generated and policy violation-generated remediation requests.

Type	Name	Purpose / Usage
Task	Account Aggregation	Provisioning activities driven by integration configurations or Work Items require a re-aggregation from the target system before the identities can be updated with the access change.
Rule	FieldValue	Identifies the default value for the Provisioning Policy field.
Rule	AllowedValues	Constrains allowed values for the Provisioning Policy field.
Rule	Validation	Defines validation process for Provisioning Policy field.
Rule	Owner	Defines owner for Provisioning Policy field.
Rule	PlanInitializer	Can be specified for any IntegrationConfig or ProvisioningConfig to run installation-specific pre-processing in Plan Evaluation step before carrying out provisioning.
Rule	IdentityTrigger	Can determine the triggering of a Lifecycle Event.

Business Processes\Workflow

This section contains the following information:

Chapter 11: Business Process Management

A Business Process is a sequence of operations or steps that are launched to perform work. IdentityIQ Business Processes include standard “out-of-the-box” processes and custom “installation-specific” processes. System events trigger both standard and custom IdentityIQ Business Processes. The informal term, workflow, is used in this section to refer to a business process.

The following events can trigger a workflow:

- Role creation or modification
- Account Group creation or modification
- Identity update
- Identity refresh
- Identity correlation
- Deferred role assignment, de-assignment
- Deferred role activation, deactivation
- Any Lifecycle Manager event
- Any Lifecycle Event (marked by changes to an Identity's attributes)

Custom workflows can be defined to do a wide variety of processing tasks. You can use:

- IdentityIQ workflow library methods and rules.
- Custom BeanShell scripts and rules.

Customizing or creating workflows generally involves a combination of XML and Java/BeanShell programming. You can manage some customization activities with the IdentityIQ graphical process editor that is included in the product. To customize or create new workflows, typically you need to be comfortable writing XML and Java.

This section has the following topics:

Chapter 12: Workflow Basics

This section contains some key concepts for developing and using workflows.

Terminology

The terms, Business Process and Workflow, are used synonymously In IdentityIQ and throughout this document. The IdentityIQ user interface refers to these sets of connected actions as Business Processes, which is the term that business managers often use.

Important Workflow Objects

The IdentityIQ Object Model uses four key objects in workflows. To work with workflows, you need a basic understanding of these objects.

Table 2—Important Workflow Objects

Object	Usage
Workflow	Defines the workflow structure and steps involved in the workflow processing.
WorkflowCase	Represents a workflow in progress. Contains a workflow element in which the process is outlined and current state data is tracked. Contains identifying information about the workflow target object.
WorkflowContext	Tracks launchtime information the Workflower maintains as it advances through a workflow case. Passed into rules and scripts and to the registered WorkflowHandler. Contains all workflow variables, step arguments, current step or approval, workflow definition, libraries, and WorkflowCase.
TaskResult	Records the completion status of a task, or in this case, the workflow. Contained within the WorkflowCase.

Note: The most important object for writing workflows is the WorkflowContext object, which tracks the launchtime state of the workflow and performs other critical functions. Because WorkflowContext methods are used in workflows, data can be extracted from it as needed within any step of the workflow.

Workflows Operation

Workflows carry out a sequence of defined actions based on a triggering event and can be used for a variety of activities within the system. In its launching state, a workflow is tracked through a workflow case, which manages only one target entity at a time (one identity, one role, one provisioning plan, etc.).

Note: **If multiple identities are modified at one time in a way that requires a workflow to launch for all of the identities, a separate workflow case is created to track the processing of the workflow for each single identity.**

Provisioning Plans in Workflows

A provisioning plan contains a list of requested changes to an identity. Most workflows that change identities contain a single provisioning plan in a workflow variable. When performing Workflow customization you commonly need to inspect and sometimes need to modify the provisioning plan.

Note: **Only one provisioning plan can be referenced in a workflow case at a time.**

When you request changes for more than one identity at a time, even if the same change is requested for all the identities:

- A separate provisioning plan is created for each identity.
- A separate workflow case is created to manage the provisioning plan created for each identity.

Triggering Workflows

Events that occur in other parts of IdentityIQ and changes to attributes can trigger Workflows. Common Workflow triggers include the following items:

- **Lifecycle Manager Actions** — Requests to change an identity's roles, entitlements, or accounts can activate workflows.
- **Lifecycle Events** — Creating an identity, deactivating an identity, or moving an identity from one manager to another manager can activate workflows.
- **Non-Lifecycle Events** — Editing a role, editing an account group, and changing a password can active workflows.
- **Identity Attribute Change** — Value changes can activate workflows.
- **Policy Violations** — A policy violation can activate workflows.

The following table lists the four main areas of IdentityIQ where you can associate Workflows to system activities.

Table 3— IdentityIQ Setup for Workflows

Workflow Trigger	IdentityIQ Setup
Lifecycle Manager Requests	Select **Lifecycle Manager** from the gear icon menu and go to the Business Processes tab.
Lifecycle Events	Select **Lifecycle Events** from the **Setup** menu and specify the business process behavior.
Non-LCM-related Events	Linked to triggering events. Select **Global Settings** from the gear icon menu and go to the IdentityIQ Configuration page. Select the Identities, Roles, or Miscellaneous tab and then select a business process.

Table 3— IdentityIQ Setup for Workflows

Workflow Trigger	IdentityIQ Setup
Identity Attribute Change	Configured with a Value Change Workflow. Select **Global Settings** from the gear icon menu and go to the Identity Mapping page. Click an attribute to edit or add a new attribute. On the Edit Identity Attribute page, go to the **Advanced Options -> Value Change Workflow** option to select the business process.
Policy Violation	Select **Policies** from the **Setup** menu, select or create a new policy, and specify the business process behavior

Note: You can also configure an IdentityIQ task to trigger a workflow. This workflow set up is a more complex process. See "Advanced Workflow Topics" on page 233.

IdentityIQ Default Workflows

IdentityIQ is preconfigured with various standard workflows that manage activities. The following workflows are examples of default workflows that are included with the product:

- Provisioning of roles or entitlements
- Account management
- Identity creation
- Password management

The default workflows can be configured and customized to address the specific business requirements of each installation. Additionally, you can write new workflows and apply them to any of the actions in IdentityIQ that support workflows.

Workflow Types

Default workflows have pre-defined workflow types. IdentityIQ uses these assigned types to determine which workflows to present in the Business Process configuration list boxes. Workflows can be specified to activate based on a specific system event.

For example, role create, update, and delete actions can trigger a **RoleModeler** type of workflow. Only workflows of that type are listed in the drop-down list for that configuration option.

Note: You can assign custom types to workflows. However, custom type workflows can only be triggered through the user interface on Lifecycle Events, which can trigger workflows of any type.

The table below lists the workflow type associated with each type of action within IdentityIQ.

Table 4—Workflow Types

Process Type	Description
Policy Violation	Workflow activated to launch policy violation actions.
Batch Provisioning	Workflow activated to launch batch requests.
Scheduled Assignment	Workflow activated to when a role is ready to be assigned.

Table 4—Workflow Types

Process Type	Description
Scheduled Role Activation	Workflow activated when a role is ready to be enabled or disabled.
Managed Attribute	Workflow activated when an entitlement is created or edited.
Identity Correlation	Workflow activated when performing identity correlation tasks.
Identity Event	Workflow activated for identity event. For example, sunrise/sunset dates for deferred entitlement, role assignment, or role removal.
Identity Lifecycle	Workflow activated for Lifecycle events. For example, Lifecycle Event - Joiner or Lifecycle Event - Leaver.
Identity Update	Workflow activated when you update an identity through the **Identity** -> **Identity Warehouse** page. Typically requires few or no approvals.
Identity Refresh	Workflow activated for identities that are refreshed using the Identity Refresh task. This type of process can be used for additional customization during refresh, and to present provisioning policy forms if accounts need to be created as a result of automated role assignment.
LCM Identity	Workflow associated with Lifecycle Manager Identity related tasks, for example, LCM Create and Update.
LCM Provisioning	Workflow activated for Lifecycle Manager provisioning tasks.
LCM Registration	Workflow activated for registration tasks.
Policy Violation	Workflow activated to initiate policy violation actions.
Role Modeler	Workflow associated with Role functions. For example, Owner Approval and Role Activation.
Subprocess	Designation of a workflow which is part of a larger workflow.
Password Intercept	Workflow activated when a password change interception event is received.

Sub-process Workflows

Some complex workflows are divided into multiple sub-process workflows that are activated by a master workflow. Using sub-process workflows with a master workflow can:

- Simplify the structure of the master workflow
- Make workflows easier to manage
- Promote re-usability because more than one master workflow can reference the same sub-processes

As a standard practice, these smaller workflows are assigned the **Subprocess** type of workflow. This type is not associated with any system functionality. However, using the Subprocess type designation enables you to easily identify the workflow as a sub-process of a larger workflow.

Transient Workflows

Transient workflows are launched in a special mode that does not persist any information to the database. A workflow remains in the transient state until the workflow reaches an approval step. If the workflow launches to completion without an approval step, nothing is stored in the database unless the browser terminates or the session times out the workflow and any progress made is lost.

Note: If the browser terminates or the session times out the workflow and any progress made is lost.

Examples of transient workflows include:

- QuickLaunch workflows that can present a series of forms before performing any relevant actions.
- Self-registration workflows that do not require authentication
- Workflows for users trying the registration process, who do not have an inbox where they can see their past attempts

To create a transient workflow, add a variable named **transient** and set the value to **true**.

For transient workflows to work correctly, the user interface code needs to manage the workflow case in a special way, through a WorkflowSession.

The case persists when any of these things happen:

- an approval for someone that is not the submitting user
- a step with a wait='x' in it
- a step with background='true'

Chapter 13: Using the Business Process Editor with Workflows

The IdentityIQ user interface provides a graphical tool for defining and editing workflow processes. You can use the IdentityIQ Business Process Editor to:

- Create a new workflow or edit an existing workflow
- Set up the workflow structure.
- Create the steps that define the behavior or the workflow.
- Outline the transitions between the steps.
- Define forms.
- Assign conditions.

This tool also provides a graphical representation of the process flow that can be used to create documentation about the activities included in the workflow.

Typically, administrators use the graphical editor to outline the process and then move to the XML representation to add to or adjust the details of each step. After you save the process, you can view, edit or export the XML representation from the IdentityIQ Debug pages.

Note: Because some workflow steps cannot be defined with the graphical editor, workflow development can involve direct editing in the XML representation and some amount of Java coding. An understanding of XML and Java syntax is a general requirement for workflow development.

Creating and Editing Workflows

Use the Business Process Editor to create a new workflow or edit an existing workflow. Original workflows can also be created from existing processes.

Basic Workflow How-To Tasks

You can perform the following tasks:

- "How To View or Edit a Workflow" on page 179
- "How To Create a New Workflow" on page 180
- "How To Use an Existing Workflow to Create a New Business Process" on page 180

How To View or Edit a Workflow

1. Navigate to **Setup** -> **Business Processes**.
2. Select an existing workflow from the **Edit an Existing Process** list.
3. Navigate through each of the process tabs to view or modify the workflow data.
4. To save changes to an existing workflow, click **Save**.

How To Create a New Workflow

1. Navigate to **Setup** -> **Business Processes**.
2. Click **New** to create a new workflow and then enter a name for your process.
3. Specify a name and description for the workflow. Use a short descriptive name for the workflow and use a the description that provides an overview of the workflow function.
4. In the **Type** field:
    a. Select from the drop-down list of predefined workflow types. The available types are restricted to the process options related to the workflow.
    b. To enter a custom type, manually enter the type name in the box instead of selecting one from the list. See the Workflow Basics chapters for any limitations to custom types.
5. Navigate through each of the process tabs and specify workflow data.
6. Click **Save**.

How To Use an Existing Workflow to Create a New Business Process

1. Navigate to **Setup** -> **Business Processes**.
2. Select an existing workflow from the **Edit an Existing Process** list.
3. Navigate through each of the process tabs to view or modify the workflow data.
4. Click **Save As** and enter a unique name for the workflow.

Process Editor Tabs

The Process Editor has the following tabs:

Table 5—Process Editor Interface Tabs

Interface Tab	Inputs
Process Details	Specify Name, Type, and Description of the workflow. See "Process Editor Tabs" on page 180.
Process Variables	Specify any variables that apply to the workflow. Variables in any input variables, return values, and working variables for use within the process's steps. See "Process Variables Tab" on page 182.
Process Designer	To graphically represent the process, specify the actions involved in each step, and provide the evaluation conditions for moving from one step to another. See "Process Designer Tab" on page 183.
Process Metrics	Review statistics gathered for the process as it launches. See "Process Metrics Tab" on page 193.

Process Details Tab

The Process Workflow tab has the following options:

Table 6—Process Detail Tab - Available Process Types

Process Type	Description
Policy Violation	Workflow activated to launch policy violation actions.
Batch Provisioning	Workflow activated to launch batch requests.
Scheduled Assignment	Workflow activated to when a role is ready to be assigned.
Scheduled Role Activation	Workflow activated when a role is ready to be enabled or disabled.
Managed Attribute	Workflow activated when an entitlement is created or edited.
Identity Correlation	Workflow activated when performing identity correlation tasks.
Identity Event	Workflow activated for identity event. For example, sunrise/sunset dates for deferred entitlement, role assignment, or role removal.
Identity Lifecycle	Workflow activated for Lifecycle events. For example, Lifecycle Event - Joiner or Lifecycle Event - Leaver.
Identity Update	Workflow activated when you update an identity through the **Define** -> **Identity** page. Typically requires few or no approvals.
Identity Refresh	Workflow activated for identities that are refreshed using the Identity Refresh task. This type of process can be used for additional customization during refresh, and to present provisioning policy forms if accounts need to be created as a result of automated role assignment.
LCM Identity	Workflow associated with Lifecycle Manager Identity related tasks, for example, LCM Create and Update.
LCM Provisioning	Workflow activated for Lifecycle Manager provisioning tasks.
LCM Registration	Workflow activated for registration tasks.
Policy Violation	Workflow activated to initiate policy violation actions.
Role Modeler	Workflow associated with Role functions. For example, Owner Approval and Role Activation.
Subprocess	Designation of a workflow which is part of a larger workflow.
Password Intercept	Workflow activated when a password change interception event is received.

Process Variables Tab

The Process Variables tab lists variables you can use with the workflow. For most of the default processes, the variables are listed in a collapsed, advanced view. You can expand the view to show the details for each variable. Variables include:

- Input variables for workflow
- Output variables for workflow
- Working variables used for processing a workflow

Variables are marked as **Input**, **Required**, **Editable**, or **Output**.

To delete a variable, expand the variable and click **Remove**.

Table 7—Process Variable Flags

Object	Usage
Input	Specifies that the variable is one of the arguments to the workflow, passed in when it is launched.
Output	Stores the variable in the workflow's task result to allow the user to view the progress and results of the workflow. To view the results, navigate to **Setup** -> **Tasks** -> **Task Results**.
Editable	Enables the variable to be edited in the basic view.
Required	Indicates that the variable must contain a value (non-null) when the workflow starts.
Description	A brief description of the variable and its function.

Note: The order of variable declarations can make a difference. For variables in the XML that reference other variables in their initializations, the referenced variable must be declared first.

When variables are created through the user interface, the new variables are inserted in the list above the existing variables. When the XML representation of the workflow is generated, the variables are listed in the order they were created, which is the opposite of the display order in the user interface.

Basic View

IdentityIQ has several built-in business processes that are available when you install the product. The commonly used processes are available through the Basic View which is a simplified, form-based view. The information you edit in the Basic View can be also be configured or removed using the Advanced View. The Basic View includes the following business processes:

Note: Business processes with the LCM label are part of IdentityIQ Lifecycle Manager, which is licensed separately.

- Identity Update
- LCM Create and Update
- LCM Manage Passwords
- LCM Provisioning
- LCM Registration

How to Use the Basic View

1. Navigate to the Debug page and edit the XML of the business process.

2. Manually add and configure the **configForm** attribute to reference the form to be presented in the Basic section of process variables. See also “Editing Workflow XML” on page 195.

   Note: If the reference exists in the business process, but the form does not, an error is displayed and you are returned to the Advanced view.

Variable Initialization

To initialize variables for the workflow, specify an initial value for the variable in this panel. For best results, use initial values for the workflow variables, rather than creating multiple process steps to initialize each variable.

There are five ways that initialization can occur:

Table 8—Variable Initialization Options

Object	Usage
String	Assigns a literal value to the variable.
Reference	Sets the variable value through a reference to one of the other workflow process variables.
Script	Sets the variable with a Beanshell script inside the workflow.
Rule	Sets the variable by calling a Beanshell rule outside the workflow.
Call Method	Assigns the return value of a call to a compiled Java method in a workflow library to the variable.

Note: Variable values passed into the workflow through workflow arguments supersede variable initial values. Therefore, any value provided in an argument overwrites the initial value for that argument.

Timing of Variable Definition

Variables that are known at the beginning of workflow development can be defined before the graphical process design begins. Throughout the development process you might need to define other variables. Variables are not restricted to only those that were previously defined on the Process Variables tab. Variable definition can be done before, during, or after the design process.

Process Designer Tab

The majority of the work in creating and modifying a workflow is done on the Process Designer tab. The steps and transitions you create for workflow determine the workflow activities and can include the following items:

- “Process Steps” on page 183
- “Approval Steps” on page 187
- “Form Steps” on page 189
- “Step Conditions” on page 192
- “Step Transitions” on page 192

Process Steps

A workflow involves a minimum of three steps: a start step, a processing step, and a stop step or END. For best results, all workflows should contain a start and stop step and that these two steps contain no actions. Workflows

can contain as many or as few processing steps as are necessary to manage the required actions. To add steps using the Process Designer, navigate to the Process Editor and click the desired step type in the **Add A Step** section. You can drag steps around the Process Designer grid to line them up visually in a logical progression.

To add a new step:

1. Click **Add a Step** in the left-hand column to display panel that contains available steps.

   Note: Only steps associated with the process type and that exist in the Step Library are listed in the Add a Step panel.

2. Click and drag the desired step to a position in the process design grid.

To edit to the contents of a step,

1. Right-click the step icon and select **Edit Step**.
2. The step details window displays. You can:
   - Record the **Name** and **Description** of the step.
   - Name the **Result Variable**, a variable to receive the resulting value of the step action.
   - Specify the **Action** for the step.

Each step can take one of the types of actions listed in the following table.

Table 9—Process Step Action Types

Object	Usage
Script	Executes a segment of Java BeanShell code that is included in the step.
Rule	Executes a workflow Rule — a block of Java BeanShell code encapsulated in a reusable rule.
Subprocess	Launches another defined workflow, passing control to it until it completes. When you select this option, the list of available subprocesses, workflows of type Subprocess, displays and you are given the option to enable step replication.
Call Method	Calls a compiled java method in the IdentityIQ workflow library, exposed through the standard workflow handler. When you select this option, the list of available methods displays.

Note: For any of these actions, an appropriate value must be specified or selected before the action can be saved.

For example, if Script is selected, a BeanShell script must be entered in the box. If you choose the Subprocess object, a subprocess must be selected from the list. If the value is not specified, the step is saved with no associated action. Developers who use subprocesses must write the subprocesses before they can complete the step definition of steps in the master process.

The **Enable Monitoring** flag on this window turns on metrics tracking for the step. See “Process Metrics Tab” on page 193 for more information on process monitoring and metrics.

Script

Scripts are java BeanShell code that you write in order to execute a desired action. You write scripts directly in the **Source** box in the detail window for the step.

Note: The script examples in this document all show very short java BeanShell code blocks. There is no set length for a script. A script block within the XML can be any length needed to accomplish the required processing. However, long scripts are frequently encapsulated in rules, as discussed in the next section.

Rule

Rules are also blocks of java BeanShell code. Code encapsulated in a rule is available for reuse by other areas of the application that can launch a rule of the same type. Rules created through this window are of type Workflow and can be used by any workflow. When you choose **Rule** as the **Action**, you can select an existing workflow rule from the list or create a new rule in the rule editor. To open the rule editor, Click the **. . .** icon.

Subprocess

Subprocesses are other workflows. You can use subprocesses to subdivide complex processes into smaller segments that can be easily managed and reused by other workflows. Subprocesses are complete workflows that contain a start step, a stop step, and as many processing steps as are needed to complete their activities.

You can enable step replication to enable multiple subprocesses to run to completion at the same time instead of having them run serially. For example, in an approval step, you can launch multiple approval subprocesses, to multiple approvers, that can take an approval all of the way through provisioning instead of the approval step waiting for all approvals to complete before provisioning can begin.

When you enable replication, you must select an item from the main workflow for replication and an argument that is passed to the action containing the replicated item. Only one item can be replicated per step, and all of the items must be passed the same argument. A new subprocess is generated for each item replicated.

Call

The IdentityIQ workflow library contains a set of methods that you launch within a workflow. Methods are exposed through the standard workflow handler that the workflow engine calls every time an action occurs in a workflow. Every workflow has access to the methods in the standard workflow handler. Additional libraries of methods are also available to use in workflows.

Note: When no library list is specified for the workflow, the default includes access to the Identity, Role, PolicyViolation, and LCM libraries.

Through the XML, you can specify other libraries, including custom libraries for an installation. The user interface does not provide an option to manage the library list

Note: Specifying a library list overrides the default. You must explicitly include in the library list any default libraries that contain methods the workflow needs. See “Workflow Element” on page 197 for more details on specifying a library list.

When **Call Method** is selected for the workflow step, **Action**, the method name is selected from the **Call Method** list. The methods in these workflow libraries are listed and briefly described in “Workflow Library Methods” on page 218.

Step Arguments

When arguments need to be passed to the script, rule, subprocess, or library method launched by a step, you must specify the argument on **Arguments** tab for the step. Arguments can be specified in the following ways:

Table 10—Step Argument Specifications

Type	Usage
Basic View	Some steps copied from the step library include a configuration form to simplify the specification of arguments. When a step has a configuration form, this is called the Basic View and is shown by default. The Basic View allows you to set arguments using literal values.
Advanced View	The advanced view gives you more control over how the argument values is calculated.
Return Variables	Each step can return only one result variable, which can be specified through the user interface. When a step has an action that launches a subprocess, you can also use return variables. Multiple values can then be passed back from the subprocess to the main workflow. Because the user interface does not provide a vehicle for declaring return variables, You must specify the return variables directly in the XML. See “Return Variables” on page 170 for more information.

Table 11—Step Argument Specifications

Type	Usage
String	A literal value. For example, the name of an email template to use.
Reference	A reference to one of the workflow's process variables.
Script	A segment of java BeanShell code that returns a value.
Rule	A workflow rule that returns a value. This functions similar to Script except BeanShell is contained within a re-usable rule.
Call Method	A call to a workflow library method that returns a value.

When a script, rule, or library method is used to calculate an argument value, the configuration can be more complex. If the argument definition needs data to be passed in, you can pass the data by:

- Providing all the current values of workflow variables.
  OR
- Declaring the value of step arguments above an argument.

If desired, you can use ordered step arguments instead of workflow variables if the only use for the value is within this step.

For example, when these two step arguments are declared in this order, the method called to populate Identity_mgr can use the value in Identity_name in its processing if needed.

Table 12—Step Argument Example

Argument Name	Value Type	Value Source
Identity_name	Reference	IdentityName
Identity_mgr	Rule	getManagerRule

More on Start and Stop Steps

Similar to other steps, start and stop steps can contain actions that launch scripts, rules, subprocesses, or calls to workflow library methods. By convention, these steps are included in every workflow but are used only to designate a clear starting and ending point for the workflow. These steps are generally empty steps with no action. Occasionally, debugging messages can be printed from these steps to trace workflow progress during development.

Step Icons

When steps are first added through the Process Designer, only three icon types are available: Start, Stop, and Generic Step. A variety of other icons are available. You can use different icons to make it easier to determine the actions each step performs.

To change an icon for a step:

1. Right-click the step icon and click **Change Icon**.
2. Select the desired icon style from the pop-up window.

Approval Steps

Approval steps are a special type of step in IdentityIQ. You can use Approvals to gather data from a user through a work item. In an approval, the user is asked to review a requested action, such as, granting a role to an identity, and then give their approval for the action to be processed.

To create a basic approval through the user interface:

1. Right-click the step.
2. Click **Add Approval**.

Note: A step can contain an action or an approval, but not both. Approval steps are used for approval processing. Approval steps are not used to perform other actions such as scripts, subprocesses, etc.

To edit an approval that exists in a step

1. Right-click the step and click **Edit Approval**.
2. Alternatively, you can choose **Edit Approval** from the Step Details window.

Approvals are flexible and meet a variety of business needs. An approval can be constructed many ways. approvals range from a simple one-person approval to a complex approval process that involves multiple people with different approval modes and notification schemes.

Approval Details

Every approval includes the following fields to be completed on the Details tab for the approval:

Table 13—Approval Step Details

Object	Usage
Name	User-defined name for the approval.
Send	Comma-separated list of process variable names to be sent to the approval.
Return	Comma-separated list of variables names to copy from the completed approval work item back into the workflow.

Table 13—Approval Step Details

Object	Usage
Renderer	JSF (Java Server Faces) include to render the work item details. Not required if using a default renderer.
Mode	Specifies how approval is processed when multiple owners are specified.
Owner	Approver for the approval. Can be more than one Identity name and is specified by string, reference, script, rule, call method. When more than one owner is specified, mode determines how and when the item is submitted to each listed owner. Parallel, parallelPoll, and any modes submit the approval work item to all owners at the same time. Serial and serialPoll modes wait until the first owner completes the approval before submitting to the next approver in the list.
Description	Defines work item description. Shown as the work item Name in the approver's inbox. Set using string, reference, script, rule, call method.

Approval Arguments.

You can set arguments to the approval on the **Arguments** tab. Generally, variables are passed to approval through the send list. However, any arguments that require transformation, through script, rule, or library method, must be sent through an Arg element. Args defined with reserved system names are passed through the Arg element with the reserved name specified. See “Approval Steps” on page 212 for information on reserved system names.

Work Item Configuration

You can specify some details about the notification and escalation/reminder policy for a work item on the **Work Item Configuration** tab. The work item appears in the owner's IdentityIQ inbox and requires their input. If no configuration is specified, the default work item configuration is used.

To change the configuration for the work item

1. Select **Override Work Item Configuration.**
2. To include an electronic signature in the approval step, select **Override Electronic Signature Configuration**.

The following configuration options are available on the **Work Item Configuration** tab:

Table 14—Work Item Configuration Options

Option	Description
Initial Notification Email	To change the notification email template, select the template from the list

Table 14—Work Item Configuration Options

Option	Description
Escalation	Choose an escalation policy: **None**: no escalation**.** **Send Reminders**: allows configuration of reminder options, such as days before first reminder, frequency, email template**.** **Reminders then Escalation**: allows reminder option configuration plus escalation option configuration, such as reminders before escalation, escalation owner rule, escalation email**.** **Escalation Only**: allows configuration of escalation options, such as days before expiration, escalation owner rule, escalation email).

Child Approvals

Use Child Approvals to customize approval processing or presentation for the different sets of identities involved in the approval process. For example, a change in a user's assigned region requires someone in HR sign off and also requires manager approval. Although the approval of the user's own manager is required, any HR individual can completes the sign-off. This type of approval can be created through child approvals.

To create a child approval:

1. Click **Add Child Approval** on the **Details** tab for the parent approval.
2. Click the child approval in the **Approval Children** hierarchy to select it for editing.

To set up the approval described in the example, create two child approvals:

- HR Approval set up — any of the identities who meet the criteria can make the decision for the group
- Manager Approval set up — identity's manager specified as the owner.

Note: The reference variables HRApprovers and identityManager for the example are process variables defined with initialization scripts that retrieve the appropriate sets of Identities.

If either approval requires a custom work item configuration, you can specify the configuration on the **Work Item Configuration** tab for the approval. Work item configurations are inherited by child approvals if configurations are not specifically overridden for the child. If you want a single custom work item configuration for the entire set of approvals, the configuration should be specified on **Work Item Configuration** tab for the parent approval. In this case, the child approvals inherit the parent configuration.

Form Steps

An approval step can also display a form. Forms are a general way to request information from the user and do not necessarily represent an approval. For example, you can use forms to request a missing attribute such as the department name for an identity or ask the requester for more information about why they are making the request.

You can define a form inside the workflow step or you can reference an external form that is shared with other workflows.

To reference an existing form:

1. Right-click the step and click **Add Form**.

2. In the first screen, click **Reference Form**.
3. In the form reference screen, select a form from the table and select the owner who will be shown the form.

To create a custom form for gathering data from a user:

1. Right-click the step and click **Add Form**.
2. In the first screen, click **Create**.
3. In the form editor, specify the general form properties.

Table 15— Form Step Properties

Field	Description
Description	Work item description text displayed on the user's Home Page.
Send	Comma-separated list of process variables to be passed as initial values for the form fields.
Return	Comma-separated list of form fields to copy back into process variables when the work item is closed.
Owner	The identity to be shown the form. Can be a simple identity name, a name stored in a process variable, or a name calculated by a script, rule, or library method.

A form includes one or more fields that define what information you want to show and the information you are asking the user to provide. This form field editor is similar to the field editor for provisioning policies and uses most of the same options.

Table 16— Form Step Values - Bottom

Field Attribute	Description
Name	System-accessible name for field. Used to reference field programmatically.
Display Name	Label that is displayed on form for the field.
Help Text	Tool tip help text for field.
Type	Field type. Impacts rendering of field on form.
Multi-valued	Flag to determine if the field can contain multiple values (multi-selectable).
Read Only	Field displays a value that cannot be changed.
Hidden	Field is not displayed.
Owner	Field owner. Does not apply to form fields.
Required	Value must be entered.
Refresh Form on Change	Form is refreshed when the value for this field is changed. *TIP! This field is useful when the value of a field in the form depends on the value in another field.*
Display Only	Does not apply for workflow forms.
Authoritative	Does not apply for workflow forms.
Value	Literal, script, or rule to set the initial value of the field.

Table 16— Form Step Values - Bottom

Field Attribute	Description
Allowed Values	Allowed values for the field. Displays as a drop-down list box or combo box based on the multi-valued setting.
Validation	Rule or script that validates the value of a field when the form is saved/submitted. Prevents submission if the value is not valid.
Dynamic	Delays the launch of allowed values, scripts, or rules until the field is selected, instead of launching as soon as the form loads.

The form editor also provides the option to specify buttons to include on the form.

To add a button definition:

1. Click **Add Button**.
2. Select the button **Action** and specify a behavior of the button.
3. Specify addition button options as described in the table below, and click **Save**.

Table 17— Button Properties

Function	Description
Action	Select the action the button takes when pressed. Choose from the following actions: **Next** — assimilates form data and advances to the next state, such as OK/Save/Approve/Submit functionality. Sets status of approval to Approved. **Cancel** — Stops form editing, returns to previous page in the user interface, and leaves work item active. **Back** — assimilates form data and returns to the previous state. Sets status of approval as **Rejected** and advances workflows. **Refresh** — Assimilates the posted form data and regenerates the form.Not a state transition. Refresh is a re-display of the form.
Label	Text to display on the button.
Parameter	Name of an optional value to be sent with the form fields when this button is pressed.
Read Only	Non-actionable button.
Skip Validation	Ignores the validation when the form is posted.
Value	Optional value to be sent with the form fields when this button is pressed.

During initial form specification, defined buttons and fields are listed together in the left panel in the order they are added. If some buttons were added before some fields, the button can be intermixed. On the final form, buttons are always grouped together at the end of the form. When the Form Editor is revisited later, the fields are listed together first, in the order they were created, and then the buttons follow in the order they were created.

Note: Buttons can be reordered in the XML to display in a different order on the form.

Custom forms can also be created or edited through XML. Various advanced form options, such as sections, multi-column layout, are only available through the XML. See, “Forms” on page 239 for more information.

Step Conditions

Normally when a transition is made into a step, the step action is executed. In some cases you might want the execution of the step to be optional. You can add a step condition to control whether or not the step action executes. Step conditions can also simplify transition lines in the process because you do not have to create many complex transitions to skip over steps. You can advance from one step to another and let the step conditions determine if the step is executed.

To edit the step conditions:

1. Right-click any process step.
2. Click **Add Step Condition**.
3. Specify addition button options as described in the table below, and click **Save**.

You can express conditions as any of the following:

Table 18—Step Transition Conditions

Type	Description
Reference	Evaluation of a defined process variable. Must be a Boolean variable.
Script	Segment of java code that evaluates process variables.
Rule	Workflow rule that contains a reusable segment of java code to evaluate process variables.
Call Method	Call to launch a Java method in the IdentityIQ workflow library. Exposed through standard workflow handler.

Selecting the **Negate** option changes the evaluation to the opposite condition. For example, if the condition evaluates to False, the negate option changes it to True.

Step Transitions

Steps are connected through Transitions. Transitions can connect one step to the next sequentially. Alternatively, steps can include evaluation statements that enable conditional processing, such as certain data conditions that can cause the workflow to execute Step A versus Step B.

To add a transition do the following:

1. Right-click the process step for starting the transition and select **Start Transition**.
2. Navigate to the process step for ending the transition, right-click and select **End Transition**.
3. Right-click the transition icon and select **Edit Transition** to set the condition.
4. To add additional conditions to this transition, repeat the process.

To edit the transition conditions:

1. Right-click the transition diamond
2. Click **Edit Transitions**.
3. Specify addition button options as described in the table below, and click **Save**.

A step can have as many transitions to next steps as needed. Transition conditions are evaluated in the order they are listed. The first transition that has no condition, or whose condition evaluates to true is taken. Use the up and down arrows in the transitions dialog box to re-order the transitions. As a recommended practice, the final transition should have no condition. That transition is taken when no other transition conditions are met. If a step only has transitions with conditions, and none of the conditions are met, the workflow ends.

Conditions can be expressed as any of the following:

Table 19—Step Transition Conditions

Type	Description
String	Not used. This condition is an artifact of the common structure used for variable setting and does not apply to conditions. A literal value of **True** or **False** can be specified but does not allow any evaluation in the transition. **True** always launches the associated step and **False** always bypasses the associated step.
Reference	Evaluation of a defined process variable. Must be a Boolean variable.
Script	Segment of java code that evaluates process variables.
Rule	Workflow rule containing reusable segment of java code to evaluate process variables.
Call Method	Call to launch a Java method in the IdentityIQ workflow library. Exposed through standard workflow handler.

Transition conditions must evaluate to boolean values. If the value is true, the workflow moves to the step that the transition references. If the value is false, the next transition in the list is evaluated.

Selecting the **Negate** option changes the evaluation to the opposite condition. For example, if the condition evaluates to False, the negate option changes it to True.

Note: Because all processing options should end with the stop step, every workflow should end with a step that transitions to Stop.

Process Metrics Tab

The Process Metrics tab displays the following statistics that are useful for troubleshooting workflows:

- Number of times the workflow launched.
- Number of times the workflow succeeded or failed.
- Average and maximum duration of the workflow.
- Date the workflow last launched.

You can view additional process metrics, including data tracked at the step level, through the **Intelligence -> Advanced Analytics** -> **Process Metrics Search** tab.

To turn on metrics tracking:

1. For individual workflow steps, select **Enable Monitoring** in the Details window.
2. Alternatively, you can right-click on a step and select **Enable/Disable** from the drop-down menu on the step.

To turn on monitoring for all steps in a workflow, click **Monitor** at the bottom of the business process editor window.

Chapter 14: Editing Workflow XML

There are various options for editing workflow XML. You can:

- Create the initial workflow through the user interface and then edit the workflow directly.
- Complete all workflow development in XML.
- Write original XM or use XML from an existing workflow as a template for a new process

All of these methods are valid and can be used as desired.

Accessing the XML

The XML for existing workflows can be viewed and edited through the IdentityIQ Debug pages or can be exported through the IdentityIQ Console.

Debug Pages

To view the XML in the Debug pages, navigate to the Debug pages and Select **Workflow** from the object list to view a list of all defined workflows in the system.

to view the XML representation, click the name of the workflow. From the Debug pages you can edit and save changes. A workflow can also be copied from here and pasted into an external editor of choice.

- View and edit the XML.
- Save changes to the XML.
- Copy and paste the XML to an external editor.

IdentityIQ Console

You can export one or more workflows from IdentityIQ through the console. The console export is the most efficient way to get the XML for all workflows extracted from the system at one time. The IdentityIQ console export command can extract all the Workflow XMLs together into a single file.

After export the XML, you can parse the XML into a separate file for each workflow and save the files in the installation source code control system for later use in system environment migrations or in product upgrade processes.

Table 20—Important Workflow Objects

Object	Usage
Workflow	Defines the workflow structure and steps involved in the workflow processing.
WorkflowCase	Represents a workflow in progress. Contains a Workflow element in which the process is outlined and current state data is tracked, as well as identifying information about the workflow target object.

Table 20—Important Workflow Objects

Object	Usage
WorkflowContext	launchtime information that Workflower maintains as it advances through a workflow case. Passed into rules and scripts and to the registered WorkflowHandler. Contains all workflow variables, step arguments, current step or approval, workflow definition, libraries, and workflowCase.
Task Result	Records the completion status of a task, or in this case, the workflow, contained within the WorkflowCase.

Re-importing the XML

Because the system only launches Workflow XML that is saved within IdentityIQ, XML documents that are edited externally must be re-imported for the changes made to them to take effect.

To re-import an externally saved XML document, use the console import command or from the Import from File page accessed from the Global Settings page.

Dollar-Sign Reference Syntax

You can reference workflow variables inside XML tags and in user interface fields using `$()` notation. These are resolved into their variable values. For example, if a variable identityName is defined and contains the full name of an Identity, for example, John Smith, an Arg specified as:

```
<Arg name="FullIdentityName" value="$(identityName)">
```

passes “John Smith” as the value for the variable **FullIdentityName**.

When the variable is used alone, it functions the same as specifying value="ref:identityName. However, the more common usage is to include the variable in a longer string such as:

```
<Arg name="Title" value="Role Update for $(identityName)">
```

which passes “Role Update for John Smith” as the value for the variable Title.

XML Content

This section describes the elements present in the workflow XML and explains their usage.

Header Elements

The following three lines must be included as shown in any workflow document. The <sailpoint> tag must, of course, be matched with a </sailpoint> tag at the end of the workflow document.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE SailPoint PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<sailpoint>
```

Workflow Element

The Workflow tag identifies the name and type of the workflow.

```
<Workflow  explicitTransitions="true" name="WF-Training Hello World Workflow"
type="IdentityUpdate">
```

The attributes of a workflow element including the following:

Table 21—Workflow Element Attributes

Workflow Attribute	Purpose
configForm	A soft reference to a process variable form presented in the Basic View of Process Variables tab, or a step form presented in the Basic View of the Arguments tab on the Step Editor panel accessed from the Process Designer tab of the Manage Business Process page.
name	Short descriptive name for the workflow this is displayed in user interface selection list-boxes and list of existing business processes on the Process Editor window.
type	Workflow type. Type is used to filter workflow selection lists in configuration windows where you select the workflow based on system activities.
explicitTransitions	Boolean value indicating that transitions between steps are explicitly specified and workflow should not resort to implicit, fall-through, transitions when no transition conditions evaluate to **true**. The default setting is **false.** If you so omit this argument and the specified transition conditions all evaluate to **false**, the workflow uses implicit transitions and launches the next sequential step in the XML. However, if you edit a workflow using the Business Process Editor, the value is changed to **true**. **Note: If the developer makes the last transition in any set unconditional, which is considered best practice, the transitions between steps are smoother.**
libraries	Lists workflow libraries the workflow needs. **Note: If this attribute is not specified, workflows automatically have access to Identity, Role, PolicyViolation, and LCM libraries.**
stepLibraries	Lists workflow step libraries the workflow can access. **Note: If this attribute is not specified, workflows automatically have access to the Generic Step Library, which provides access to the Start, Stop and Generic steps.**
handler	The default workflow handler is sailpoint.api.StandardWorkflowHandler. This attribute does not need to be specified when the default is used. In this case, the best practice is to omit it. **Note: If you use a custom workflow handler, the custom handler must EXTEND the default handler and not replace it. The custom handler must be specified in the workflow Handler argument.**

Variable Definitions

The recommended best practice is to identify all variables for the workflow at the top of the XML document. The variable definitions come next in the XML.

At a minimum, variable elements require a name. Other attributes can indicate the variable type and use, such as input, required, editable, return. A description can be specified for each variable. When needed, an initialization value can also be provided. Using the initialization option is the recommended practice rather than creating separate steps to initialize each variable. Using initialization values is more efficient, easier to read, and easier to debug, because Trace reports initializations as they occur. For more information, see “Initializer Options” on page 199.

```
<Variable input="true" name="project" output="true" required="true">
    <Description>
      Project that has account requests in the QUEUED state.
    </Description>
  </Variable>


<Variable editable="true" initializer="true" name="doProvisioning">
    <Description>Set to true to cause immediate provisioning after the
assignment</Description>
  </Variable>
```

Some parts of the variable definition are expressed within attributes on the Variables element. Other parts are expressed through nested elements of their own.

Table 22—Variable Attributes

Variable Attribute	Purpose
name	Variable name.
type	Variable type. Type declaration is not enforced by the application and is used primarily for documentation.
initializer	Initialization value for the field.
input	Flag indicating that the variable is an argument to the workflow. Omitted if not true.
output	Flag indicating that the variable is a return value for the workflow. Omitted if not true.
required	Flag indicating that the variable is a required field for the workflow. Omitted if not true.
editable	Flag indicating that the variable can be edited by the workflow. Omitted if not true.
Nested Tag within Variable Element	
Description	Provides a description of the purpose for the variable.
Script	Alternative to script in the initializer attribute value. Should be used for initializer scripts of any length or complexity.
Source	Nested within the Script tag and contains the java BeanShell source for the action to be executed.

Initializer Options

The Initializer attribute requires additional attention. When these attributes are set through the user interface, you can specify the attribute as a string, script, rule, call, or reference. The same options are available directly through the XML.

Note: The initializer for a variable is only used when a value for the variable is not passed in to the workflow.

Table 23—Initializer Options

Initializer Type	Description and Examples

string	Assigns a literal value to the variable.
	Note: String is the default initializer option so the "string:" prefix can be included or omitted.
	Examples:
	`<Variable initializer="string:true" name="trace"/>`
	`<Variable initializer="spadmin" input="true" name="fallbackApprover">`

script	Assigns a value based on the results of a Java BeanShell script.
	Examples:
	(1) In-line Script. Only use for very short, simple scripts.
	`<Variable initializer="script:(identityDisplayName != void) ? identityDisplayName : resolveDisplayName(identityName)" input="true" name="identityDisplayName">`
	(2) Script within nested <script> element. Use for most script initializers - scripts of any complexity or length.
	`<Variable initializer="script:resolveDisplayName(launcher)" input="true" name="launcherDisplayName">`
	`<Description>`
	`The displayName of the identity being who started this workflow.`
	`Query for this using a projection query and fall back to the name.`
	`</Description>`
	`<Script>`
	`<Source>`
	`// Lookup the launcher's display name for use in email templates.`
	`String returnString = launcher;`
	`Identity launcherId = context.getObject(Identity.class, launcher);`
	`if (null != launcherId) {`
	`returnString = launcherId.getDisplayName(); // First+Last`
	`}`
	`return returnString;`
	`</Source>`
	`</Script>`
	`</Variable>`

Table 23—Initializer Options

Initializer Type	Description and Examples
rule	Assigns a value based on the return value of a workflow Rule. Examples: `<Variable initializer="rule:wfrule_GetIdentityName" name="IdentityName">`
call	Assigns a value based on the return value of a call to a workflow library method. Example: `<Variable initializer="call:getObjectName" name="roleName">`
ref	Assigns a value based on a reference to another workflow variable. This type is rarely used. Example: `<Variable initializer="ref:otherVar" name="myVar"/>`

Workflow Description

A description element should be included to describe the purpose of the workflow. Although the description element is not used in the workflow process, it is recommended for usability. In the user interface, the contents of this element are displayed on the Process Details tab of the Business Process page. This element should be included near the top of the workflow, either before or after the variable definition section.

```
<Description>
     Workflow called when a role is ready to be enabled.
</Description>
```

Rule Libraries

Some methods the workflows use are grouped together into Rule Libraries. These Rule Libraries are defined as rules in IdentityIQ. However, these libraries contain sets of related but unconnected methods that workflow steps can directly within a script action. Because the rule methods are in rules, rather than in the compiled Java classes, their functionality can be easily modified to meet the needs of each installation. To make the methods within one of these rules available to steps within the workflow, the RuleLibraries element must be declared. See the following example.

Note: Each Reference element applies to one library. Include only the libraries that contain the required methods in the RuleLibraries declaration for the workflow.

```
<RuleLibraries>
    <Reference class="sailpoint.object.Rule" name="Workflow Library"/>
    <Reference class="sailpoint.object.Rule" name="Approval Library"/>
    <Reference class="sailpoint.object.Rule" name="LCM Workflow Library"/>
</RuleLibraries>
```

Note: You can create and reference custom libraries using this same syntax.

Step Libraries

Step libraries are designed to offer a group of common functions that can be added to existing workflows from the **Add a Step** panel Business Process Editor. Step libraries are a collection of steps encapsulated by a workflow with the template attribute marked true. The steps do not have any transitions and they are not executable. A Step Library must be defined. See the following example.

Note: The type does not have to be StepLibrary. However, using the StepLibrary type ensures that these workflows do not appear in other parts of the product.

```
<Workflow name="Provisioning Step Library"
                  type="StepLibrary"
                  template="true">
```

When you edit a new or existing workflow, you can include a list of step libraries by including a comma separated list in the stepLibraries attribute. See the following example.

```
<Workflow name="LCM Provisioning"
                  type="Provisioning"
                  taskType="LCM"
                  libraries="Identity,Role,PolicyViolation,LCM,BatchRequest"
                  stepLibraries="Common,Provisioning"
                  handler='iiq.api.StandardWorkflowHandler'>
```

In the example above, when you edit a business process with the LCMProvisioning type, the Common and Provisioning steps are available in the **Add a Step** panel of the Business Process Editor.

Steps within a step library workflow can also include a soft reference to a step form that provides a simplified form-based interface that you can use to add arguments to some steps in the workflow. This form-based interface adds a Basic view option to the **Arguments** tab of the Step Editor. The Basic view is built using the information contained in the referenced form. The Advanced view is a list of all possible arguments and is built using the list of arguments that the step library references.

When you add a step form reference to a step library, use the configForm attribute, See the following example.

```
<Workflow name="Provisioning Step Library"
                  template="true"
                  type="StepLibrary">

 <Step configForm="Provisioning Approval Step Form"
                  icon="Task"
                  name="Account Approval">
<Arg name="approvalMode"/>

<Arg name="approvalScheme"/>

<Arg name="approvalSet" value="ref:approvalSet"/>
...
```

In the example above, when you edit an approval step in the Step Editor, the Basic and Advance Views of the Arguments tab are displayed.

Built-in Steps

IdentityIQ includes several built-in steps. The **Start**, **Stop**, and **Generic** steps apply to all workflow types. The following table lists the names, descriptions, and associated workflow types of additional built-in steps.

Table 24—Included Business Process Steps

Step	Description	Process Type
Notify	Allows users to select categories of recipients to notify, the specific recipient, recipients for each category, and the specific email template to use for each category.	Identity Lifecycle LCM Provisioning
Account Approval	Used for provisioning request approvals. The process assumes many of the Provisioning Workflow structures exist.	Identity Lifecycle LCM Provisioning

Step Elements

The core of the workflow is contained within the step elements. At a minimum, a step should contain:. The action attribute determines what processing the step performs. Steps usually contain one or more nested <Transition> elements and ideally also contain a nested <Description> element that tells the reader what the step is intended to do.

- an icon
- name
- posX attribute
- posY attribute

The action attribute determines what processing the step performs. Steps usually contain one or more nested <Transition> elements and ideally also contain a nested <Description> element that tells the reader what the step is intended to do.

```
<Step icon="Start" name="Start" posX="250" posY="126">
    <Description>
        The workflow's processing starts with this step.
    </Description>
    <Transition to="Initialize"/>
  </Step>
```

Note: Similar to variables, some parts of a step definition are included as attributes of the step and others are expressed as nested elements within the step.

Table 25—Step Element Attributes

Step Attribute	Purpose
configForm	A soft reference to the form that is presented to the Basic View of the Arguments tab on the Step Editor panel.
name	Short but descriptive name for step displayed in user interface graphical display below the step icon.
icon	Icon to display for the step in the user interface graphical Process Designer. Valid icon values include: Start, Stop, Default (Generic Step), Analysis (Launch Impact Analysis), Approval, Audit, Catches, Email, Message (Add Message), Provision, Task (Launch Task), and Und
posX, posY	X and Y indicate positions where the step icon should be displayed on the user interface graphical Process Designer grid. **Note: If you omit the posX and posY values, the icon is displayed at the top right of the grid. You can drag the icon around to create the desired layout at a later time.**
action	The processing action to take for the step, such as a script, rule, subprocess, or call. See “Step Actions” on page 207.
wait	Pauses the action for a specified duration, see “Wait Attribute” on page 211.
catches	Causes the step to be launch when **Complete** status is caught, rather than through a transition from another step. See “Catches Attribute” on page 211.
resultVariable	Variable name that contains the return value from the step.
Nested Tag within Step Element	
Description	Provide a description of the step purpose.
Transition	Identifies the next step the process moves to when the current step is complete, see “Transition Element” on page 204.
Arg	Passes variables to the step. Used for steps that require data to be passed in to them.
Return	Receives return values from subprocess steps, see “Return Elements” on page 209.
Script	Alternative to script in the Action attribute for the step. Use these step attributes for action scripts of any length or complexity.
Source	Nested within the Script tag and contains the java BeanShell source for the action to execute.

Transition Element

The transition element indicates the name of the next step the process executes following completion of the current step and is always nested within a step in the model. Transitions can contain conditions based on a string, script, rule, call method, or reference (similar to a variable initialization). The return value for conditions must be a Boolean (True/False). When multiple transitions are stipulated, they are evaluated in the order they are listed, and the transition for the first condition met is followed. The last transition in the list should, as a best practice, not contain any conditions so it can be used as the default action.

Transitions contain two attributes:

- to — next step
- when — condition for progressing to the next step

When a script is evaluated as the condition for a transition, it is often specified through these nested elements instead of as a **when** attribute on the transition element, especially if you use a long script.

Table 26— Nested Tag Within Transition Element

Nested Tag Within Transition Element	Purpose
Script	Alternative to script in the transition **when** attribute. The script should be used for scripts of any length or complexity.
Source	Nested within the Script tag. This tag contains the BeanShell source for the condition evaluation.

Example:

```
<Transition to=”end”>
    <Script>
       <Source>
            (“cancel”.equals(violationReviewDecision) || ((size(policyViolations)
             > 0 ) &amp;amp;&amp;amp; (policyScheme.equals(“fail”))))
       </Source>
    </Script>
</Transition>
```

Conditions in the **when** attribute can be specified using the following types of conditions:

Table 27—Transition Element Conditions

Condition Type	Description and Examples
string	Not used.This condition type is an artifact of the common structure used for variable setting and does not apply to conditions. A literal value of **True** or **False** can be specified. However, using one of those literal values does not enable any evaluation in the transition. **True** always executes the associated step and **False** always bypasses the step.

Table 27—Transition Element Conditions

Condition Type	Description and Examples

script	Evaluates script result value to determine step transition. Very short scripts are specified inline on the transition element, within the **when** attribute. Longer scripts are expressed within nested <script> and <source> elements.
	
	Note: Because script is the default transition when option, the "script:" prefix can be included or omitted.
	
	Examples:
	
	(1) In-line Script. Use only for very short, simple scripts.
	
	`<Transition to="Exit On Policy Violation"`
	`  when="script:((size(policyViolations)> 0)`
	
	`  &amp;&amp; (policyScheme.equals(&quot;fail&quot;)))"/>`
	
	(2) Longer script within nested <script> tag should be use for transition scripts of any complexity or length.
	
	`<Transition to="end">`
	`  <Script>`
	`  <Source>`
	`  ("cancel".equals(violationReviewDecision) ||`
	`((size(policyViolations)`
	`  > 0 ) &amp;&amp; (policyScheme.equals("fail"))))`
	`  </Source>`
	`  </Script>`
	`</Transition>`

rule	Evaluates the return value of a workflow rule to determine step transition.
	
	Examples:
	
	`<Transition to="Process Approval"`
	`when="rule:RequireApprovalRule">`

call	Evaluates return value of a call to a workflow library method to determine step transition. Example: `<Transition to:"Check Status" when="call:requiresStatusCheck" />`
ref	Evaluates a defined, Boolean, workflow variable to determine step transition. Example: `<Transition to="Refresh Identity" when="ref:doRefresh"/>`
Unconditional	Specified as last transition option to give a default path for the transition. Example: `<Transition to="Approve"/>`

Step Actions

Most steps involve much more than a name and a transition. Steps also include an action attribute that executes the workflow processing. The action of a step can be a script or can a rule, subprocess, or a call to a workflow library method.

Table 28—Step Actions

Action Type	Description

Script	Similar to scripts in other parts of the workflow XML, the script can be contained within the action attribute or can be nested within the Step in a <Script> block.
	
	Examples:
	
	(1) In-line Script, used only for very short, simple scripts.
	
	`<Step action="script:approvalSet.setAllProvisioned();" icon="Task" name="Post Provision">`
	`  <Transition to="Stop"/>`
	
	`</Step>`
	
	(2) Longer script within nested <script> tag. Used for action scripts of any complexity or length.
	
	`<Step name="Start" icon="Start" posX="20" posY="20">`
	`  <Script>`
	`  <Source>`
	`  String wfName = wfcontext.getWorkflow().getName();`
	`  System.out.println("Starting workflow: [" + wfName + "]");`
	`  </Source>`
	`  </Script>`
	`  <Transition to="Compile Provisioning Project"/>`
	`</Step>`

Rule	A step can execute a block of Java BeanShell code encapsulated in a reusable workflow Rule. Example: `<Step action="rule:WFRule_verifyIdentity" icon="Task" name="Verify Identity" posX="600" posY="202">`

Table 28—Step Actions

Action Type	Description

Subprocess	When you include a <WorkflowRef> element within the step and reference the SailPoint**.object.Workflow** class and the specific workflow by name, a subprocess is defined.
	
	Example:
	
	`<Step icon="Task" name="Initialize" posX="320" posY="126">`
	`  …`
	`  <WorkflowRef>`
	`  <Reference class="sailpoint.object.Workflow" name="Identity Request Initialize"/>`
	`  </WorkflowRef>`
	`  <Transition to="end">`
	`</Step>`

Call	Calls to workflow library methods can be used to do step processing. **Note: Call is the default action option. Therefore the "call:" prefix can be included or omitted.** Example: `<Step action="call:refreshIdentity" icon="Task" name="Refresh Identity" posX="618" posY="242">`

Arguments

Any variables to be passed to a script, rule, subprocess, or library method must be declared as step arguments through <Arg> elements. Similar to other variables, the values for arguments can be specified by string, script, rule, call, or reference. The default specification type is string. Therefore, the “string:” qualifier can be omitted. However, arguments are also commonly passed by referencing workflow variables.

```
Step icon="Task" name="Initialize" posX="320" posY="126">
    <Arg name="w" vaflolue="ref:flow"/>
    <Arg name="formTemplate" value="string:Identity Update"/>
    <Arg name="identityName" value="ref:identityName"/>
     …
    <Description>Call the standard subprocess to initialize the request,
       this includes auditing, building the approvalset, compiling the plan into
       project and checking policy violations.</Description>
    …
    <WorkflowRef>
        <Reference class="sailpoint.object.Workflow" name="Identity Request
           Initialize"/>
    </WorkflowRef>
    <Transition to="end">
```

```
</Step>
```

When an argument is specified as a script, rule, or call, for example, <Arg name="myVar" value="rule:myWFRule"/>, any needed arguments to the script, rule, or library method cannot be explicitly specified.

Because these scripts, rules, and library methods automatically have access to the workflow context object, the scripts can access workflow variables directly through the workflow context get methods. These scripts/rules/methods can also access any step arguments that were defined before them in the step declaration. For example, the method that identifies the value for the Manager argument can use the value in the identityName argument in its processing, if needed. See the following example.

```
<Step icon="Task" name="Processing Step" posX="320" posY="126">
    <Arg name="identityName" value="ref:identityName"/>
    <Arg name="Manager" value="call:getManager"/>
    …
</Step>
```

The following table lists the available Arg attributes

Table 29—Available Arg Attributes

Arg Attribute	Purpose
name	Variable name in process to which the data is being passed.
value	Value to pass into the variable, such as string, script, rule, call, reference.

Return Elements

To return more than one value from a subprocess, you can declare <Return> elements for the step. At a minimum, a return element contains: a **name** attribute and a **to** attribute. The name attribute is the name of the variable in the subprocess workflow and the **to** attribute is the variable name in the calling (current) workflow. If these names are the same in both workflows, a **to** attribute is not required. However, specifying a to attribute is a best practice for clarity.

Use the merge attribute when the variable is a List and the returned values should be appended to the current workflow's list instead of replacing it. Similar to Args, value attribute for return elements can be specified as a string, script, rule, call, or reference. String is the default. If the value argument is omitted, the value of the name variable is copied as-is into the to variable, However, a script/rule/method can be used to transform or modify the value as it is passed.

- **name** attribute — name of the variable in the subprocess workflow
- **to** attribute — variable name in the calling (current) workflow

Note: If these names are the same in both workflows, a to attribute is not required. However, specifying the to attribute is best practice.

```
<Step icon="Task" name="Initialize" posX="320" posY="126">
    <Arg name="flow" value="ref:flow"/>
    <Arg name="formTemplate" value="string:Identity Update"/>
    <Arg name="identityName" value="ref:identityName"/>
     …
    <Return name="project" to="project"/>
```

```
    <Return merge="true" name="workItemComments" to="workItemComments"/>
    <WorkflowRef>
        <Reference class="sailpoint.object.Workflow" name="Identity Request
           Initialize"/>
    </WorkflowRef>
    <Transition to="end">
</Step>
```

The following table lists the available Return attributes.

Table 30—Available Return Attributes

Return Attribute	Purpose
name	Variable name in process from which the data is returned.
to	Variable name in the workflow step to which the data is passed.
value	Value to pass into the variable, such as string, script, rule, call, reference.
merge	Flag indicating that the value should be merged into the target variable instead of replacing the variable. This attribute is used for list variables.
local	Only applies to returns on Approvals, see “Approval Steps” on page 212. A flag that indicate the value is passed to local storage within the parent approval and not passed to a workflow case variable. This attribute is used for complex approvals where a work item state is saved for later analysis in a script.

Call

Use calls to workflow library methods to do step processing. Similar to subprocesses, they sometimes require arguments to be passed to them. You declare method arguments the same way as subprocesses. You use Library methods with a call action. See the following example.

```
<Step action="call:refreshIdentity" icon="Task" name="Refresh Identity" posX="618"
posY="242">
    <Arg name="identityName" value="ref:identityName"/>
    <Arg name="correlateEntitlements" value="string:true"/>
   <Description>Add arguments as necessary to enable refresh features. Typically you
      only want this to correlate roles. Don't ask for provisioning since that
      can result in provisioning policies that need to be presented and it's
      too late for that. This is only to get role detection and exception
      entitlements in the cube.</Description>
    <Transition to="Notify"/>
  </Step>
```

The methods available for the call action are those included in the libraries attribute for the workflow element, if specified. If no libraries attribute is specified, the workflow automatically has access to the methods in the Identity, Role, PolicyViolation, and Lifecycle Manager libraries. If other libraries, including custom libraries, are explicitly listed in the libraries attribute, any of the default libraries whose methods are needed by the workflow

must also be explicitly included in the list to be available. See “Workflow Library Methods” on page 218 for details about the methods available in each library.

Note: Installations can create custom libraries for commonly used and required business methods. However, custom library methods must be named with unique names that do not conflict with standard library method names. Conflicts resolve as a reference to the standard library method. It is possible to extend a standard library and overload its method names. Extending a standard library is not consider a best practice. Therefore, the best practice is to create new names for nonstandard methods. Creating new names makes it clear that the method is not a standard method.

Wait Attribute

The step wait attribute causes the workflow to pause in its execution for the duration specified. The wait value can be specified as a string, script, rule, call, or reference. String is the default.

```
<Step name="Wait for next check" wait="ref:provisioningCheckStatusInterval">
    <Description>
       Pause and wait for things to happen on the PE side.
       Use the configurable interval to determine how long
       we wait in between checks.
    </Description>
    <Transition to="CheckStatus"/>
  </Step>
```

This attribute creates a special type of step with the sole purpose of creating a pause in the action. Wait steps are commonly used in re-try logic to enable behind-the-scenes processing to occur before the workflow attempts to repeat an action.

Catches Attribute

These steps are not caused through a transition from a previous step. These steps are caused by a thrown message that the steps intercepts or catches. Currently, only a complete message is thrown and can be caught. This process occurs when one of the following items occurs:

- All sequential steps in a workflow are executed to completion.
  OR
- Failure condition results in the termination of the workflow.

```
<Step catches="complete" icon="Task" name="Finalize">
    <Arg name="project" value="ref:project"/>
    <Arg name="approvalSet" value="ref:approvalSet"/>
    <Arg name="trace" value="ref:trace"/>
    <Description>
       Call the standard subprocess that can audit/finalize the request.
    </Description>
    <WorkflowRef>
      <Reference class="sailpoint.object.Workflow" name="Identity Request Finalize"/>
```

```
</WorkflowRef>
<Transition to="end"/>
```

The primary purpose of these steps is to update the IdentityRequest object, which tracks and reports the status of a LifecycleManager request, making the history of LCM request processing available even after the TaskResult for the workflow was purged.

Each installation can drive custom logic based on catching this complete message.

Approval Steps

Approval is one of the most common actions that a workflow process performs. The IdentityIQ Approval model is constructed to simplify the process of defining an approval structure. Approvals are a special type of step that contain an <Approval> element, specifying how the approval work item is presented for approval.

Some approval steps are designed to get a user's approval on a requested change, as the name implies. However, the approval element can be used any time data needs to be gathered from a user.

Typically, when you use approval steps to gather non-approval data, you use a custom form to:

- Present the work item to the user
  and
- Request the needed information from the user.

For information on creating approval steps, see the section above. Through the XML, the custom form is manually defined within an approval step. You can also specify custom forms for traditional approvals when you need to present the information differently than the standard approval forms layout. See"Workflow Forms" on page 236 for more details on usage of custom forms.

Similar to other Workflow elements, you specify some modifiers as attributes on the approval element and specify other modifiers through nested elements within the approval.

Table 31—Approval Step Attributes

Approval Attribute	Purpose
mode	Specifies how an approval is processed. Mode can be determined from string, script, rule, call, or reference String is the default. The user interface only supports the selection of a string of one of the values listed below.The XML also enables reference to a process variable containing one of those values or the specification of a script, rule, or method call that can determine one of those values programmatically. Valid values are: **serial** - approvers are specified in order and the item is passed to each approver in that order. If any approver in the chain rejects, the item is rejected. **serialPoll** - approvers are specified in order and item is passed to each approver in that order. Data is collected on approvals and rejections. However, if one approver rejects, does not necessarily result in the item being rejected. The action decision is expected to be specified in AfterScript logic. **parallel** - item is sent to all named approvers at one time. The item is rejected if any approver rejects it. **parallelPoll** - item is sent to all named approvers at one time. Data is collected on approvals and rejections but rejection by one does not mean rejection of item. The action decision is expected to be specified in AfterScript logic. **any** - item is sent to all named approvers at one time. The first approver to respond makes the decision for the group.
owner	One or more approvers can be specified by string, script, rule, call, or reference. STring is the default. The mode determines how and when the item is submitted to each listed owner when more than one is specified.
renderer	JSF include to render the work item details.
return	Comma-separated values (CSV) list of variable names to copy from completed work items back into workflow.
send	CSV list of variable names to include in the work items.
description	Defines work item description. For nested approvals, child approvals use the work item defined by the parent approval unless the child approval defines its own work item. You can set the description by string, script, rule, call, or reference String is the default.
validation	Used to validate any information the user entered during the approval. This attribute can be specified as string, script, rule, call, or reference. Script is the default. You generally use a nested validationScript element instead of a validation argument.
Nested Tag within Approval Element	

Table 31—Approval Step Attributes

Approval Attribute	Purpose
AfterScript	Provides instructions for additional processing to be done on the item after the approval is complete, and only if approved. Often uses methods in the Approval Rule Library and LCM Workflow Rule Library. If those methods are to be used, the rule libraries must be explicitly included in the workflow using the <RuleLibraries> element. **Note: ParallelPoll and serialPoll items always execute this script after all responses are collected. With either of these modes, the logic in this script should aggregate the results and determine if the item should be approved or rejected. The business determines the criteria for approval or rejection, for example majority rule, any approval=approval, etc.** In either poll mode, the AfterScript is inherited by child approvals if one is not specified.In other modes, child approvals do not inherit the after script.
InterceptorScript	This script is more complex than the AfterScript and is used less often. The script is called in several places in the approval processing: at the approval start, pre-Assimilation, post-Assimilation, when the work item is archived, and at the end of the approval. The stage of the processing is passed to the script as an argument called method that can be used to determine what the script should do at that time. The workflow context's args are also passed to the script. Method values for conditional analysis within InterceptorScript logic: startApproval preAssimilation postAssimilation archive endApproval**If an InterceptorScript and AfterScript exist, the InterceptorScript postAssimilation logic launches before the AfterScript.**
validationScript	Script to perform validation on the work item. For example, you can use this script to validate any data the user enters on the approval before the data is assimilated. This script is inherited by any child approvals.
Source	Nested within the AfterScript, InterceptorScript, and validationScript tags and contains the java BeanShell source for the script.

Table 31—Approval Step Attributes

Approval Attribute	Purpose
Arg	Arguments available to the approval action. Specified by string, script, rule, call, or reference. Most variables are passed to approval through send list. However, args that require any transformation must be sent through an Arg element. Additionally, the following args defined with reserved system names are passed through the Arg element with that name specified: workItemRequester workItemDescription workItemType workItemTargetId workItemTargetName workItemTargetClass workItemDisableNotification workItemNotificationTemplate workItemEscalationTemplate workItemReminderTemplate workItemEscalationRule workItemEscalationStyle workItemHoursTillEscalation workItemHoursBetweenReminder workItemMaxReminders workItemPriority workItemIdentityRequestId workItemArchive
Return	Return value defines how things should be assimilated from a work item back into the workflow case. This attribute is an alternative to the return attribute CSV of variables. It is more complex and also more powerful. This attribute is rarely used in approvals. It is most often used when returning an approval work item variable to a workflow variable of a different name or when you need to transform the variable contents of a work item with a script. The use of these types of return elements follows the same rules as step returns from steps that subprocesses, with addition of local attribute options. See, “Return Elements” on page 209.

The following basic approval step example presents an account change to the identity's manager for approval. The AfterScript records the approval decision and creates an audit record.

```
<RuleLibraries>
    <Reference class="sailpoint.object.Rule"  name="Approval Library"/>
    <Reference class="sailpoint.object.Rule"  name="LCM Workflow Library"/>
</RuleLibraries>

<Step icon="Approval" name="Manager Approval">
<Approval mode="serial" owner="script:getManagerName(identityName, launcher,
fallbackApprover);" renderer="lcmWorkItemRenderer.xhtml"
send="approvalSet,identityDisplayName,identityName,policyViolations">

<Arg name="workItemDescription" value="Manager Approval - Account Changes for User:
$(identityDisplayName)"/>
<Arg name="workItemNotificationTemplate" value="ref:managerEmailTemplate"/>
<Arg name="workItemRequester" value="$(launcher)"/>

<AfterScript>
        <Source>

          import sailpoint.workflow.IdentityRequestLibrary;
          assimilateWorkItemApprovalSet(wfcontext, item, approvalSet);
IdentityRequestLibrary.assimilateWorkItemApprovalSetToIdentityRequest(wfcontext,
approvalSet);
auditDecisions(item);

</Source>
      </AfterScript>

</Approval>

    <Description>
      If approvalScheme contains manager, send an approval for all
      requested items in the request. This approval will get the entire
      approvalSet as part of the workitem.
    </Description>

<Transition to="Build Owner ApprovalSet"
    when="script:isApprovalEnabled(approvalScheme, &amp;quot;owner&amp;quot;)"/>
<Transition to="Build Security Officer ApprovalSet"
    when="script:isApprovalEnabled(approvalScheme, &amp;quot;securityOfficer&amp;quot;)"/>
```

```
<Transition to="end"/>


  </Step>
```

Note: In the AfterScript in this example, the methods not qualified by the library name are in the LCM Workflow Rule Library that is available to the workflow through the <RuleLibraries> declaration.

The assimilateWorkItemApprovalSetToIdentityRequest method is part of the IdentityRequestLibrary, this is available to the script through the import of that library in the script.

Library methods called through step action attributes are available through the workflow libraries attribute list,. However, when the library methods are executed from within scripts, the library must be specifically imported for the script.

Nested Approvals

Child approvals created through the user interface are expressed as nested approval elements in the XML.When nested approvals exist, the parent ceases to be an approval of its own.In those case, the sole purpose of the parent approval is to organize and contain the child approvals. The mode on the parent determines how to process the set of peer child approvals.

```
   <Approval mode="string:parallel" name="Approve Region" owner="ref:regionApprover"
        send="identityName,region">
       <Arg name="workItemDescription" value="string:Approve Region for
$(identityName)"/>
       <Approval name="childApproval1" owner="string:Walter.Henderson"
           send="identityName,region"/>
       <Approval name="childApproval2" owner="string:Alan.Bradley"
           send="identityName,region"/>
    </Approval>
```

In the example above, childApproval1 and childApproval2 are processed in parallel. Because both of these child approvals are identical (no custom work item config and no children of their own), the same objective can be accomplished with a single approval with multiple owners:

```
   <Approval mode="string:parallel" name="Approve Region"
        owner="string:Walter.Henderson,Alan.Bradley" send="identityName,region">
       <Arg name="workItemDescription" value="string:Approve Region for
$(identityName)"/>
   </Approval>
```

Nested approvals can be used effectively when different approval levels are implemented with custom configurations and specifications. For example, the workItemConfig for each of the child approvals can be different, which can result in a notification scheme, escalation policy, etc. for the different approvers.

Nested approvals can be governed by a different approval mode from the one used on the master set and/or can contain their own child approval levels. One child approval can be done as an any approval, one that accepts the ruling of the first responder of several listed approvers, while the highest approval level is managed serially. Another child approval can implement custom workItemConfigs for its own child approvals. The example below illustrates all of these concepts.

Nested approvals can be used effectively when different approval levels are implemented with custom configurations and specifications. For example, the workItemConfig for each of the child approvals can be. The following example that illustrates all of these concepts.

```
   <!-- Approval submitted to HR and to supervisor and manager in serial manner -->
   <Approval mode="string:serial" name="Approve Region" owner="spadmin"
       send="identityName,region">
      <Arg name="workItemDescription" value="string:Approve Region for
$(identityName)"/>

      <!-- HR Personnel approve region (whoever responds first makes decision) -->
      <Approval name="HRApproval" mode="string:any"
           owner="ref:HRApprovers" send="identityName,region"/>

      <!-- Supervisor and Manager approve region serially after HR approves -->
     <!-- Each has a different email template (work item config) for notification -->
    <Approval mode="string:serial" name="SupMgrApproval" send="identityName,region">
         <Approval name="Supervisor" send="identityName,region" owner="Tom.Jones">
           <WorkItemConfig escalationStyle="none">
              <NotificationEmailTemplateRef>
                 <Reference class="sailpoint.object.EmailTemplate"
                   name="SupervisorApprovalEmail"/>
              </NotificationEmailTemplateRef>
           </WorkItemConfig>
         </Approval>
         <Approval name="Manager" send="identityName,region" owner="Mary.Peterson">
           <WorkItemConfig escalationStyle="none">
              <NotificationEmailTemplateRef>
                 <Reference class="sailpoint.object.EmailTemplate"
name="ManagerApprovalEmail"/>
              </NotificationEmailTemplateRef>
           </WorkItemConfig>
         </Approval>
      </Approval>
   </Approval>
```

This ability to nest approvals, with options to assign different approval modes and work item configurations to each, enables implementers to create highly customized approval structures to meet the needs of the installation.

Workflow Library Methods

Workflow Libraries are sets of compiled java methods. To be accessible to workflows, these libraries must be specified as a comma separated list in the libraries attribute of the workflow element. The classes for libraries are named as follows: SailPoint**.workflow.[library]Library.class**. Only the [library] portion is specified in the libraries attribute.

The following example makes methods from the SailPoint**.workflow.IdentityLibrary.class** accessible to the workflow.

```
Example:

<Workflow libraries="Identity" explicitTransitions="true" name="Hello World
Workflow" type="IdentityUpdate">
```

Note: If no Libraries attribute is specified on the Workflow element, the workflow can access the Identity, Role, PolicyViolation, and LCM libraries by default.

The following table lists the workflow libraries and the methods available. Although the Standard Workflow Handler is not technically a library, the methods in it are accessible to every workflow and are called through the same syntax as library methods.

Standard Workflow Handler

Table 32—Standard Workflow Methods

Method / Usage	Description	Expected Args *=required
Object getProperty(WorkflowContext wfc)	Returns value of the named system property.	name*
public Object isProperty(WorkflowContext wfc)	Returns true if the named system property has a value.	name*
public Object getMessage(WorkflowContext wfc)	Returns localized message for use in task results	message* type (severity) arg1-arg4 (up to 4 parameters for the message)
public Object addMessage(WorkflowContext wfc)	Adds message to the workflow case.	message* type (optional severity) arg1-arg4 (up to 4 parameters for the message)
public Object addLaunchMessage(WorkflowContext wfc)	Adds message to workflow case that is displayed in the user interface. Not kept in task result. For example, **Request was submitted**.	message* type (optional severity) arg1-arg4 (up to 4 parameters for the message)
public Object setLaunchMessage(WorkflowContext wfc)	Replaces previously added launch message with a new message based on new state.	message* type (optional severity) arg1-arg4 (up to 4 parameters for the message)
public Object log(WorkflowContext wfc)	Sends something to log4j.	message* level*
public Object print(WorkflowContext wfc)	Prints text to the console.	message*

Table 32—Standard Workflow Methods

Method / Usage	Description	Expected Args *=required
public Object audit(WorkflowContext wfc)	Creates an audit event. Enables workflows to put custom entries in audit log, which displays in the user interface.	source* action* target string1 - string4
public Object sendEmail(WorkflowContext wfc)	Sends an email message.	to* cc bcc from subject body template* templateVariables sendImmediate exceptionOnFailure
public Object launchTask(WorkflowContext wfc)	Launches a defined task.	taskDefinition* taskResult sync (true=synchronous execution)
public Object scheduleRequest(WorkflowContext wfc)	Launches a generic event request.	requestDefinition* requestName (name to assign to request) scheduleDate scheduleDelaySeconds owner

Table 32—Standard Workflow Methods

Method / Usage	Description	Expected Args *=required
public Object scheduleWorkflowEvent(WorkflowContext wfc)	Launches a workflow event request.	requestName (name to assign to request) scheduleDate scheduleDelaySeconds owner workflow* (name of workflow to launch) caseName (optional case name to override default)
public Object commit(WorkflowContext wfc)	Commits a transaction. Not commonly needed in workflows. Most commonly used for role approvals.	creator archive
public Object rollback(WorkflowContext wfc)	Rolls back a transaction. Not commonly needed in workflows. Most commonly used for role approvals.	none

Identity Library

Table 33—Identity Library Methods

Method / Usage	Description	Expected Args *=required
public String getManager(WorkflowContext wfc)	Returns the name of the manager for the specified identity.	identityName
public Object calculateIdentityDifference(WorkflowContext wfc)	Derive a simplified representation of the changes made to an identity for an approval work item.	oldRoles newRoles plan approvalSet

Table 33—Identity Library Methods

Method / Usage	Description	Expected Args *=required
private void addLinksInformation(WorkflowContext wfc)	Modifies workflow context lists of links (accounts) to be added, moved, or removed for the identity as a result of the provisioning plan.	linksToAdd linksToMove linksToRemove plan
public List<Map<String,Object>> checkPolicyViolations(WorkflowContext wfc) Evaluate policy violations that can be incurred by the provisioning plan/project's actions	Evaluates policy violations that the provisioning plan/project actions can incur.	policies identityName* project plan (either plan or project is required)
public void activateRoleAssignment(WorkflowContext wfc)	Assigns a role or roles to the identity.	identity* (ID) role* (ID) detected (Boolean indicating if role was detected vs. assigned)
public void deactivateRoleAssignment(WorkflowContext wfc)	Removes role assignments from the identity.	identity* (ID) role* (ID) detected (Boolean indicating if role was detected vs. assigned)
public void refreshIdentity(WorkflowContext wfc)	Performs an identity refresh on one identity.	identity (ID) identityName (either identity or identityName is required)
public void refreshIdentities(WorkflowContext wfc)	Performs an identity refresh on a set of identities. Can specify one or more identityNames, a filterString, or a list of roles. Processes the first of the above listed options that is non-null.	identityName identityNames (CSV) filterString identitiesWithRoles (CSV) (any one of these 4 is required)
public Object compileProvisioningProject(WorkflowContext wfc)	Compiles a provisioning plan into provisioning project.	plan identityName

Table 33—Identity Library Methods

Method / Usage	Description	Expected Args *=required
public Object buildProvisioningForm(WorkflowContext wfc)	Creates a form to display provisioning policy questions. When requiredOwner is passed as an argument, a form owned by this user is returned. If no more forms for this user exist, null is returned. When preferredOwner is passed as an argument, a form owned by this user is returned. If there are no remaining forms for that owner, a form owned by a different user can be returned.	project* template (name of form to serve as page template) owner preferredOwner (owner or preferredOwner required but mutually exclusive)
public Object assimilateProvisioningForm(WorkflowContext wfc)	Collects data from completed a provisioning form and stores answers with questions on provisioningProject.	project* form*
public Object assimilateAccountIdChanges(WorkflowContext wfc)	Updates ApprovalSet with any changes to accountIDs.	project* approvalSet
public Object buildPlanApprovalForm(WorkflowContext wfc)	Builds a form that represents all attributes in a provisioningPlan for an approval before the provisioning occurs.	plan* template
public Object assimilatePlanApprovalForm(WorkflowContext wfc)	Collects data from a form and puts the data back into the provisioningPlan. Assumes buildPlanApprovalForm.	form plan*
public Object provisionProject(WorkflowContext wfc)	Called by the Identity Update and LCM Workflows after provisioning forms are completed. Provisions the remaining items in the project.	project* noTriggers (Boolean)

Table 33—Identity Library Methods

Method / Usage	Description	Expected Args *=required
public Object finishRefresh(WorkflowContext wfc)	Called by the Identity Refresh workflow, after approvals are done and account completion attributes are gathered. Provisions what it can and completes the refresh process.	identitizer refreshOptions (map of args for creating new Identitizer if needed) previousVersion project
public Object buildApprovalSet(WorkflowContext wfc)	Called by the Lifecycle Manager workflows. Builds a simplified ApprovalSet representation of the items in the provisioning plan.	plan*
public Object processApprovalDecisions(WorkflowContext wfc)	Processes decisions made during approval process audit and react. Modifies the project masterPlan and recompiles the project if the recompile argument is set to true.	project* dontUpdatePlan disableAudit approvalSet* recompile
public Object processPlanApprovalDecisions(WorkflowContext wfc)	Processes decisions made during approval process audit and modifies the Used before the plan is compiled into a provisioningProject.	plan* dontUpdatePlan disableAudit approvalSet*
public Object auditLCMStart(WorkflowContext wfc)	Creates an audit event to mark the start of an Lifecycle Manager workflow.	approvalSet* flow (name of applicable UI flow)
public Object auditLCMCompletion(WorkflowContext wfc)	Creates an audit event to mark the completion of anLifecycle Manager workflow.	approvalSet* flow
public void disableAllAccounts(WorkflowContext wfc)	Used by lifecycle events to disable managed accounts for the identity specified in the workflow.	None
public void enableAllAccounts(WorkflowContext wfc)	Used by Lifecycle events to enable all accounts on the identity specified in the workflow.	None

Table 33—Identity Library Methods

Method / Usage	Description	Expected Args *=required
public void deleteAllAccounts(WorkflowContext wfc)	Used by Lifecycle events to delete all accounts on the identity specified in the workflow.	None
public ProvisioningPlan buildEventPlan(WorkflowContext wfc)	Go through all links that the workflow's specified Identity hold and creates a plan to enable or disable all of the Identity's accounts. Specified by **op**.	op* (operation)
public void updatePasswordHistory(WorkflowContext wfc)	Adds a password to the link password history	plan*
public ProvisioningProject assembleRetryProject(WorkflowContext wfc)	Adds any account request for an original provisioning project that are retryable and then adds them to a new provisioning project. Rarely used in custom workflows.	project
public Object retryProvisionProject(WorkflowContext wfc)	Executes the retry provisioning project, created in assembleRetryProject. Rarely used in custom workflow.	project
public Object mergeRetryProjectResults(WorkflowContext wfc)	Merges results from the retry project onto the main project. Called between retries. Rarely used in custom workflow.	project* retryProject*
public Boolean requiresStatusCheck(WorkflowContext wfc)	Identifies if the project contains any Results that are queued with a requestID stored on the result.	project
public Object checkProvisioningStatus(WorkflowContext wfc)	Calls down to the connector for each Result in the plan that is marked queued with a requestID specified.	project

Table 33—Identity Library Methods

Method / Usage	Description	Expected Args *=required
public Integer getProvisioningStatusCheckInterval(Workflow Context wfc)	Compute intervals between status checks for a request. The default is 60 minutes.	none
public Integer getProvisioningMaxStatusChecks(Workflow Context wfc)	Computes the maximum number of status checks permitted during a request. The default is infinite.	none
public Integer getProvisioningMaxRetries(WorkflowContext wfc)	Computes the maximum number of retries permitted during a request. The default is infinite.	none
public Integer getProvisioningRetryThreshold(WorkflowContext wfc)	Computes the retry threshold, the interval between retries, to use for a request. the Default is 60 minutes.	none

The methods are available for use. However these methods are rarely used in a custom workflow. It is recommended that custom workflows the workflow subprocesses instead of calling the library methods directly.

Note: This information is included for reference purposes and to document the purpose of the methods and what is passed to them. These explanations are also included to ensure that customizations do not remove calls to important methods from the subprocess workflows and to ensure that customizations only add other functionality around these method calls.

IdentityRequest Library

Table 34—IdentityRequest Library Methods

Method / Usage	Method / Usage	Expected Args *=required
public Object createIdentityRequest(WorkflowContext wfc)	Create s an IdentityRequest object from current workflow context information. Tracks status and history of request processing.	project* flow source policyViolations
public Object updateIdentityRequestState(WorkflowContext wfc)	Modifies the IdentityRequest's state.	identityRequestId

Table 34—IdentityRequest Library Methods

Method / Usage	Method / Usage	Expected Args *=required
public Object refreshIdentityRequestAfterApproval (WorkflowContext wfc)	Refreshes the IdentityRequest to include the provisioningEngine that processes the item. Updates the state and adds any expanded attributes that are provisioned.	project
public Object refreshIdentityRequestAfterProvisioning (WorkflowContext wfc)	After provisioning, copies interesting task result information back to the IdentityRequest.	project
public Object refreshIdentityRequestAfterRetry (WorkflowContext wfc)	Scans the retry project and updates the IdentityRequestItem retry count.	project
public Object completeIdentityRequest (WorkflowContext wfc)	Marks IdentityRequest status complete. Puts final plan in request and refreshes the request based on the final project.	project policyViolations autoVerify (Boolean)

Approval Library

Table 35—Approval Library Methods

Method / Usage	Method / Usage	Expected Args *=required
public SailPointObject getObject(WorkflowContext wfc)	Returns the object being approved.	none
public String getObjectClass(WorkflowContext wfc)	Returns the simple class name of the object being approved.	none
public String getObjectName(WorkflowContext wfc)	Returns the name of the object being approved.	none
public SailPointObject getCurrentObject(WorkflowContext wfc)	Returns the current persistent version of the object held in the workflowCase (approvalObject).	none
public Identity getObjectOwner(WorkflowContext wfc)	Returns the current owner of the object being approved. Uses database lookup.	none

Table 35—Approval Library Methods

Method / Usage	Method / Usage	Expected Args *=required
public Identity getNewObjectOwner(WorkflowContext wfc)	Returns the object owner. In the workflow context, the owner could be different than the database-recorded owner.	none
public String getObjectOwnerName(WorkflowContext wfc)	Returns name of ObjectOwner from getObjectOwner.	none
public String getNewObjectOwnerName(WorkflowContext wfc)	Returns name of NewObjectOwner from getNewObjectOwner.	none
public boolean isOwnerChange(WorkflowContext wfc)	Return true if object being approved has had an owner change.	none
public boolean isSelfApproval(WorkflowContext wfc)	Returns **True** if the user who launches workflow is the same as the owner of the object being approved. Used to bypass an owner approval. Assumes that the user will approve if the user is the one who is initiating the request.	none

Policy Violation Library

Table 36—Policy Violation Library Methods

Method / Usage	Method / Usage	Expected Args *=required
public Object delete(WorkflowContext wfc)	Deletes the current approval object associated with this workflow.	none
public Object ignore(WorkflowContext wfc)	Ends the workflow associated with the current approval object without performing any actions.	none
public Object mitigateViolation(WorkflowContext wfc)	Mitigates by temporarily allowing a policy violation.	expiration* comments
public Object getRemediatables(WorkflowContext wfc)		none

Table 36—Policy Violation Library Methods

Method / Usage	Method / Usage	Expected Args *=required
public Object getRemediatables(WorkflowContext wfc)		remediator actor Use if remediator argument is not specified and actor is. Use remediator in new method calls. comments remediations*

Role Library

Table 37—Role Library Methods

Method / Usage	Method / Usage	Expected Args *=required
public Object launchImpactAnalysis(WorkflowContext wfc)	Starts an impact analysis task for a role in workflow.	none
public Object getRoleDifferences(WorkflowContext wfc)	Calculates the differences between a role held in workflow and the database version of the role.	none
public Object auditRoleDifferences(WorkflowContext wfc)	Creates one audit event for each attribute difference between role states. Compares workflow vs database.	source action target string1
public Approval buildOwnerApproval(WorkflowContext wfc)	Sets up an approval for the owner of an object. Currently used only for roles.	none

Table 37—Role Library Methods

Method / Usage	Method / Usage	Expected Args *=required
public List<Approval> buildApplicationApprovals(WorkflowContext wfc)	For role approvals only. Builds an approval structure for the owners of each application referenced in the role profiles. Normally processed as parallelPoll to allow application owners to submit comments or modify the role without terminating the approval process.	none
public void enableRole(WorkflowContext wfc)	Marks role as enabled.	role (name)
public void disableRole(WorkflowContext wfc)	Marks role as disabled.	role (name)
public void setRoleDisabledStatus(WorkflowContext wfc)	Marks role with disabled status indicated in the disabled arg. true = disabled false = enabled	role (name) disable (Boolean)
public void removeOrphanedRoleRequests(WorkflowContext wfc)	Removes incomplete requests. Used to activate/deactivate roles that no longer exist.	none
public String getApprovalAuditAction(WorkflowContext wfc)	Called by the post-approval audit steps, Audit Failure and Audit Success, of Role Modeler. Owner Approval workflow to determine what type of action should be recorded in audit log. If the role is marked as disabled, returns disableRole. if the role is NOT marked as disabled, returns updateRole .	none

LCM Library

Currently, the Lifecycle Manager Library contains no public methods. All of its methods were moved to the Standard Workflow Handler.

Monitoring Workflows

After a workflow is initiated, the workflow can launch to completion quickly. Sometimes a workflow can take additional time to complete its specified actions. Approval steps often create a delay in the processing while the workflow waits for the approver to review the work item and make a decision on it.

To observe a workflow in flight and understand how much of the process is complete and what actions are pending, You can examine the Task Result for the workflow on the **Monitor -> Tasks -> Task Results** page. The TaskResult for a workflow exists for a period of time following the successful completion of the workflow. Based on the retention period set, the TaskResult can be purged soon after the process launches to completion. While the workflow is still in progress, the TaskResult continues to exist and can be examined for current step and status information.

Viewing the Workflow Case XML

You can examine the workflow case in XML format from the IdentityIQ console or from the Debug pages.The status of each step can then be determined from the data recorded in the workflow case.

To get the **workflowcase** XML from the console:

1. Launch the console.
2. List the workflow cases.
3. Get the specific workflow case in question by name. See the following example.

```
iiq console
> List workflowcase
[system will list all in-flight workflowcases by ID and name]
> get workflowcase "[workflowcase name]"
[system will display the XML for the workflow case]
```

To view the workflowcase XML from the IdentityIQ Debug pages:

1. Select WorkflowCase from the object list.
2. Click the specific workflow case from the list to display its XML.

Chapter 15: Advanced Workflow Topics

This chapter includes the following advanced Workflow topics:

- Loops within Workflows
- Launching a Workflow from a Task
- Workflow Forms

Loops within Workflows

A loop occurs when a step transitions back to a step that executed previously. The state of that step is re-initialized and the step is executed again. A loop can transition back any number of steps. You define a loop transition the same way you would any other transition. However, you must just select a target step that appears before the loop transition in the process designer.

In most case, when you create a loop transition, you want to give it a conditional When expression. If a loop transition is unconditional, the workflow can enter an infinite loop and not be able to finish.

Launching Workflows from a Task or Workflow

You can launch workflows from tasks or other workflows without using a system event to trigger the workflow.

Workflows Launched from Custom Tasks

You can launch workflows from a custom task in IdentityIQ. Because tasks are compiled java classes, the custom task must be written as a Java method.

To create a workflow from a custom task:

1. Create a WorkflowLaunch object in the Java method.
2. Populate the object with the data the workflow requires.
3. Use the Workflower class to launch the workflow.

It is often necessary for one workflow to launch another workflow. This can be performed in Beanshell using code similar to the previous example. However, using the workflow library method scheduleWorkflowEvent is easier. Not only does this method launch a workflow, it also allows you to delay the launch until a time in the future.

To have one workflow to launch another workflow, create a step and select scheduleWorkflowEvent as the action. This method requires the following arguments:

```
import java.util.HashMap;
import sailpoint.api.sailpointContext;
import sailpoint.api.Workflower;
import sailpoint.integration.ProvisioningPlan;
```

```
import sailpoint.integration.ProvisioningPlan.AccountRequest;
import sailpoint.integration.ProvisioningPlan.AttributeRequest;
import sailpoint.object.Identity;
import sailpoint.object.Workflow;
import sailpoint.object.WorkflowLaunch;
import sailpoint.tools.GeneralException;
import sailpoint.tools.xml.XMLObjectFactory;


HashMap launchArgsMap = new HashMap();

String myIdentityName = "T339222";
Identity myIdentity = context.getObjectByName(Identity.class,
myIdentityName);

//Create Provisioning Plan and add needed attribute values
ProvisioningPlan plan = new ProvisioningPlan();
plan.setIdentity(myIdentity);
AccountRequest accountRequest = new AccountRequest();
AttributeRequest attributeRequest = new AttributeRequest();

accountRequest.setApplication("IIQ");
accountRequest.setNativeIdentity(wbIdentity);
accountRequest.setOperation("Modify");

attributeRequest.setOperation("Add");
attributeRequest.setName("assignedRoles");
attributeRequest.setValue("Benefits Clerk");

accountRequest.add(attributeRequest);
plan.add(accountRequest);

//Add needed Workflow Launch Variables to map of name/value pairs
launchArgsMap.put("allowRequestsWithViolations","true");
launchArgsMap.put("approvalMode","parallelPoll");
launchArgsMap.put("approvalScheme","worldbank");
launchArgsMap.put("approvalSet","");
launchArgsMap.put("doRefresh","");
launchArgsMap.put("enableRetryRequest","false");
launchArgsMap.put("fallbackApprover","admin");
launchArgsMap.put("flow","RolesRequest");
launchArgsMap.put("foregroundProvisioning","true");
launchArgsMap.put("identityDisplayName","John.Smith");
launchArgsMap.put("identityName","John.Smith");
launchArgsMap.put("identityRequestId","");
launchArgsMap.put("launcher","admin");
launchArgsMap.put("notificationScheme","user,requester");
launchArgsMap.put("optimisticProvisioning","false");
launchArgsMap.put("plan",plan);
launchArgsMap.put("policiesToCheck","");
launchArgsMap.put("policyScheme","continue");
launchArgsMap.put("policyViolations","");
launchArgsMap.put("project","");
launchArgsMap.put("requireViolationReviewComments","true");
launchArgsMap.put("securityOfficerName","");
launchArgsMap.put("sessionOwner","admin");
launchArgsMap.put("source","LCM");
```

```
                launchArgsMap.put("trace","true");
                launchArgsMap.put("violationReviewDecision","");
                launchArgsMap.put("workItemComments","");

                sailpoint.object.ProvisioningPlan spPlan = new
sailpoint.object.ProvisioningPlan();
                spPlan.fromMap(plan.toMap());
                launchArgsMap.put("plan", spPlan);

                //Create WorkflowLaunch and set values
                WorkflowLaunch wflaunch = new WorkflowLaunch();
                Workflow wf = (Workflow)
context.getObjectByName(Workflow.class,"myWorkflowName");
                wflaunch.setWorkflowName(wf.getName());
                wflaunch.setWorkflowRef(wf.getName());
                wflaunch.setCaseName("LCM Provisioning");
                wflaunch.setVariables(launchArgsMap);


                //Create Workflower and launch workflow from WorkflowLaunch
                Workflower workflower = new Workflower(context);
                WorkflowLaunch launch = workflower.launch(wflaunch);

                // print workflowcase ID (example only; might not want to do this in
the task)
                String workFlowId = launch.getWorkflowCase().getId();
                System.out.println("workFlowId: "+workFlowId);
```

Workflows Launched by Other Workflows

Installations often have one workflow start another workflow using the scheduleWorkflowEvent method in the Standard Workflow Handler. One of the initiating workflow steps launches the method through a call action.

Arguments to the step including the following:

Table 38— scheduleWorkflowEvent Arguments

Name	Value
workflow	Name of the workflow to launch.
requestName	Name to be assigned to the request. **Note: If not specified, the name of workflow is the default.**
scheduleDate	Date and time you want the workflow to launch. Must be specified with a java.util.Date value. If this argument is not set, the workflow launches immediately.
scheduleDelaySeconds	An alternative to using scheduleDate. When set, the value is the number of seconds to delay before launching the workflow.
caseName	Specify a user friendly name for workflowCase to be displayed in the user interface. **Note: If no name is specified, the default is the name of workflow.**

Table 38— scheduleWorkflowEvent Arguments

Name	Value
launcher	Name of the identity to be displayed as the launcher of the new workflow case. If this argument is not specified, the launcher of the initiating workflow is used.

A workflow that is launched by another workflow is different from a workflow that is launched as a subprocess. If a workflow is launched as a subprocess, the calling workflow waits until the subprocess is completed. After the workflow returns control to the caller, the processing continues.

A workflow that is launched by another workflow causes a completely separate workflow to begin launching. After the new workflow is started, the original or calling workflow moves on to its next step.

Workflow Forms

Standard work item forms are available for presenting approval or other data requests to approvers. However, some installations prefer to use custom forms for these activities. Based on the type of the data collection effort, a custom form might be required. You can build a custom form using a <Form> element in the XML that is embedded within the <Approval> element.

Note: The <Approval> element can be used to collect data from a user, even if the workflow is not an approval. You generally use custom forms for these activities because the normal approval forms do not apply. However, you can also use custom forms for traditional approval activities when you need a different presentation format.

The basic elements in a Form definition are:

```
<Form>
   <Attributes>(map of name/value pairs that influence the form renderer)
   <Button> (determine form processing actions)
   <Section>(Subdivision of form; can contain nested Sections and Fields)
    <Field>(can contain Attributes map, Script to set value, Allowed Values Definition
script, and Validation Script)
```

For detailed information about working with forms, see “Forms” on page 239.

Process Variable and Step Forms

You use forms added to steps on the **Process Designer** tab in the Business Process Editor to request required data from a user that a process needs. For example, you can add a form to a step to request a value for a missing attribute.

However, to present information on the Basic Views of the **Process Variables** tab and the **Arguments** tab of the Step Editor, you use process variable and step forms.

To simplify the information displayed on the Process Variables tab:

- Variables are displayed in more logical groups.
- Variables that are rarely, if ever, modified are hidden.

Changes made in the Basic View are persisted to the Advanced View and more complex configuration can be performed there if needed.

The step forms are referenced from the workflows or stepLibraries. These forms define the form that is presented on the **Arguments** tab of the Step Editor panel and works similar to the process variable forms.

Both of these forms are referenced from workflows using the configForm variable. The forms can be defined, viewed and edited on the IdentityIQ debug page.

Chapter 16: Forms

Forms are used to present items to users for input in several components of IdentityIQ. They are used with:

- Application and role provisioning policies
- Identity provisioning policy (only applicable for installations using Lifecycle Manager)
- Data entry and approvals in workflow steps
- Present simplified views for process variable and step argument editing in workflows
- Report filter specification

The form implementation and available features varies slightly in these areas, so some features might apply to one use and not to the others, this is noted throughout this document.

For more information about using Forms with IdentityIQ, refer to the forms documents on the SailPoint customer support site or contact your support agent for more information.

Specifying Custom Forms

Form specification is different for each available use. All types of provisioning policies can be specified through the IdentityIQ user interface. In all cases, some of the more advanced and custom forms for workflows can be generated through the Business Process Editor. Some of the more advanced options. however, are only available through subsequent editing of the XML definition. Workflow forms created through the Business Process Editor are embedded within the workflow XML. Workflow forms created through the Business Process Editor are embedded within the workflow XML. Alternatively, they can be defined as independent form objects which can be referenced by multiple workflows, by creating them directly in XML and importing them into IdentityIQ. Report forms must be created as external XML documents and imported into IdentityIQ.

Role/Application Provisioning Policies

Provisioning forms are presented to a user when a provisioning request cannot be completed without user input. The data collection fields that are presented on the form come from the role or application's Provisioning Policy, which is defined by the <Form> element inside the Bundle (role) or Application object's XML. The actual form presented to the user during provisioning of roles or application accounts are system-generated at run-time based on skeleton forms that are pre-defined in IdentityIQ. Requests made through LCM are built with the Identity Update form. Requests that come through the Identity Refresh workflow use the Identity Refresh form. These forms contain a read-only section at the top that displays identifying information about the request, for example, Account ID, First name, Last name. The fields defined in the provisioning policy forms are added to the form at run time, when the form is presented to a user.

Provisioning policy forms define the fields required for the role or application account to be provisioned, often including a default value or script/rule for calculating a value. When a field cannot be calculated by the system during provisioning of an account or role, it must be presented to a user through a form to get the required value. When multiple accounts or roles are part of the same provisioning request, the form might display a collection of fields pulled from various provisioning police forms. On the form, the fields are, by default, grouped in sections according to the application or role to which they belong; this grouping can be overridden, by specifying a section attribute on each of the fields, naming the section into which each field should be placed. See the section attribute description in“Fields” on page 246.

Defining Application Provisioning Policy

An application provisioning policy can be defined for an application on the Provisioning Policies tab of the Application Configuration page, Applications -> Application Definition. Separate policies can be defined for create, update, and delete requests. Additionally, provisioning policies can be specified for creating or updating groups.

The required fields should be specified in the policy with the appropriate field attributes defined. These attributes can include a default value or a script/rule to calculate a default value for the field that can be based on the Identity attributes for the Identity for whom the request is being made. The field **Name** should match the corresponding native attribute on the application. If **Review Required** is selected, the field is always presented on a form during provisioning-request processing, even if a default value is provided or calculated successfully.

For creation-type operations you can specify dependencies between applications and application attributes that imply ordering of the provisioning requests.

Field Properties and Value Properties

The provisioning policy field attributes are grouped into two categories: Field Properties and Value Properties.

Field Properties describe field meta data. This includes the field's name, display name, tool tip help text, type, and owner. It also includes indications of whether the field is single or multi-valued, read-only, hidden, required, or review required. Fields can also be marked with a flag to indicate whether changes to the field value should cause the form to be reloaded. The Read-Only and Hidden attributes can be set to a static value (True or False) or can be defined programmatically through a rule or script. The rule and script options are used to dynamically hide and show the field, or change its edit properties, when the form is reloaded based on changes to values of other fields.

The Value Properties section includes properties specifically related to the field's value. A default value, a set of permitted values, and the field's validation logic can all be set here. The Dynamic attribute determines whether the field's value should be reevaluated on every form reload, when the form is reloaded based on a change in another field's value. It should only be selected when the field's value is rule or script based, such that it might change during the form processing based on other field values entered there.

The default value can be specified as a static value or can be calculated programmatically by a rule or script. In an account creation provisioning policy, an additional option, Dependent, is available as part of the ordered provisioning implementation, which is only available on account create provisioning policies. When the dependent option is selected, an application and attribute must also be selected and the value of the field is set to that attribute value for the Identity. Only applications on which this application is dependent are available for selection here.

The **Allowed Values** list can be specified as a list of values or can be set dynamically by a rule or script. Field validation is optional and can be managed by a reusable rule or with a script.

Defining Role Provisioning Policies

Role provisioning policies are specified through the Role Editor: go to Setup -> Roles, select the role name, and click **Edit Role**. Then click **Add Provisioning Policy** and specify the fields for the policy.

Select the application to which the role provisioning policy applies and then specify the fields for the policy. Fields are specified for role provisioning policies exactly as they are specified for application provisioning policies. Role provisioning policies and application provisioning policies are not the same or to be used interchangeably, however.

Role provisioning is not intended for initial role assignment or for the provisioning of account attributes that are not entitlements. Using role provisioning and application provisioning interchangeably cause conflicts and should be avoided.

Role provisioning is designed to be used for profiles that use complex logic, where it is unclear what should be provisioned or de-provisioned. The role provisioning policy is used to state what to provision, "x and y" or "p and q," and to use the contents of the Identity to make that decision.

Identity Provisioning Policy

The Identity Provisioning Policies are optional forms that can be specified to define the fields that must be provided when an Lifecycle Manager Create or Edit Identity request is submitted. When no Identity Provisioning Policy is defined for the create function, IdentityIQ automatically builds a form that includes the entire set of defined Identity attributes (standard and extended) for the installation. The auto-generated update provisioning policy form contains only identity attributes marked as editable. An Identity Provisioning Policy can be defined to select a subset of those fields, to affect the presentation of those fields, for example, grouping in sections or multi-column layout, or to build in some logic to auto-populate some of the fields.

A third identity provisioning policy also exists to support self-service registrations for IdentityIQ. This form is presented when self-service registration is enabled and a new user requests an IdentityIQ account. The form prompts the user for the information required to create a new user account for the installation.

To create an Identity Provisioning Policy, go to Identity Provisioning Policy of the Lifecycle Manager configuration page. Three policies are available: Create Identity, Update Identity, and Self-service Registration. If a policy has already been defined, the name is displayed. Click the name to open and edit the policy. If no policy has been defined for one of these types, click **Add Policy** to add a new one. Add fields to the policy, defining field attributes as needed on the field definition parallels for an application or role provisioning policy.

Identity Provisioning Policy forms are saved as independent form objects. System Configuration entries (entry key="createIdentityForm", "updateIdentityForm", and "registerForm") point to the appropriate forms for each identity provisioning policy by name. The identity provisioning policy forms are saved as <Form> objects inside the UIConfig attributes map under the keys lcmCreateIdentityProvisioningPolicy and lcmUpdateIdentityProvisioningPolicy on the IdentityIQ Debug pages. These form definitions can be edited directly to implement some of the presentation options, for example, multi-columns or sections. The configurable option available on the user interface do not include these features.

Note: Form features related to the Section attribute (which includes subdividing the form into sections and creating multi-column form configurations) are not supported through the user interface. These must be managed directly in the Form Object XML. Any fields added through the user interface after dividing the form into sections are automatically added to the first section. These fields can be moved to other sections by editing the XML.

Workflow Forms

Several standard work item renderers are provided with IdentityIQ for presenting approvals or other data requests to users. These are written as JSF pages. It is possible to write custom forms in JSF, specifying the JSF page as the renderer for the approval. This is rarely done. Customers who want to use custom forms generally specify these through a Form object.

Forms are used in workflows to present data-gathering pages to a user and define data presentation for approval activities. In many cases, implementations rely on the standard approval work item forms for normal approval actions so do not need to implement custom forms for their approval steps, but they still might choose to use custom forms for non-approval data-gathering activities to which the normal approval forms do not apply. A custom form can be added to a workflow through the Business Process Editor (Setup -> Business Processes) by right-clicking a step and choosing **Add Form** or by adding a form element to a step in the Workflow XML.

Whether the form is specified for an approval or a data-gathering activity, the form element must be embedded within an approval element in the XML. The user interface auto-creates it within an approval element. The workflow XML to specify a custom form looks like this:

```
?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE SailPoint PUBLIC "sailpoint.dtd" "sailpoint.dtd">

<sailpoint>

<Workflow explicitTransitions="true" libraries="Identity" name="Example Workflow"
type="IdentityLifecycle">

…

  <Step name="Display Form">

    <Approval name="Please enter some data" owner="admin" return="selectedApprover"
send="trace ">

       <Form>

        …          <!-- Form content goes here -->

      </Form>

    </Approval>

  </Step>
```

A custom form can also be created as an independent form object, defined in a separate XML document and imported into IdentityIQ, visible through the console or the Debug pages by viewing the Form objects, and referenced in the approval element as an argument like this:

```
 <Approval…>

     <Arg name='workItemForm' value='Custom Form Name'/>

      …

 </Approval>
```

This option promotes form reuse across workflows. However, these independent form objects cannot be edited through the Business Process Editor like the embedded forms can.

Process Variable and Step Forms in Workflows

While forms added to steps on the Process Designer tab of the Business Process Editor are used to request data required by the process from a user, such as a value for a missing attribute, the process variable and step forms are used to define the information presented on the Basic Views of the Process Variables tab and the Arguments tab of the Step Editor.

These forms are created as an independent form object, defined in a separate XML document and imported into IdentityIQ. They are visible through the console or the Debug pages by viewing the Form objects.

The process variables forms are used to simplify the information displayed on the Process Variables tab by hiding those variables that are rarely, if ever, modified and displaying variables in more logical groups. Changes made in the Basic View are persisted to the Advanced View and more complex configuration can be performed there if needed.

The step forms are referenced from the workflows or stepLibraries. These forms define the form that is presented on the Arguments tab of the Step Editor panel and works similarly to the process variable forms.

Both of these forms are referenced from workflows using the configForm variable. The forms can be defined and viewed and edited on the IdentityIQ debug page.

Report Forms

Report definitions often include a reference to a Form object for displaying the report-specific filter options to the report user. In the Report XML, the form is referenced with a <ReportForm> tag:

```
<ReportForm>
<Reference class="sailpoint.object.Form" id="39535985298ff9839ff98dd" name="My Custom Report Form" />
</ReportForm>
```

The Form is defined separately in its own XML document and imported into IdentityIQ as a Form object. Each section within the form is created as a separate page in the Edit Report window, where you specify the filters that are applied to the report. The report-specific forms are always merged with the Report Form Skeleton, which defines the Standard Properties and Report Layout pages that apply to every report.

Components of a Form Definition

The basic elements in a Form definition are:

```
<Form>
   <Attributes>(map of name/value pairs that influence the form renderer)
   <Button> (determine form processing actions)
   <Section>(Subdivision of form; may contain nested Sections and Fields)
    <Field>(may contain Attributes map, Script to set value, Allowed Values Definition script, and Validation Script)
```

Within each of these sections of the form definition, certain attributes might apply to some form uses and not to others. The table below provides a high-level overview of which of the available form elements can be specified for each.

Form Usage	Form Component Availability			
	Field	**Button**	**Section**	**Field**
Application and role provisioning policies (Form)				✓
Identity provisioning policy	✓*	✓*	ü	ü
Workflow approval	ü	ü	ü	ü
Report			ü	ü

* Limitations on the Attribute and Button elements for Identity provisioning policy are discussed in “Attributes” on page 244 and “Buttons” on page 245.

Form

The Form element should contain a single attribute to define the form: a name.

```
<Form name="My Custom Form">
```

If the form is stored in the database as an independent Form object, the name must be unique, no two Form objects can share the same name. This restriction does not apply to Forms defined in-line within a workflow approval step. Name is required for independent Form objects, it is recommended but is not required on in-line forms.

Attributes

Forms can include a map of attributes that are used by the renderer. These are applicable only to workflow forms and, in a limited way, to the identity provisioning policy form.

Attributes are specified with the following keys:

Key	Description
pageTitle	Title to render at top of page, typically larger and a different color than the form title; also displayed in browser window header bar in some cases
title	FormTitle, shown at top of form body
subtitle	Form subtitle, shown below title
readOnly	<entry key="readOnly= "" value="true""/> makes form read-only so the fields are rendered as uneditable text or as disabled HTML components
hideIncompleteFields	hideIncompleteFields="true" hides any fields that do not have all of their dependencies met. Usually only set programmatically to control field presentation in provisioning policy forms, though can be specified in a workflow form XML. This does not create dynamically displayed fields on forms. Fields are not displayed on a form after their dependency values on the same form are entered. Use the new hidden attribute on Fields and Sections to achieve this dynamic display functionality.

Note: The Boolean attributes are only specified if they are true; they default to false when omitted.

Attributes maps are specified as shown here:

```
 <Attributes>
   <Map>
      <entry key="pageTitle" value="Review Non-Employee Request"/>
      <entry key="readOnly" value="true"/>
   </Map>
</Attributes>
```

Attributes do not apply to report forms because the sections in a report form are pulled out of the report's form definition and combined into the Report Form Skeleton for display in the user interface. Even if they are specified in the report form, these attributes are never applied to the resultant form that is displayed to the user.

On an identity provisioning policy form, the pageTitle attribute is ignored because the main page title is programmatically set based on the other action being performed (Create New Identity, Edit Identity Attributes for [Username], or New User Registration). The title and subtitle attributes are displayed in the user interface when

specified in the form's attributes map. The readOnly, and hideIncompleteFields function on this form type should not be used because they do not provide useful functionality for this type of form.

Buttons

Buttons enable the user to indicate which action to take next and how to process the data on the form. Buttons only apply to workflow forms. Buttons can be specified on identity provisioning policy forms, but the window does not support any action on them other than next (submit). Since **Submit** and **Cancel** buttons already exist on the window and perform the appropriate functions for the window, additional buttons are unnecessary. They cannot be specified in a role or application provisioning policy form, and they are not used by the report executor when it combines the specific report's form with the Report Form Skeleton.

Buttons require two attributes, a label and an action. The label determines the text displayed on the button. The action determines what the system does in response to clicking that button. There are four available actions:

- **next**: save (and validate fields with validation scripts where specified) any entered form data and set the work item status to approved. This can then be used in the Transition logic to advance the workflow to the next step (OK/Save/Approve/Submit functionality).
- **back**: save entered form data (no validation is performed) and set the work item status to rejected. This can then be used in the Transition logic to return to a previous step or any other appropriate action for a rejection. Saved value is redisplayed on this form if the workflow logic process back through this step again.
- **cancel**: close the form, suspend the workflow and return to previous page in user interface, this leaves work item active. awaiting a different action choice by the user.
- **refresh**: save the entered form data and regenerate the form; not a state transition - just a redisplay of the form (rarely used).

These are examples of button elements.

```
<Button label='Submit' action='next'/>
```

```
<Button label="Cancel" action="cancel"/>
```

Sections

Sections divide a form into logical groupings that are visually marked on the window with boxes around the fields in each section. They can be specified in the XML for all policies except application and role provisioning policies. By default, a separate section is created on the provisioning form for each application (each application's provisioning policy form is rendered in its own section). However, fields in a provisioning policy form can be specified with a section attribute that causes them to be displayed in different sections from the defaults. Sections are treated differently on report forms, each section becomes a separate page on the Edit Report window rather than just a separate section on a contiguous form.

Sections are specified in the form object's XML with a <Section> tag and can be modified by the attributes shown in the table below.

Section Attributes	Description
name	Internal name for section (might be referenced by field objects in some forms).

Section Attributes	Description
label	If non-null, the label is displayed above the section fields in a box on the section border. For report forms, the label is specified in the Edit Report window's sections list. Labels can be specified with text, message catalog keys, or variables (specified with `$(varName)` notation).
type	Rendering type (optional). When no type is specified, fields in the section are editable fields, displayed one field per row, unless the columns attribute specifies otherwise. Other type options are: **datatable**: makes fields in the section non-editable; generally used to display a read-only informational table to give the form user a context for the form's requested data **text**: indicates the section is a block of informational text; if multiple fields are included in a text section, each field is rendered on a separate line with line breaks between them
columns	Number of columns contained in the form section; fields are placed in columns left to right, one field per column before moving to the next row. For example, in a 2 column layout (columns="2"), 4 fields are displayed: Field 1 Field 2 Field 3 Field 4

These are examples of Section elements in the XML for forms.

```
<Section name="authorizations" label="Authorizations" type="datatable">
```

```
<Section columns="2" label="rept_app_section_label" name="customProperties">
```

Sections contain nested field elements and might contain nested sections when sub-groupings are needed.

Fields

Fields are the core element of forms, they are the mechanism by which data gets communicated to and from the user. Fields offer options that affect the appearance or functionality of the field. Some of these are commonly used and others are used very infrequently. Some of these are specified as in-line attributes in the <Field> tag and others are specified as nested elements within the Field.

Field attributes appropriate to all form uses are:

Field Attributes	Description
name	Name for the field that can be referenced in code as the variable name in which the field's value is stored. Avoid using the following field names: accept accept-charset action autocomplete enctype method name novalidate target As well as global attributes: accesskey class contenteditable contextmenu,data-* dir draggable dropzone hidden id itemid itemprop itemref itemscpe itemtype lang spellcheck style tabindex title
displayName	Label for the field; can be text or a message key.
helpKey	Tool tip help text; can be text or a localizable message key. Example: `<Field name="firstName" displayName="First Name" helpKey="Enter the person's first name" />`

Field Attributes	Description
type	Field datatype; influences the display widget used to display the field on a form. **Valid values are:** string, int, long, boolean, date, and SailPoint object types (Identity, Bundle, Permission, Rule, Application), default is string. SailPoint objects are displayed as drop-down lists or combo boxes if multi="true" is also specified. Specifying type="boolean" renders the field as a checkbox; specifying type="date" adds a calendar from which the date can be selected. To pre-select an object in the list, specify the name of the object (not an actual object) as the "value" attribute. Example: `<Field name="role" displayName="Role" type="Bundle" value="TRAKK Basic" />`
multi	Boolean indicating whether the field is multi-selectable. This attribute is only appropriate to drop-down lists, which are then displayed as combo boxes. Used this with SailPoint object field types or with a nested AllowedValues / AllowedValuesDefinition element that populates a selection list for the field. Example: `<Field name="apps" displayName="Applications" type="Application" multi="true"/>`
readOnly	Boolean indicating that the field cannot be edited on the form. The value is displayed as text not in an editable box. Not necessary to specify on fields in a datatable section, since they are already read-only.

Field Attributes	Description
hidden	Boolean that, when true, prevents the field from being displayed on the form. This attribute is used in reporting to make fields available for inclusion on the report detail grid but not actually include them by default. This can be used in any form, but might not be commonly implemented: • In role/application provisioning policies, fields are only shown if the user needs to enter data, so forcing fields to be hidden is not helpful. • In workflows, the hideIncompleteFields attribute on the Form object is more likely to be used with the dependencies attribute on Field to defer field display until dependencies are fulfilled. • In Identity provisioning policy, fields that should be hidden can be omitted from the form instead, however, this could be used for fields that always contain the same value for all users to set that value and suppress the field from the data entry form. For example, `<Field name="status" hidden="true" value="NewHire" /` **Note: Attributes mark hidden are not included in the plan. You must manually add the includeHiddenFields property to the form to include the hidden fields in the plan.** **<entry key="includeHiddenFields" value="true"/>**
required	Boolean indicating whether a value is mandatory for the field; required="true" marks field with * on form to indicate required and prevents form submission without a value for the field Example:`<Field name="myfield" displayName="My Field" required="true"/>`
postBack	Boolean that, when true, causes the form to refresh when the field's data value changes, running any rules or scripts that run on form load `<Field name="application" displayName="Application" type="Application" postBack="true"/>` Supports conditional display/editing of sections or fields based on other field attributes values, automatic population of fields based on other fields, and validation of fields based on actions taken on the form. It only runs when a field loses focus, so it can be used on selection fields or text entry fields.

columnSpan	Used when the section is configured with multiple columns; specifies the number of columns the field should span
	
	`<Section columns="2" label="Identity Info" name="identInfo">`
	
	`<Field name="fname" displayName="First Name" columnSpan="2"/>`

Field Attributes	Description
filterString	Used for fields where type is a SailPointObject class to specify a filter to restrict the set of selectable objects presented in the drop-down list. filterString is specified according to the filter string syntax and should be specified in single quotes so double quotes can be used within the string. Example: `<Field displayName="Role" name="role" type="Bundle" filterString='name.startsWith("TRAKK")' />`

section	Statically defined fields in a form's XML are defined within a section element, so any section attribute specified on those fields is ignored. However, the section attribute can be used to organize fields in an application or role provisioning policy or on a dynamically rendered form.
	
	Application and role provisioning policy forms do not have section elements, so the section attribute can be used to force fields to be grouped differently than the default (default is by application or by role).
	
	Example:
	
	These two fields are put on the form in separate sections, labeled "Important Items" and "Optional Items" respectively.
	
	`<Field name="myField" displayName="My Field" section="Important Items"/>`
	
	`<Field name="optField" displayName="Optional Field" section="Optional Items"/>`
	
	The section attribute on fields is also used in dynamically created forms (such as in Reports where fields are added to the form programmatically through an initialization script). This attribute enables the code to specify the section of the form into which the field should be added.

Field Attributes	Description

displayType	Forces string fields to display as specified, used only for string fields
	
	Valid displayTypes are: radio, combobox, textarea, and label
	
	displayType="radio" and "combobox" are used to override the default display format for permitted-values fields (radio is the default for 2 options while >2 options is rendered as comboBox by default). textarea is used to make a string field display as a text area instead of a regular entry field.
	
	`For label, you can use the field displayName for the text/message key of the label.`
	
	`<Field name="dept" displayName="Department" displayType="radio"`
	
	`<AllowedValues>`
	
	`<String>Accounting</String>`
	
	`<String>Manufacturing</String>`
	
	`<String>Engineering</String>`
	
	`</AllowedValues>`
	
	`</Field>`
	
	`<Field name="comments" displayName="Comments" displayType="textarea" />`

value	Sets the default/initial value for the field. This can be overwritten on the form in most cases as long as the field is not marked readOnly. This is used within sections of type="text" to specify the text to display For application or role provisioning policies, setting a value (whether with this attribute or through a nested <Value>, <RuleRef>, or <Script> element) prevents the field from being included on the form unless reviewRequired is specified since provisioning policies only collect values from a user that they cannot determine or calculate independently. In workflow approvals, value can be specified by string, rule, script, call, or reference (string is default). In reports forms, the value is a reference to the report taskDefinition's input parameter from which to retrieve the starting / default value for the field, for example, value="ref:applications". Example: `<Field name="role" displayName="Role" type="Bundle" value="TRAKK Basic"/>`

Field Attributes	Description
dynamic	This attribute performs two separate functions: (1)For fields with an AllowedValuesDefinition, delays running of allowed values scripts/rules until the field is clicked, instead of running at form load, so it can make use of other data entered on the form instead of just data available on initial form load. (2)During form refresh in response to a value change of a field marked for postBack, only fields marked as dynamic (dynamic="true") have their value scripts/rules re-run; otherwise, the initial value calculated for the field on form load remains in effect as the field's default value

These attributes only apply to the application and role provisioning policies:

Field Attributes	Description
dependencies	List (CSV) of other fields that must be evaluated before this field. Dependencies on the provisioning policy (form) field cause that field to be deferred to a subsequent form that is presented after the form on which its dependencies are presented. The field might also be calculated based on those dependencies instead of presenting it on a later form. This attribute can also be used with dynamic/allowedValues fields. Values of dependencies fields are made available to the allowedValues script or rule, even if the field is presented on a different form.
reviewRequired	Enables a default value to be assigned to the field while still including the field on the form displayed to a user. This enables the default to be edited. If reviewRequired="true" is not specified, provisioning policy form fields with a default value (or value script/rule that returns a value) are omitted from the user-facing form and the default value is automatically used.
authoritative	Boolean that specifies whether the field value should replace the current value rather than be merged with it. Valid for multi-valued attributes only: `<Field name="costCenter" displayName="Cost Center" multi="true" authoritative="true"/>`

Fields can also contain nested elements that help control the display or use of the field.

Nested Elements within Field Elements	Description

Description	Field description, used for XML self-documentation. Not displayed in user interface.
	
	`<Description>`
	`  This field stores the Identity's first name.`
	`</Description>`

Attributes	Attribute map used to control field rendering, specific to the field type. The most common attributes are height and width which are usually specified for textarea fields and for entry boxes that need to be other than the default rendering size. Units for height and width are in pixels
	
	`<Attributes>`
	`  <Map>`
	`  <entry key="height" value="200"/>`
	`  <entry key="width" value="100"/>`
	`  </Map>`
	`</Attributes>`
	
	Two special attributes - xtype and vtype - are discussed in the section below.

Value	Alternative to "value" attribute on <Field>. This is required when specifying complex datatypes such as Map or List.
	
	`<Value>`
	`  <List>`
	`  <String>Thing 1</String>`
	`  <String>Thing 2</String>`
	`  </List>`
	`</Value>`
	
	Also needed for fields of type Date, which are specified as the utime representation of the date:
	
	`<Value>`
	`  <Date>1231971297</Date>`
	`</Value>`
	
	Can be used to specify simpler types like String, Boolean, etc. but not commonly done because value attribute is simpler.

Nested Elements within Field Elements	Description

Script	Script used to initialize the value of the field, alternative to the value element/attribute. Automatically created for fields whose value is set by script through user interface specification.
	
	Example:
	
	`<Script>`
	`  <Source>`
	`  [beanshell code goes here]`
	`return [value or variable that contains value to assign to the field];`
	`  </Source>`
	`</Script>`

RuleRef	Reference to a reusable rule for initializing field value. Alternative to <Script> (and value attribute). Automatically created for fields whose value is set by script through user interface specification.
	
	`<RuleRef>`
	`<Reference class="Rule" name="My Rule" id="402839343985ff930d" />`
	`</RuleRef>`

AllowedValues	Specifies a set of values from which the user can select to assign the field value. Automatically created for fields with an allowed values property set to Value (with a list of values specified) through user interface specification.
	
	`<Field name="dept" …>`
	`  <AllowedValues>`
	`  <String>Accounting</String>`
	`  <String>Manufacturing</String>`
	`  <String>Engineering</String>`
	`  </AllowedValues>`
	`</Field>`
	
	The list renders as radio buttons when only two options exist (and multi is not allowed), as a listbox for more than two options, and as a combobox for multi-selectable fields.

Nested Elements within Field Elements	Description

AllowedValuesDefinition	Populates a list of values from which the user can select a value for the field. This field contains either a <Script> block that specifies the list programmatically or a <RuleRef> that points to a rule containing the beanshell for generating the list. Automatically created for fields with an allowed values property set to Script or Rule through user interface specification.
	
	`<AllowedValuesDefinition>`
	`  <Script>`
	`  <Source>`
	`  import sailpoint.object.*;`
	`  List l = new ArrayList();`
	`  for(WorkItem.State enumItem :`
	`WorkItem.State.values()) {`
	`  List l2 = new ArrayList();`
	`  l2.add(enumItem.toString());`
	`  l2.add(enumItem.getMessageKey());`
	`  l.add(l2);`
	`  }`
	`  return l;`
	`  </Source>`
	`  </Script>`
	`</AllowedValuesDefinition>`
	
	Alternative to AllowedValues element and more commonly used. The list renders as radio buttons when only two options exist (and multi is not allowed), as a listbox for more than two options, and as a combo box for multi-selectable fields.

ValidationScript	Script used to examine and validate the field value entered by the user. The value entered is passed to the validation script in the variable named "value."
	
	`<ValidationScript>`
	`  <Source>`
	`  if (value > 10) {`
	`  return "Value must be less than or equal to 10";`
	`  else`
	`  return null;`
	`  </Source>`
	`</ValidationScript>`
	
	Returns null if no errors and an error message (as string or SailPoint**.tools.message** object) if validation errors exist.

ValidationRule	Reference to reusable rule for field validation. This is the alternative to ValidationScript.
	
	`<ValidationRule>`
	`  <Reference class="Rule" name="My Validation Rule"`
	`id="4028392342f5ff9301" />`
	`</ValidationRule>`

Nested Elements within Field Elements	Description

OwnerDefinition	Used only for application and role provisioning policies to determine the Identity to whom the fields should be presented. This enables specification of a RuleRef, script, a Value element or a Value attribute:
	
	`<OwnerDefinition>`
	` <RuleRef>`
	`  <Reference class="rule" name="My Owner Rule" id="4038293483598523" />`
	` </RuleRef>`
	`</OwnerDefinition>`
	`or`
	`<OwnerDefinition>`
	`  <Script>`
	`  <Source>`
	`  import sailpoint.object.*;`
	`Identity myIdentity=context.getObjectByName(Identity,"Walter.Henderson");`
	`  return myIdentity;`
	`  </Source>`
	`  </Script>`
	`</OwnerDefinition>`
	`or`
	`<OwnerDefinition value="IIQApplicationOwner"/>`
	
	Can provide either the string name of owning Identity or the Identity object.
	
	As with Field value, OwnerDefinition value can also be expressed as a nested element. It can be a string Identity name or an Identity object:
	
	`<OwnerDefinition>`
	`  <Value>`
	`  <String>Walter.Henderson</String>`
	`  </Value>`
	`</OwnerDefinition>`
	
	Three special names exist that are translated by IdentityIQ into the appropriate Identity so an OwnerDefinition script is not required for them:
	- IIQParentOwner - owner of the role or application containing the provisioning policy form
	- IIQApplicationOwner - owner of the application associated with the provisioning policy form
	- IIQRoleOwner - owner of the role containing the provisioning policy form
	
	`<OwnerDefinition value=""/>` assigns and presents the field to the access requester.
	
	The user interface offers these options for setting field owners: **Requester** (sets OwnerDefinition to ""), **Application Owner** (sets to "IIQApplicationOwner"), **Role Owner** on Role Provisioning Policies(sets to "IIQRoleOwner"), and **Rule** and **Script** (save as OwnerDefinition with nested RuleRef or Script, as shown above).

Nested Elements within Field Elements	Description

AppDependency	Applies only to application provisioning policies as part of the ordered provisioning function; sets the value for a field based on the value of an attribute on another application on which it is dependent
	
	`<Field displayName="Login ID" name="login" type="string">`
	
	`<AppDependency applicationName="LDAP" schemaAttributeName="employeeNumber"/>`
	
	`</Field>`
	
	This can only be specified when the application has dependencies declared and can only reference attributes on an application on which the application is dependent. The user interface option for field value named **Dependency** creates this element in the field definition.

Working with the Form Editor

The Form Editor provides a graphical user interface enabling you to create and edit forms without having to edit the xml directly.

The Form Editor contains the following sections:

- Detail View — detailed information about the selected form
- Expandable Tree View — provides an ordered, hierarchical view of the form components
- Edit Options — the available attributes for the selected form item

Detail View

This section displays the detail information about the selected form on clicking the **Details** button. The following table lists displayed attributes for the respective Form Type:

Form Type	Attributes
Application Provisioning Policy Form	Title, Subtitle, Wizard, Owner
Role Provisioning Policy Form	Title, Subtitle, Wizard, Application, Owner
Workflow Form	Title, Subtitle, Wizard

Expandable Tree

The expandable tree section provides an ordered and hierarchical view of the form components.

The tree section can be subdivided into the following components:

- Action buttons — buttons for following actions:
  - **Add Section** — adds section to the expandable tree view
  - **Add Button** — adds button to the bottom of the expandable tree view
    Note: The **Add Button** is applicable to Workflow Approval Forms.
  - **Preview Form** — displays the form layout for all included form components in editor. Helps to preview the form while developing a form to see how it renders on actual operations.
- Components in the tree view — Following are the different components of the tree panel:
  - **Section** — Multiple sections can be added to the tree panel through the **Add Section** button. Using **+** icon Fields and Row with Columns can be added. The Section item can be expanded or collapsed by clicking on them.
    - **Add Field** — Fields can be added under the Section.
    - **Add Row with Columns** — Rows with a maximum of four columns can be added under the Section using the **Choose how many columns in this row** drop down list under the **Edit Options** section.

      Note: When using Rows, the columns attribute of Section and columnSpan attribute of Field would be calculated by Form Editor and existing values would be overwritten.
  - **Button** — All the defined Buttons are added at the end in the tree panel.

Reordering Form Components

Form components can be reordered using drag/drop feature in the following way:

- Sections — sections can be reordered. Sections cannot have sections within them.
- Rows — rows can be reordered within the Section or dragged and dropped into any Section.
- Fields — fields can be reordered within Rows or dragged and dropped into any Section.
- Buttons — can be reordered only within Buttons.

Note: For inappropriate moves of the form components the not allowed icon is displayed.

Edit Options

The Edit Options section on the Editor page displays the attributes that must be modified for the respective actions.

Note: Click on the Apply button after the attributes are modified.

Section

Attributes	Description
Basic	
Name	Internal name for section.

Attributes	Description
Label	Label of Section determines Section text on Edit page. Labels can be specified as text, message catalog keys, or variables (specified with $(variableName) notation).
Subtitle	Section subtitle as description (displayed at the top of the section, above all fields in the section).
Settings	
Hidden	Boolean that, when true, prevents the field from being displayed on the form.
Read Only	Section properties are read only.
Hide Nulls	When set to true, hides fields within the section which have a null value.

Fields and Rows

Attributes	Description
Settings	
Name	Name for the field that can be referenced in code as the variable name in which the fields value is stored
Display Name	Label for the field; can be text or a message key.
Help Text	The text that appears when hovering the mouse over the help icon.
Type	Select the type of field from the drop down list. Select from the following: Boolean — true or false values field. Date — calendar date field. Integer — only numerical values field. Long — similar to integer but is used for large numerical values. Identity — specific identity in IdentityIQ field. Secret — hidden text field. String — text field Application — list of existing application Role — existing type of bundles
Type Settings	
Multi-Valued	Enable this to have more than one selectable value in this field of the generated form.
Refresh On Change	Boolean that, when true, causes the form to refresh when the fields data value changes, running any rules or scripts that run on form reload.

Attributes	Description
Authoritative	Enable to have the field value override the current value rather than merge with it. Applicable only for multivalued attributes.
Required	Boolean indicating whether a value is mandatory for the field; required="true" marks field with * on form to indicate required and prevents form submission without a value for the field.
Read Only	Determine how the read only value is derived: True — value based on the selection from the drop down list Rule — value is based on a specified rule Script — value is determined by the execution of a script
Hidden	Boolean that, when true, prevents the field from being displayed on the form. True — value based on the selection from the drop down list Rule — value is based on a specified rule Script — value is determined by the execution of a script
owner	The owner of the field/row. This is determined by selecting from the following: None — no owner is assigned to this field/row Requester — sets the Owner to this field/row Rule — use a rule to determine the owner of this field/row Script — use a script to determine the owner of this field/row
Value Settings	
Value	Sets the default/initial value for the field/row. This can be overwritten on the form if the field/row is not marked as Read Only. Select None, Value, Rule or Script option.
Allowed Values	Specifies a set of values from which the user can select to assign the field value. Automatically created for fields with an allowed values property set to Value (with a list of values specified) through user interface specification. Select Value, Rule or Script option.
Validation	Ability to specify a script or rule for validating the user's value by selecting None, Rule or Script.

Button

Attributes	Description
Settings	
Action	Action determines what the system does in response to clicking the associated button. Select one from the drop down list: Next — used in the Transition logic to advance the workflow to the next step Back — used in the Transition logic to return to a previous step or any other appropriate action for a rejection. Refresh — save the entered form data and regenerate the form; not a state transition just a redisplay of the form
Read Only	Determines whether to show a button or not on the form renderer.

Attributes	Description
Skip Validation	Determines if client-side required item validation is necessary based on the button clicked by the user. Validation is required if the button is not configured to skip validation, the action is NEXT and there are required items.
Label	Determines the text displayed on the button.
Parameter	Action-parameter of the button.
Value	Action-parameter value.

Form Examples

This section contains examples of XML specifications for the various types of forms.

Application and Role Provisioning Policies and Workflow Forms can all be created through the user interface, though some advanced features might require XML editing to implement. All form types are recorded as XML objects that can be edited through the debug pages as needed. This section reviews the form types in their XML format and shows how they are rendered as a form in the user interface based on that XML definition.

Application and Role Provisioning Policy

Application provisioning policies are specified as <Form> within the <Application> definition. Role provisioning polices are <Form> within the <Bundle> definition. Applications might have more than one provisioning policy form - one for (account) creation, update, and delete provisioning activities plus additional policies for group creation and update. Roles might only have one for role assignment to an Identity.

This sample <Form> definition provides examples of fields slotted into separate sections, assigned to different owners by value or by script, with an permitted values set, and with a validation script. Application provisioning policies are specified within a <Forms> element that wraps all of the specified provisioning policy forms together.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Form PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<Form name="New Acct Policy" type="Create">
  <Field displayName="Name" name="name" required="true" reviewRequired="true"
type="string">
    <OwnerDefinition value="IIQApplicationOwner"/>
  </Field>
  <Field displayName="Phone" name="phone" required="true" section="Extra Info"
type="string">
    <OwnerDefinition value="IIQApplicationOwner"/>
  </Field>
  <Field displayName="Office Number" name="off_no" required="true" section=""
type="integer">
    <OwnerDefinition>
      <Script>
        <Source>return identity.getManager();</Source>
      </Script>
    </OwnerDefinition>
    <ValidationScript>
      <Source>
   try {
int number=Integer.parseInt(value);
```

```
if (number &lt; 100) {
return "Office numbers are all 100 or greater.";
} else{
return null;
}
} catch (NumberFormatException e) {
return "Non-numeric value provided; must be numeric.";
}
</Source>
    </ValidationScript>
  </Field>
  <Field displayName="Region" name="region" required="true" type="string">
    <AllowedValues>
      <String>Americas</String>
      <String>EMEA</String>
      <String>APAC</String>
    </AllowedValues>
  </Field>
```

Application Provisioning Policies can render on multiple forms, depending on the field Owners. Multiple provisioning policy forms can be combined into one form if a request spans multiple applications or roles that each need to gather additional data from the same user.

Identity Provisioning Policy

The XML below creates an identity provisioning policy which implements many of the available form options, including:

The form includes multiple field types (: string, object, and secret -. Secret hides enteredthe text). as it is entered. Object fields are rendered as drop-down list boxes pre-populated with all available items of that type.

- Multi-column configurations
- Multi-column spans for some fields
- Allowed values lists
- Tool tip help prompts
- Field validation (runs when user clicks Submit)
- Filter on object lists for example, show only Manager Identities in Manager drop down list
- Conditional display of sections based on entered field values
- Population of fields based on values entered in other fields

The form includes multiple field types: string, object and secret. Secret hides the text as it is entered. Object fields are rendered as drop-down list boxes pre-populated with all available items of that type.

```
     <?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Form PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<Form name="Identity Create Policy" type="CreateIdentity">
  <Description>This is the provisioning policy used when creating a new identity thru
LCM.</Description>
  <Section columns="2">
    <Field displayName="First Name" name="firstname" required="true"
reviewRequired="true" type="string"/>
```

```
    <Field displayName="Last Name" name="lastname" postBack="true" required="true"
type="string"/>
    <Field columnSpan="2" displayName="Username" dynamic="true" helpKey="cube name"
name="name" required="true" type="string">
      <Script>
        <Source>
           if ((null != firstname) &amp;amp;&amp;amp; (null != lastname)) {
               return (firstname + "." + lastname);
           }
           return null;
        </Source>
      </Script>
      <ValidationScript>
        <Source>
            // validation variable comes in as "value"; messages value returned
           // is displayed on screen below field on validation; success should return
            // empty messages list
            import sailpoint.tools.Message;
            import sailpoint.object.Identity;

            List messages = new ArrayList();

            Identity existing =
(Identity)context.getObjectByName(Identity.class,value);
            if (existing == null) {
               // No Identity found with that name, so return empty messages -
               // validation successful
               return messages;
            } else {
               Message msg = new Message();
              msg.setKey("Username: " + value + " already exists. Modify this name
to make it unique.");
               messages.add(msg);
               return messages;
            }
        </Source>
      </ValidationScript>
    </Field>
```

```
    <Field displayName="Password" name="password" reviewRequired="true"
type="secret"/>
    <Field displayName="Password Confirmation" name="passwordConfirm"
reviewRequired="true" type="secret"/>
    <Field displayName="Employment Type" displayType="combobox" name="status"
postBack="true" type="string">
      <AllowedValues>
        <String>Employee</String>
        <String>Contractor</String>
      </AllowedValues>
    </Field>
  </Section>
  <Section label="Employee Only Fields">
    <Attributes>
      <Map>
        <entry key="hidden">
          <value>
            <Script>
              <Source>
                 if ("Employee".equals(status)) {
                     return false;
                 } else {
                     return true;
                 }
               </Source>
            </Script>
          </value>
        </entry>
      </Map>
    </Attributes>
   <Field displayName="Manager" filterString="managerStatus == true" name="manager"
type="sailpoint.object.Identity"/>
   <Field displayName="att_email" dynamic="true" name="email" reviewRequired="true"
section="" type="string">
      <Script>
        <Source>
            if (("Employee".equals(status)) &amp;&amp; (null != firstname) &amp;&amp;
(null != lastname)) {
                    return (firstname + "." + lastname + "@demoexample.com");
```

```
                }
                return null;
            </Source>
      </Script>
    </Field>
   <Field displayName="Location" name="location" reviewRequired="true" type="string"
value="Austin">
      <AllowedValues>
        <String>Austin</String>
        <String>Brazil</String>
        <String>Munich</String>
        <String>London</String>
        <String>Brussels</String>
        <String>San Jose</String>
        <String>Chicago</String>
        <String>Taipei</String>
        <String>Tokyo</String>
      </AllowedValues>
    </Field>
  </Section>
</Form>
```

Workflow Form

This example XML creates a custom form that displays the Identity's name and asks the user to select a region to which the Identity should be assigned. It demonstrates use of an AllowedValuesDefinition and a ValidationScript as well as Sections of all three types, text, datatable, and default. This form is embedded in the Workflow XML, as it would be if the form were created through the Business Process Editor **Add Form** option. The form could alternatively be created as a standalone form object and referenced as an argument to the approval, as described in “Workflow Forms” on page 241.

```
<Step name="Need Region" posX="359" posY="182">
  <Approval name="Need Region" owner="ref:launcher" return="region"
          send="identityName">
    <Arg name="workItemDescription"
           value="string:Fill in Region for $(identityName)"/>
    <Form>
     <Attributes>
        <Map>
          <entry key="pageTitle" value="Get Region"/>
          <entry key="title" value="Need Region for Identity"/>
        </Map>
     </Attributes>
     <Button action="back" label="Abort"/>
     <Button action="next" label="Submit"/>
     <Button action="cancel" label="Return Item to Inbox"/>
```

```
     <Section name='userInstructions' type='text'>
        <Field value="Employees must be assigned to a region.  Please provide the
correct region for this employee."
/>
     </Section>

     <Section type="datatable">
       <Field displayName="Employee Name" name="identityName"/>
     </Section>

     <Section name="Edit These Fields">
        <Field displayName="Region Value" name="region" required="true"
          type="String">
         <AllowedValuesDefinition>
           <Script>
             <Source>
          import java.util.ArrayList;
    import sailpoint.api.*;
    import sailpoint.object.*;

    List regions = new ArrayList();
    QueryOptions qo = new QueryOptions();

    qo.setDistinct(true);
    qo.addOrdering("region", true);

    List props = new ArrayList();
    props.add("region");

    Iterator result = context.search(Identity.class, qo, props);
    while (result.hasNext()) {
  Object [] record = result.next();
String region= (String) record[0];
regions.add(region);
    }
    return regions;
             </Source>
           </Script>
         </AllowedValuesDefinition>
         <ValidationScript>
             <Source>
    // validation variable comes in as "value"
          import sailpoint.tools.Message;
               List messages = new ArrayList();
               if(value.length() &lt; 6) {
                  Message msg = new Message();
                  msg.setKey("New region must be at least 6 characters.");
                  messages.add(msg);
               }
               return messages;

             </Source>
         </ValidationScript>
       </Field>
    </Section>
  </Form>
```

```
 </Approval>
</Step>
```

Report Forms

Report forms are used to display report-specific filters to the user in the Edit Report window. The form must be created as an independent Form object and referenced from the report definition in a <ReportForm> element.

At run-time, the form is combined with the Report Form Skeleton, which defines the Standard Properties and Report Layout pages. Each section named in the form is created as its own Report Properties page, displayed between the Standard Properties and Report Layout pages. The page name, shown in the **Sections** list and at the top of the form, is specified as the Section's label attribute.

```
   <Form name="Uncorrelated Accounts Report Custom Fields">
      <Section label="Uncorrelated Accounts Parameters" name="customProperties">
<Field displayName="report_input_correlated_apps" filterString="logical==false
&amp;&amp;
authoritative==false"
helpKey="rept_input_uncorrelated_ident_report_correlated_apps"
name="correlatedApps" type="Application" value="ref:correlatedApps"/>
      </Section>
    </Form>
```

An example of a simple report form is shown below. It contains only one section, formatted in two columns with several datatypes represented (dates, objects, and boolean). The displayName and helpKey values on this report are localizable message keys. The values are all pulled from the TaskDefinition's input arguments, if any are provided there, to set the fields' default values.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Form PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<Form created="1346776069392" id="4028460239921ba40139921bf510019a"
modified="1346776080142"
name="Application Owner Access Review Report Form">
  <Section columns="2" label="rept_cert_custom_section_title"
name="customProperties">
    <Field displayName="rept_cert_field_create_start"
helpKey="rept_cert_help_create_start"
name="createStartDate" type="date" value="ref:createStartDate"/>
    <Field displayName="rept_cert_field_create_end"
helpKey="rept_cert_help_create_end"
name="createEndDate" type="date" value="ref:createEndDate"/>
    <Field displayName="rept_cert_field_signed_start"
helpKey="rept_cert_help_signed_start"
name="signedStartDate" type="date" value="ref:signedStartDate"/>
    <Field displayName="rept_cert_field_signed_end"
helpKey="rept_cert_help_signed_end"
name="signedEndDate" type="date" value="ref:signedEndDate"/>
   <Field displayName="rept_cert_field_due_start" helpKey="rept_cert_help_due_start"
name="dueStartDate"
type="date" value="ref:dueStartDate"/>
    <Field displayName="rept_cert_field_due_end" helpKey="rept_cert_help_due_end"
name="dueEndDate"
type="date" value="ref:dueEndDate"/>
    <Field displayName="rept_cert_field_apps" helpKey="rept_cert_help_apps"
multi="true"
name="applications" type="Application" value="ref:applications"/>
    <Field displayName="rept_cert_field_tags" helpKey="rept_cert_help_tags"
```

```
multi="true" name="tags"
type="Tag" value="ref:tags"/>
    <Field displayName="rept_cert_field_cert_group"
helpKey="rept_cert_help_cert_group" multi="true"
name="certificationGroups" type="CertificationGroup"
value="ref:certificationGroups"/>
    <Field displayName="rept_cert_field_show_exclusions"
helpKey="rept_cert_help_show_exclusions"
name="exclusions" type="boolean" value="ref:exclusions"/>
  </Section>
</Form>
```

This form is rendered as shown in the **Report Properties** section of the **Edit Report** window.

In report forms, sections can be created without Field definitions, allowing the report's taskDefinition's initialization rule/script to create the form fields. Several of the standard reports, for example, use an initialization rule to generate a pages of Application and/or Identity attribute filters based on the installation's system data, the defined standard and extended attributes, so the report forms themselves are defined with empty sections. The Privileged Access Report form provides an example of a dynamically built form.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Form PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<Form created="1346776069939" id="4028460239921ba40139921bf73301b0"
modified="1346776080079"
name="Privileged Access Report Form">
  <Section columns="2" label="rept_priv_access_section_priv_account_attrs"
name="Privileged Account
Attributes">
    <Attributes>
      <Map>
        <entry key="subtitle" value="rept_priv_access_section_instructions"/>
      </Map>
    </Attributes>
  </Section>
  <Section columns="2" label="rept_priv_access_section_account_props" name="Account
Properties">
    <Field columnSpan="1" displayName="rept_identity_roles_field_app"
helpKey="rept_identity_roles_helpN_app" multi="true" name="applications"
type="Application"
value="ref:applications"/>
  </Section>
  <Section columns="2" label="rept_priv_access_section_identity_props"
name="Identity Properties"/>
  <Section columns="2" label="rept_priv_access_section_identity_extended_props"
name="Identity Extended
Properties"/>
</Form>
```

Form Models

Form models are used to simplify that process of passing values between the workflow variables and the form. Form models enable the specification of a Map through which a set of variables can be handed to the form by the workflow. The model is defined in the workflow (or pre-defined model is used), enabling the workflow and form to pass a collection of variables at one time through the specified model. The form renderer is set up to use

the model so form fields can name the desired attribute directly without having to reference the model name as well.

Since actions in workflows often center around Identities, a map for the identity object, called IdentityModel, is pre-built in IdentityIQ. A workflow library method - getIdentityModel - can be called by a workflow step to create an IdentityModel map to use in a subsequent step that renders a form. To create an empty map, call this method with no arguments. To pre-populate the map with an identity's current values, specify an identity name or ID as an argument to the step. This method is used with no arguments in the new self-service registration workflow to prepare to create a new identity from the data the user enters on the form.

For an example of the simplification offered by a form model: A workflow form needs to display and permit the user to edit 10 identity attributes. Without form binding, all 10 would have to be defined as individual business process variables and all 10 would have to be sent to and returned from the approval in the workflow. The form would also require all 10 to be defined as form arguments. With a model, the whole Identity can be automatically stored in a single business process variable with a single method call, only one variable (identityModel) must be passed to and returned from the form, and no form arguments need to be defined at all.

Use these steps to use the IdentityModel in a workflow form:

1. Go to Se**tup -> Business Process -> Process Variables** tab.
2. Define a process variable in the workflow (identityModel).
3. In an early step in the workflow, initialize and populate the identityModel by calling the getIdentityModel method in the IdentityLibrary workflow library. Specify the identityModel process variable as the Result Variable for that step.

   To pass an identity name or ID to the method, specify it as an argument to the step (identityName or identityId).
4. In the approval that contains the form, create an argument called workItemFormBasePath and specify the identityModel process variable as its value. The form base path is understood by the form renderer and is automatically applied to permit the form access to the map fields. This enables the passing of the model to and from the form.
5. Reference the components of the identityModel in the form as though they were passed as individual variables, for example, as "firstname", not as "identityModel.firstname".

   Note: When a base path is specified as a form argument, the form renderer assumes all fields on the form are accessed through that base path, so all attributes to be included in the form or returned from it must be included in the model.

   ```
   <Section>
   <Field displayName="user_name" name="name" required="true"
   type="string"/>
   <Field displayName="first_name" name="firstname" required="true"
   type="string"/>
   <Field displayName="last_name" name="lastname" required="true"
   type="string"/>
   <Field displayName="email" name="email" required="true" type="string"/>
   …
   ```
6. (Optional) To provision changes to the identity based on the form, call the buildPlanFromIdentityModel() method in the Identity Library. This examines the versions of the model passed to the form and back from it, identifies differences between them, and creates a provisioning plan to make the required changes.

Refer to the LCM Registration workflow, which ships with IdentityIQ Lifecycle Manager, for a full example of implementing the identityModel.

No other models currently ship with the product, but custom models can be created through some manual coding in the initialization stage.

1. Declare the model variable as a process variable (same as the identityModel), for example appModel.
2. Initialize the model manually, since no library method exists to populate custom models. Instead of a method call in the initialization step, the step executes a script or rule written to populate the desired data into a HashMap that is stored in the custom model variable.
3. Specify the custom model variable as the workItemFormBasePath argument to the workflow's form step.
4. Reference components in the custom model by name in the form. As with identityModel, no reference to the base path should be specified in the form field names.

Identity Model Structure

The IdentityModel map delivered with IdentityIQ contains the following entries:

- all standard Identity attributes and all extended Identity attributes (most as strings; lists when multi-valued)
- detectedRoles (List)
- assignedRoles (List)
- manager (String name, rather than ID
- info map which contains:
  - omanager map (includes ID, name, and displayName of Manager Identity)
- lastRefresh, lastLogin, and passwordExpiration dates
- isWorkgroup, managerStatus, correlated, and correlatedOverriden flag values
- assignedScope name and controlsAssignedScope flag value
- transformerOptions (map of primer identityName or identityId used to populate the IdentityModel)
- class (sailpoint.object.Identity)
- transformerClass (sailpoint.transformer.IdentityTransformer)

Accessing Identity Model Attributes

Any identity model attributes can be displayed on a form or set based on data entered in a form field by supplying the model attribute name as the field name.

Access any single-valued attribute at the top level by specifying its name in the field's name attribute:

```
<Field displayName="first_name" name="firstname" type="string"/>
```

To display the contents of a multi-valued extended identity attribute, use the following syntax. Multi-value extended identity attributes are shown in the identityModel as a list of string values.

```
<Field displayName="Cost Centers" multi="true" name="costcenter" type="string"/>
```

Access any nested attribute, for example, those with a map within the map, using dot notation:

```
<Field displayName="Manager ID" name="info.manager.id" type="string"/>
```

Note: Values in the info map should not be altered through the form, as they will not be updated in the model; they are treated as read-only data that provides supplementary data for the corresponding top-level attribute, and they are automatically refreshed based on updates to that top-level attribute.

Display the contents of an object list in the map, such as assignedRoles, detectedRoles, or workgroups, on a form by creating it as a combo box. Do this by specifying the type as the correct object type and specifying multi="true":

```
<Field displayName="Detected Roles" filterString="type==&apos;it&apos;" multi="true"
name="detectedRoles" type="iiq.object.Bundle"/>
```

```
<Field displayName="WorkGroups" filterString="workgroup == true" multi="true"
name="workgroups" type="Identity"/>
```

The application of a filterString to the workgroups list ensures that only workgroup identities display.

The links list contains a map for each link (account) held by the identity. Access attributes inside that list by referencing the name of the desired link and using dot notation to traverse the map:

```
<Field displayName="App Owner" name="links['HR_Employees'].sys.nativeIdentity"
type="string"/>
```

In development and debugging, it can be helpful to examine the identityModel in XML or as a string representation to clearly see its structure. The identityModel is visible in the workflowCase for any workflow where it is used and is printed to stdout if the trace variable is set to true for the workflow. It can be printed as a string from a workflow step with a System.out.println(identityModel.toString()); statement.

Referencing a Form Model

Form models can be accessed by rules and scripts within workflows or forms within workflows using the `$()` parsing tokens, for example, `$(identityModel.name)`. When variables are referenced with this syntax, the ScriptPreParser expands the short hand path references into the proper `MapUtil.get()` reference. This notation can be used in scripts run from fields, variables, steps, transitions or step actions. The script can explicitly specify the full path to the variable in the model, for example, `$(identityModel.name)`, or it can reference the variable directly when a modelBasePath has been defined, for example `$(name)`.

Note: In order to set a basePath in a form, an argument named modelBasePath (defined as Rule.MODEL_BASE_PATH) must be set declaring the path to be used for all the expanded variables in the form. For example, <Arg name='modelBasePath' value='identityModel'/>. The modelBasePath does not have to be specified at the top level; for example, it can point to a list or map within the top-level map such as <Arg name='modelBasePath' value='identityModel.links[AD]'/> (this is the map representing the user's AD account link).

Note: This $() notation can only be used for retrieving values from the model, it cannot be used to set or change values in the model.

Syntax

Use the following syntax rules when writing references within the scripts:

- A dollar sign with parentheses `[$()]` is used as the parsing token to indicate what contents should be expanded. For example, `$(foo.bar)`.
- Double quotes are valid when enclosing spaces within the variable: `$(foo."bar baz")`. However an expansion token within a quoted string is not processed: `"$(not.expanded)"`.
- Brackets can be used within a variable to access elements in a list: `$(foo.bar[baz=bingo].buzz)` `$(foo.bar[baz="path with spaces"].buzz)`
- When the modelBasePath is set to a sub-map or list within the model, the forward slash escape character (/) can be used to jump to the root of the basePath. This escape character must be the first character after the expansion token. If basePath is set to 'identityModel.links[AD]' and the desired reference is for identityModel.firstname the variable would be written as `$(/firstname)` which would be converted to iiq`.tools.MapUtil.get(identityModel, "firstname")`. Otherwise `$(firstname)` is converted to iiq.tools.MapUtil.get(identityModel, "links[AD].firstname").
- If no basePath is set and the variable only contains a single word, no expansion occurs and a warning is written to the log indicating a possible error condition.

Example Syntax

The following are all valid:

No base path:

- `$(foo.bar)` —> iiq`.tools.MapUtil.get(foo, "bar")`
- `$(foo."bar baz")` —> iiq`.tools.MapUtil.get(foo, "\"bar baz\"")`
- `$(foo.bar[baz="path with spaces"].buzz)` —> iiq`.tools.MapUtil.get(foo, "bar[baz=\"path with spaces\"].buzz")`

Base path = 'foo'

- `$(foo.bar)` —> iiq`.tools.MapUtil.get(foo, "bar")` (assuming basePath is set to 'foo'. This respects the basePath and does not try to find a "foo" attribute within the "foo" map)
- `$(bar)` —> iiq`.tools.MapUtil.get(foo, "bar")` (assuming basePath is set to 'foo')

Base path = 'foo.bar[AD]'

- `$(baz)` —> iiq`.tools.MapUtil.get(foo, "bar[AD].baz")` (assuming basePath is set to 'foo.bar[AD]')
- `$(/baz)` —> iiq`.tools.MapUtil.get(foo, "baz")` (assuming basePath is set to 'foo.bar[AD]')

System Administration

This section contains the following information:

Chapter 17: Configure Risk Scoring

Use the risk scoring configuration pages to define the algorithms used by IdentityIQ to determine risk scores for identities and applications within your organization. Risk scores are used throughout the product to highlight high risk users and accounts and to trigger notices when configured to do so.

To access Risk configuration options, go to **Identities -> Identity Risk Model** or **Application -> Application Risk Model**. Configuring risk scoring requires the assignment of administrative capabilities within IdentityIQ.

To configure risk scoring for identities and applications refer to following:

- "Identity Risk Score Configuration" on page 275
- "Application Risk Score Configuration" on page 279

Identity Risk Score Configuration

IdentityIQ uses a combination of base access risk and compensated scoring method to determine the overall Identity Risk Scores, or Composite Risk Score, used throughout the product. You configure Baseline Access and Composite risk scoring for applications by navigating to the Define > Application Risk Model area of the product interface.

Base access risk is a measure of inherent user access risk. Base risk scores are set on each role, entitlement, and policy defined. This type of score ranges from 0 (lowest risk) to 1000 (highest risk). The account weight assigned to any additional entitlements that are assigned to an identity also have an impact base risk scores. Account weights are factored in to the entitlement baseline access risk scores.

IdentityIQ applies a series of compensating factors to each base risk score to calculate compensated scores. These compensated scores are then weighted using a maximum contribution percentage and combined to form an overall Composite Risk Score for each user.

The compensating factors and weighted values enable IdentityIQ to accurately identify high-risk users based on more than just the roles they are assigned within your enterprise.

For example, a user assigned only low risk roles might be considered high risk if they have never been included in a certification process or the roles they do have are in violation of separation of duty policies.

Scoring Definitions

There are a number of scores, or types of scores, that contribute to the overall Identity Risk Score, or Composite Risk for each IdentityIQ user. The basic scores that are used to determine the overall score are:

Table 39—Access Risk Scoring Definitions

Score	Definition
Base Risk Score	The score assigned to each role, entitlement, or policy violation.
Total Base Risk Score	The total score of all base risk scores of the same component type on a per user basis. For example, add the base risk scores for all roles assigned to a specific user together to determine the role total base risk score.

Table 39—Access Risk Scoring Definitions

Score	Definition
Compensated Risk Score	The value of the base risk score for a component multiplied by the compensating factor for that component type.
Total Compensated Risk Score	The Total Base Risk Score for a specific component type multiplied by the Compensated Risk Score for that component type.
Composite Risk Score or Identity Risk Score	The overall risk score for a user after the composite weighing, or maximum contribution to total score factor, is applied to the total compensated risk scores for each component. The time since the last certification was performed on the user is also figured into this score with the total compensated scores for role, entitlement, and policy violation.

Use the sliding bars or manually enter a value, to define scoring on each panel.

Use the following tabs to create risk score factors for your enterprise:

- Baseline Access Risk Tab — apply base risk scores to roles, entitlements and policy violations. See "Identity Baseline Access Risk Tab" on page 276.
- Composite Scoring Tab — apply compensating factors to base risk scores. See "Identity Composite Scoring Tab" on page 278.

Identity Baseline Access Risk Tab

The Baseline Access Risk score is a measure of inherent risk. A user's Baseline Access Risk score rarely changes because their role within the enterprise is the primary factor in defining the score. This type of score ranges from 0 (lowest risk) to 1000 (highest risk).

Select one of the following options to define how IdentityIQ calculates base access risks. Each role, entitlement, and policy violation is assigned a score that falls into a band. The number of bands is configured on the Advanced Configuration page and applies to the entire IdentityIQ application.

- “Role Baseline Access Risk” on page 276
- “Entitlement Baseline Access Risk” on page 277
- “Policy Violation Baseline Access Risk” on page 277

To configure baseline access risk scores for role, entitlement, and policy violation access, navigate to the Define > Identities Risk Model area of the product interface and select the Baseline Access Risk tab.

Role Baseline Access Risk

Role Baseline Access Risk score is calculated based on the roles correlated to the identity. This list contains every role defined in IdentityIQ. To limit the number of items displayed in the list, filter the list by role name and type.

Table 40— Role Baseline Access Risk Configuration Column Descriptions

Column	Description
Name	The name of the role.
Type	The role type as defined when the role was modeled.
Description	The description of the role as defined when the role was modeled.

Table 40— Role Baseline Access Risk Configuration Column Descriptions

Column	Description
Risk Level	The current risk level assigned to the role.

Click on a role to display the configuration panel to see the role details and set or modify the risk level. Use the slider control to set the risk level or enter a value in the field on the right.

Entitlement Baseline Access Risk

Entitlement Baseline Access Risk score is calculated based on the additional entitlements correlated to an identity. Additional entitlements are entitlements that are assigned to a user, but are not part of any of the roles assigned to that user.

Entitlements fall into two categories: Permissions and Attributes. A Permission is a privilege, such as create, read, update, delete, and execute. Attributes are customized user characteristics made up of an attribute/value pair, such as group/Administrators. A risk score is configured for each Permission and Attribute/Value pair in the system. A user's Entitlement BAR score is determined by summing the risks associated with each of the additional entitlements that they hold.

Use this page to add applications to the list and to work with the entitlements on each. The Entitlement Baseline Access Risk Configuration page contains the following information:

Table 41— Entitlement Baseline Access Risk Configuration Column Descriptions

Column	Description
Application	The name of the application with which the entitlements are associated.
Account Weight	The default score assigned to any identity that is assigned entitlements on this application. Account Weight scores are not compensated. This score is not applied to the identity risk score if the entitlements assigned to the user are, either all used as part of roles assigned to the user, or if the risk score for all of the entitlements assigned to the user are zero based on certification rules.
Permissions	Click in this column to modify the weight assigned to the permissions for the associated application. Use the sliding bar or enter a value in the field on the right to modify permission weight.
Attributes	Click in this column to add, delete or modify the weight assigned to the attributes for the associated application. Select an attribute from the drop-down list, type an attribute name, and click **Add** to assign a weight to a new attribute, or modify and existing attribute in the list. Select an attribute using the check-boxes on the left and click **Delete** to remove an attribute from the list.

To add an application to the list, select an application from the drop-down list on the bottom of the page. The list contains all of the application configure to work with IdentityIQ that are not currently on the list. Use the Permissions and Attributes columns to add entitlements to applications for risk tracking.

Policy Violation Baseline Access Risk

Policy Violation Baseline Access Risk score is calculated using policy violations that are detected for a user based on defined policy rules. A risk score is configured for every rule in each policy or for the policy if no rules apply. This score is calculated by taking the sum of the risks associated with every policy or rule that the user violates.

Use the Policy Violation Baseline Access Risk page to view and modify the risk level associated with each policy or policy rule defined. The page is divided into tables based on policy type. If the policy does not contain rules, set the risk level for the entire policy. Use the slider or type a value in the field to the right.

Identity Composite Scoring Tab

Use the Composite Scoring tab to assign value to the compensating factors for each base component used to calculate the composite risk scores for users. You can also define the maximum contribution of each component to the total score. The maximum composite risk score is 1000. Use the Maximum Contribution to Total Score value to control the impact of compensated scores on composite scores.

Use the Composite Scoring tab to define the maximum impact of a total compensated score on a user's Composite Risk Score. For example, if the time since the last certification on an identity is considered low risk, you can set the Certification Age to a low value, such as 20% so that even at its maximum value that component only contributes 200 points of the total 1000. If, however, policy violations are considered high risk, you can set the Separation of Duty Violation Compensated Score to 100% so that policy violations move users into the high-risk category quickly.Use the Composite Scoring tab to define the maximum impact of a total compensated score on a user's Composite Risk Score.

Table 42— Identity Composite Scoring Configuration

Category	Compensating Control
Role Compensated Score	Based on applying the following compensating factors to each role base score: The user's role has never been certified before The user's role is approved The user's role was allowed as an exception An allowed exception on the user's role has expired Revocation of the user's role is pending Activity monitoring is enabled on one or more applications associated with the user's role
Entitlement Compensated Score	Based on applying the following compensating factors to each entitlement base score: The user's entitlement has never been certified before The user's entitlement is approved The user's entitlement was allowed as an exception An allowed exception on the user's entitlement has expired Revocation of the user's entitlement is pending Activity monitoring is enabled on one or more applications to which the user's entitlement applies

Table 42— Identity Composite Scoring Configuration

Category	Compensating Control
Policy Violation Compensated Score	Based on applying the following compensating factors to policy base score: The user's violation has never been certified before The user's violation was allowed An allowed exception on the user's policy violation has expired The user's policy violation remains uncorrected Activity monitoring is enabled on the applications on which the user's violation occurred
Certification Age Score	Based on applying the following compensating factors to an expired certification: The risk score starts increasing this many days after the latest certification The risk score reaches its maximum value this many days later
Inactive User Score	looks for inactive users. When this score is enabled any identity is found to be inactive, a default risk score of 500 is assigned for this score component

To configure composite risk scoring for identities, navigate to the Define > Identity Risk Model area of the product interface and select the Composite Scoring tab.

Application Risk Score Configuration

IdentityIQ uses a combination of Component and Composite scoring to determine the overall application risk scores used throughout the application. You configure Component and Composite risk scoring for your applications by navigating to the Define > Application Risk Model area of the product interface.

All scores are calculated by first determining the percentage of accounts that have the qualities tested by the component score. For example, if 10 out of 100 accounts are flagged as service accounts, then the raw percentage is ten percent (.10). This number is then multiplied by a sensitivity value which can be used to increase or decrease the impact of the original percentage. The default sensitivity value is 5 making the adjusted percentage fifty percent (.50). This final percentage is then applied to the score range of 1000 resulting in a component score of 500.

After the component score is calculated a weight, or compensating factor, is applied to each component score to determine the amount each contributes to the overall risk score for the application. The resulting score is the composite score. For example, a few violator accounts might increase risk more than many inactive accounts.

To view the currently configured risk information for an application, go to Application Definition page, click on a listed application, and then select the Risk tab.

Use the following tabs to create risk score factors for your enterprise:

- Baseline Access Risk Tab — apply base risk scores to roles, entitlements and policy violations. See "Application Component Scores Tab" on page 280.
- Composite Scoring Tab — apply compensating factors to base risk scores. See "Application Composite Score Tab" on page 280.

Application Component Scores Tab

Use the Component Scores tab to define the values for each account or component.

Service, Inactive, and Privileged component scores look for links that have a configured attribute. For example, the component `service` with a configured value `true`.

The Dormant Account score looks for a configured attribute that is expected to have a date value, for example `lastLogin`. This algorithm has an argument, `daysTillDormant`, that defaults to thirty (30). If the last login date is more than thirty (30) days prior to the current date, the account is considered dormant and is factored into the risk score.

The Risky Account score looks for links whose owning identity has a composite risk score greater than a configured threshold. The default threshold is five hundred (500).

The Violator Account score looks for links whose owning identity has a number of policy violations greater than a configured threshold. The default threshold is ten (10).

Note: If you check Disabled, the component is not used to determine the application risk score.

To configure component risk scoring for applications, navigate to the Define > Application Risk Model area of the product interface and select the Component Scores tab.

Application Composite Score Tab

Use the Composite Scoring tab to apply a weight or compensating factor for each component. Specify the percentage of contribution for the component scores.

To configure composite risk scoring for an application, navigate to the Define > Applications Risk Model area of the product interface and select the Composite Score tab.

Chapter 18: Partitioning

Note: Partitioning is not available on all task or certifications. Partitioning is available for Account Aggregation, Identity Refresh, and Manager Certification generation.

Note: Partitioning is not available on all application types. Partitioning is controlled by both the configuration of the applications you are using and the configuration of the connectors used to communicate with those applications.

Partitioning is used to break operations into multiple pieces, or partitions. Each partition is then placed in a global queue, and machines, or hosts, in a cluster compete to execute the partitions in the queue. Machines are added or removed from the cluster dynamically with automatic balancing. If a machine fails or is taken down while processing a partition, the partition is placed back into the queue and reassigned to a different machine. A single result object is shared by all partitions and is continually updated so you can monitor the overall progress of the partitioned operation. When all partitions have finished executing the result is marked complete.

Each instance of IdentityIQ includes a Server object containing information about what is happening in that instance. For machines running multiple instances of IdentityIQ, you must give each instance must be assigned a unique iiq.hostname and have a unique Server object.

The Server objects include a heartbeat service and is updated by a new system thread on a regular basis. By monitoring server heartbeats, machines in the cluster can detect when another machine fails. When this happens any partitioned requests that were running on that machine are restarted and picked up by a different machine in the cluster, so that failure of one machine does not terminate an entire long running task.

Server objects include some statistics, such as the number of request threads currently active and the request types that are executing. You can view the state of the machines in your cluster on the Administrator Console page. See, “Monitoring Your Environment” on page 77.

To activate partitioning you must have applications configured for partitioning, connectors configured to work with those applications, and you must enable partitioning when defining an account aggregation or identity refresh task, or scheduling a manager certification.

- Applications are configured as part of the Account Settings on the Configuration tab of the Application Configuration page, see “Configuration Tab” on page 98.
- Connector configuration information is located in the latest IdentityIQ *Integration Guide*.
- Account Aggregation and Identity Refresh information is located in “Account Aggregation” on page 296 and “Identity Refresh” on page 310.
- Scheduling Manager Certification information is located in the IdentityIQ *User’s Guide* or the online help.

Configuring Partitioning Request Objects

Partitioning is also maintained using RequestDefinition objects that are defined for each request type. These objects control how each request-type is processed. For example, these objects define the number of threads that run for each request on the instances of IdentityIQ running on a specific machine. The RequestDefinition objects must be defined on each machine, host, in a cluster.

Note: By default the maximum number of threads to run on each host is set to 1. This number can be changed to maximize performance in your environment, but should be done with caution and only after testing and tuning for your environment.

The following RequestDefinition objects are available:

- Aggregation Partition— define the maximum number of threads to run on each host during account aggregations
- Identity Refresh Partition — define the maximum number of threads to run on each host during identity refresh
- Manager Certification Generation Partition — define the maximum number of threads, the error action, and orphan action for partitioned manager certification requests
- Role Propagation Partition — define the maximum number of threads to run on each host during role propagation

To work with the RequestDefinition objects, go to the IdentityIQ Debug page and select RequestDefinition from the **Select an Object** drop-down list.

Chapter 19: Tasks

Note: Do not open multiple tabs or browsers. Opening multiple tabs might overwrite changes made in the other.

Task are used to automate the processes which build, update, and maintain the information contained within IdentityIQ. Use the basic tasks provide by SailPoint, or create and customize the task to meet the needs of your organization.

Account aggregation tasks scan applications configured to work with IdentityIQ, discover users and entitlements on those applications, and, optionally, correlate those users and entitlements with roles.

Account group aggregation tasks are used to scan applications and aggregate account groups and application object attributes. These are then used for group certification (either permissions or membership) or for displaying of group information in identity certifications.

Activity aggregation tasks scan applications, discover activity, and then correlate that activity with identities enabling you to track and monitor all activity for possible policy violations.

Activity Alert tasks aggregate and process alerts defined in your system. The aggregation tasks collects active alerts and, either launches an activity processing task to launch the associated actions, or stores the alerts until the next processing task is run.

Identity refresh tasks analyze the information collected for each identity to ensure that it is up-to-date and accurate. Among other things, identity scans can detect and report on policy violations and trigger event certifications.

Application score refresh tasks scan the specified applications and run the configured application scoring algorithms to determine application risk score. These scores are then used to update the information displayed on the Application Risk Scores page.

Missing Managed Entitlement Scan creates any entitlement objects for items added after the application was last aggregated.

Policy Scan tasks evaluate policies against identity cubes and update identity score cards with any policy violations discovered.

Refresh Composite Accounts tasks refresh composite accounts for all identities that could, potentially, have a composite account on the applications selected.

System tasks are configured, by default, to run in the background and perform maintenance, refresh system-wide data, and cleanup old or unused information.

Target aggregation task are used to scan applications for unstructured targets.

Access to components is controlled by IdentityIQ Capabilities and scope. Talk to your system administrator if you need access to additional components.

Tasks Page

The Tasks page contains a list of all of the tasks that have been created. The first time you access the Task page you see the predefined tasks provided by SailPoint. The tasks are grouped into categories based on the task type. You can expand or contract the categories on the grid using the plus (+) or minus (-) icon next to the category name.

Note: Task category headings are only displayed if a task exists in that category.

The task categories are:

- Account Aggregation
- Account Group Aggregation
- Activity Aggregation
- Activity Alerts
- Certification Refresh
- Generic
- Identity
- Scoring
- System
- Target Aggregation

See “Predefined Tasks” on page 284.

Use the search options to limit the number of tasks displayed in the table. Entering a letter, or partial name, in the **Search** field displays any tasks with names containing that letter pattern.

Use this page to create, edit, run, schedule or delete task.

See “Working with Tasks” on page 286.

The Tasks page contains the following information:

Table 43—Tasks Page field descriptions

Field Name	Description
Name	The name of the task as defined when the it was created.
Description	A brief description of the specific task.

Predefined Tasks

SailPoint provides a number of predefined tasks that can be run to aggregate, correlate and refresh information within your enterprise.

Note: The predefined tasks are not templates that can be used to create new tasks. Changes made to these tasks overwrite exiting information. To create new task you must use the New Task drop-down menu at the bottom of the page.

Note: These tasks are defined to perform specific functions within your enterprise. Deleting or altering these tasks might have negative affects on the performance of IdentityIQ.

SailPoint provides the following tasks:

Generic Tasks:

- Refresh Role Indexes — Update all role information and create the indexes needed to perform role searches. You must run this task before performing any role searching.

Identity Tasks:

- Check Active Policies — Scan all users for policy violations and update Identity Risks Scores. Edit this task to specify how policy violations are handled when detected.
- Prune Identity Cubes — Delete identities that have no account links and have no important references. Identities in any of the following states are protected:
  - Marked protected
  - Is a manager (managerStatus flag true)
  - Has capabilities
  - Bundle, Application, Workitem, or TaskResult owners
  - Work item requestor
  - Application secondary owner
  - Application remediator
  - Creator of a MitigationExpiration

  If the **protectIfCertifying** option is on, identities are protected if they are in an active certification. There is also an option to run the scan for analysis but not delete any identities.
- Refresh Entitlement Correlation — Scan all user entitlements and applications to update role assignments.
- Refresh Groups — Scan all users and update the group indexes for all identity groups.
- Refresh Identity Cube — Perform a full refresh of the identity cubes for all users. Edit this task to specify which portions of the identity cubes are refreshed by this task.
- Refresh Risk Scores — Scan all users and update the Identity Risk Scores for each.

Scoring Tasks:

- Refresh Application Scores — Runs the scoring algorithms against all specified applications and updates the Application Risk Scores page.
- Refresh Role Scorecard — Analyzes each role in the system and collects statistics about them.

System Tasks:

- Check Expired Mitigations — Scans all users for temporary exceptions allowed in a certification that have now expired. The original certifier can optionally be notified when allowed exceptions expire.
- Check Expired Work Items — Scans all work items looking for those that need to be canceled or escalated to a different user.
- Complete Orphaned Identity Requests — Removes completed requests for roles that exist in your system.
- Effective Access Index Refresh — Refreshes or rebuilds the effective access index.
- Full Text Index Refresh — Builds and refreshes the index files used for full text searches on defined fields on the access request pages of the Lifecycle Manager. The index files are rebuilt each time this task is run.
- Perform Identity Request Maintenance — Prunes old identity request objects and scans unverified access requests to check for provisioning completeness.
- Perform Maintenance — Prunes identity snapshots, task results, and certifications, escalates orphaned work items, and performs other background maintenance tasks.

  Note: Electronically signed objects are not affected by this task.

- Remove Orphan Role Requests — Stops and removes requests for roles that no longer exists in your system. For example, if the sunset date for a role passes before the request is processed, this task removes that request.
- Role Overlap Analysis — Performs impact analysis on a specified role. The task result name is annotated with the name of the selected role so you can tell multiple analysis results apart.
- Synchronize Roles — Synchronizes IdentityIQ roles with the roles on the identity management systems that are configured to work through a provisioning provider.

Working with Tasks

To run or execute a task, right-click on the task name and select **Execute** or **Execute in background**. **Execute** displays a pop-up progress window and opens the Task Result page when it is complete. **Execute in background** launches the task in the background and you must go to the Tasks Results page to track progress or view the finished task.

See “Task Results” on page 291.

Tasks that require sign off generate work items and email notifications that are assigned to the designated signers. Sign off decisions are retained with the task results for tracking purposes.

See “How to Complete Task Work Items” on page 322.

To create a new task, use the **Create new task** drop-down list to select a task type and display the New Task page.

Note: **The predefined tasks are not templates that can be used to create new tasks. Changes made to these tasks overwrite exiting information. To create new task you must use the Create New Task drop-down menu at the bottom of the page.**

See “How to Create a New Task” on page 286.

To edit an existing task, click a task or right-click and select **Edit** to display the Edit Task page.

See “How to Edit a Task” on page 288.

To schedule a task, right-click and select **Schedule** from the drop-down list to display the New Schedule dialog. You can schedule task to run once, hourly, daily, weekly, monthly, quarterly or annually to meet the requirements of your enterprise and auditors. Go to the Scheduled Tasks tab to view or edit existing schedules.

See “How to Schedule a Task” on page 289 and “Scheduled Tasks” on page 290.

To terminate a currently running task, access the Task Results page, right-click on the task to terminate and select **Terminate** from the drop-down menu. You are asked to confirm the termination request. Task that are currently running are flagged as pending in the Date Complete column of the Task Results table.

To delete a task, right-click the task and select **Delete** from the drop-down menu. Click **Yes** on the confirmation pop-up to delete the task. When you delete a task from the Tasks table, all associated task results are deleted as well.

How to Create a New Task

Use the New Task page to create a task based on the task types provided. Tasks can be as general or specific as required.

See “Task Types” on page 293 for the complete list of tasks types provided.

1. Click or mouse over the Setup tab and select **Tasks** to open the Tasks page.
2. Select a task type from the **New Task** drop-down list to open the New Task page.
3. Enter a name and brief description for the new task.
   This information is displayed on the Tasks table when the new task is saved.
4. Select a **Previous Result Action** from the drop-down list. **Delete** is select by default.
   Previous result actions determine how subsequent runs of this tasks react to existing task results.

   Delete — overwrite the previous task results with the new information.

   Rename Old — append a numeral to the name of the old task result and preserve both.

   Rename New — append a numeral to the name of the new task result and preserve both.

   Cancel — cancel the new run of the task.
5. ***Optional***: Allow concurrency. Select **Allow Concurrency** to enable two identical tasks to run at the same time.
   If enabled, allow concurreny appends a numeric value to the name of the task that started second.

   If disabled, the second task is canceled and an exception sent to the requestor.
6. ***Optional***: Require sign off.
   a. Select **Required sign off** to expand the Signoff Properties section.
   b. Select an email notification template from the Initial Notification Email drop-down list. For example, the Task Result Signoff template.
      Templates are created and defined when the application is configured.
   c. Specify the escalation criteria for the sign off request. Use the options displayed to set your escalation parameters.
      None — no reminder emails are sent and no escalation is performed for this work item.
      Send Reminders — email reminders are sent at the configured interval.
      Reminders then Escalation — the configured number of reminders are sent and then the work item is escalated to the signers manager.
      Escalation only — this work item is escalated after the configured interval with no reminders being sent.
   d. Specify the required signers.
      Enter the first letter, or letters, of an identity or workgroup to display a selection list of valid identities or workgroups containing that letter string or click the arrow to the right of the field to display all identities and workgroups and select a signer.
      You can add as many signers as required.
7. ***Optional***: Email task alert.
   Specify the configuration parameters in order to receive the status of different tasks after completion. These settings overwrite the email notification configured at the IdentityIQ Configuration level setting.

- **Email Notification**: Select **Email Notification** to enable the sending of status of task to those recipient whose email is being registered to receive the task status. Use the options displayed to set your notification.

  Disabled — no email notification would be sent.

  Warning — email notification would be sent in case of any warning after completion of task.

  Failure — email notification would be sent in case of task Failure.

Always — email notification would be sent at completion of task irrespective of the task status.

- **Email Notification Template**: (*Applicable only if **Disabled** is not selected*) Select **Task Status** template to send emails on task completion. Templates are customizable.
- **Email Recipients**: (*Applicable only if **Disabled** is not selected*) Select the identity to register them to receive task status notification on emails associated with it.

8. Specify the task options required for the task you are creating.
   Each task type displays unique task options.
   See "Tasks Page" on page 283 for details on each type.
9. Click **Save** to save the new task to the Tasks table.
   — OR —
   Click **Save and Execute** to save the task to the Tasks table and run it immediately.
   The Tasks Results page displays when the report completes.
   See **"Task Results" on page 291**.

How to Edit a Task

Use the Edit Task page to make changes to an existing task.

Note: There is no Save As function on the Edit Task page. Any changes made to an existing task overwrite the task you are editing. You must use the Create New Task drop-down menu to create a new task.

Procedure

1. Click or mouse over the Monitor tab and select **Tasks** to open the Tasks page.
2. Click on a task, or right-click on a task and select **Edit** from the drop-down list to open the Edit Task page.
3. Edit the Name and Description section as needed.

   Note: Changing the name does not save this as a new task and preserve the task being edited. Anything entered here overwrites the existing information.

4. Select a **Previous Result Action** from the drop-down list. **Delete** is select by default.
   Previous result actions determine how subsequent runs of this tasks react to existing task results.
   Delete — overwrite the previous task results with the new information.
   Rename Old — append a numeral to the name of the old task result and preserve both.
   Rename New — append a numeral to the name of the new task result and preserve both.
   Cancel — cancel the new run of the task.
5. ***Optional***: Allow concurrency. Select **Allow Concurrency** to enable two identical tasks to run at the same time.
   If enabled, allow concurreny appends a numeric value to the name of the task that started second.
   If disabled, the second task is canceled and an exception sent to the requestor.
6. ***Optional***: Require sign off.
   a. Select **Required sign off** to expand the Signoff Properties section.
   b. Select an email notification template from the Initial Notification Email drop-down list. For example, the Task Result Signoff template.
      Templates are created and defined when the application is configured.
   c. Specify the escalation criteria for the sign off request. Use the options displayed to set your escalation parameters.
      None — no reminder emails are sent and no escalation is performed for this work item.

Send Reminders — email reminders are sent at the configured interval.
Reminders then Escalation — the configured number of reminders are sent and then the work item is escalated to the signers manager.
Escalation only — this work item is escalated after the configured interval with no reminders being sent.

d. Specify the required signers.
Enter the first letter, or letters, of an identity or workgroup to display a selection list of valid identities or workgroups containing that letter string or click the arrow to the right of the field to display all identities and workgroups and select a signer.
You can add as many signers as required.

7. Add a comma separated list of host names on which to run this task. If multiple hosts are specified, the task manager selects the first active host If there are no active hosts, or if an incorrect host name is given, the task terminates, and an error message is left in the result.
8. Optional: Email task alert.
Specify the configuration parameters in order to receive the status of different tasks after completion. These settings overwrite the email notification configured at the IdentityIQ Configuration level setting.
   - Email Notification: Select Email Notification to enable the sending of status of task to those recipient whose email is being registered to receive the task status. Use the options displayed to set your notification.
   - Disabled — no email notification would be sent.
   - Warning — email notification would be sent in case of any warning after completion of task.
   - Failure — email notification would be sent in case of task Failure.
   - Always — email notification would be sent at completion of task irrespective of the task status.
   - Email Notification Template: (Applicable only if Disabled is not selected) Select Task Status template to send emails on task completion. Templates are customizable.
   - Email Recipients: (Applicable only if Disabled is not selected) Select the identity to register them to receive task status notification on emails associated with it.
9. Edit the task options required for the task you are creating.
Each task type displays unique task options.
See **"Task Types" on page 293** for details on each type.
10. Click **Save** to save the new task to the Tasks table.
— OR —
Click Save and Execute to save the task to the Tasks table and run it immediately.
The Tasks Results page displays when the report completes.
See **"Task Results" on page 291**.

How to Schedule a Task

Use the New Schedule dialog to schedule tasks to run during times of low business activity. Schedule recurring tasks as needed to maintain routine compliance within your enterprise.

The New Schedule dialog enables you to assign a unique name and description to the task schedule. This information is stored on the Scheduled Tasks page and displays in the Task Results table.

See "Scheduled Tasks" on page 290 and "Task Results" on page 291.

Procedure

1. Click or mouse over the Monitor tab and select **Tasks** to open the Tasks page.

2. Right-click on a task name and select **Schedule** from the drop-down list to open the New Schedule dialog.
3. Enter a unique name and description for this schedule task.

   Note: Task that run across time zones run at the time scheduled, relative to the time zone in which they are scheduled. For example, a task scheduled to run at 4:00 PDT runs at 1:00 EDT.

4. Enter the date and time to launch the first execution of this task.
   You can enter the date manually, or click the ... icon to select a date from the calendar.

   — OR —

   Select the Run Now field to schedule the task to run immediately after clicking Schedule. For recurring task, the task runs at the current time at the specified Execution Frequency.
5. Specify how often this task should run with the **Execution Frequency** drop-down list.
   Subsequent executions of this task occur at the time specified in the First Execution fields.
6. Click **Schedule** to save this scheduled task.
   Go to the Schedule Tasks page to view a list of all scheduled tasks.

   See **“Scheduled Tasks” on page 290**.

Scheduled Tasks

The Scheduled Tasks page contains a list of all scheduled tasks, whether recurring or one-time only. One-time tasks are removed from the list after they are executed.

Tasks that are scheduled, but do not execute due to malformed task definitions are displayed in the Scheduled Task table with an error icon (!) in the Last Executed column. Tasks that fail in this way never execute and, therefore, never display results on the Tasks Results page. To see details of the execution error, click on the task to display the Edit Schedule dialog. The error information is displayed in the **Last Launch Error** field. Errors of this type should only occur for custom task definitions, not for any of the tasks supplied with the product. To correct the error, delete the task schedule, correct the task definition, and recreate the schedule.

Use the Scheduled Tasks page to edit or delete schedules.

Use the search options to limit the number of tasks displayed in the table. Entering a letter, or partial name, in the **Search** field displays any tasks with names containing that letter pattern. Click **Advance Search** to search by task results.See “Working with Schedules” on page 291.

To create a scheduled task see “How to Schedule a Task” on page 289.

The Scheduled Tasks page contains the following information:

Table 44—Scheduled Tasks Column descriptions

Field Name	Description
Name	The name of the schedule as defined on the New Schedule page.
Next Execution	The date and time at which the task is next scheduled to execute.
Last Execution	The date and time at which the task most recently executed. This field displays an error icon (!) for tasks that do not execute due to malformed tasks definitions.
Last Result	The result of the last run of this task, for example Success or Failed.
Owner	The creator of the schedule.

Working with Schedules

To edit an existing schedule, click a schedule name or right-click and select **Edit** to display the Edit Schedule dialog.

See “How to Edit a Schedule” on page 291.

To delete a schedule, right-click the schedule name and select **Delete** from the drop-down menu. Click **Yes** on the confirmation pop-up to delete the schedule.

How to Edit a Schedule

Use the Edit Schedule dialog

1. Click or mouse over the Monitor tab and select **Tasks** to open the Tasks page.
2. Click on the Schedule Tasks tab to display the list of scheduled tasks.
3. Click a schedule name or right-click and select **Edit** to display the Edit Schedule dialog.
4. Edit the name or description for this scheduled task.

   Note: Task that run across time zones run at the time scheduled, relative to the time zone in which they are scheduled. For example, a task scheduled to run at 4:00 PDT runs at 1:00 EDT.

5. Change the date and time to launch the first execution of this task.
   You can enter the date manually, or click the ... icon to select a date from the calendar.

   — OR —

   Select the Run Now field to execute the task immediately after clicking Schedule. For recurring tasks, the task runs at the current time at the specified Execution Frequency.
6. Specify how often this task should run with the **Execution Frequency** drop-down list.
   Subsequent executions of this task occur at the time specified in the **First Execution** fields.
7. Click **Save** to save this scheduled task and return to the Schedule Tasks page.
   See **“Scheduled Tasks” on page 290**.

Task Results

The Task Results page contains a list of all of the tasks that have run or are currently running.

Use the search options to limit the number of tasks displayed in the table. Entering a letter, or partial name, in the **Search** field displays any tasks with names containing that letter pattern. Click **Advanced Search** to filter by start date, end date, or results.

Table 45—Task Results Column Descriptions

Column	Description
Name	The name of the task.
Date Complete	The date and time stamp of when the task completed running.
Result	The result status, **Pending**, **Success**, or **Failed**. A result of **Success** with an exclamation point (!) indicates that there are warnings associated with the results.

Table 45—Task Results Column Descriptions

Column	Description
Signoff	The status of the sign off request for the task results. **None** — no sign off required **Waiting** — sign off request not complete **Signed** — a sign off decision has been made
Owner	The name of the user who launched this task.

Click on a task name in the Tasks Results table to display the Task Results page. Each task type returns information specific to the options that were selected. Tasks that executed with partitioning enabled also display the partitioned results, broken down by the host name of the partitions on which they ran.

Several statistics related to task run length are maintained to help identify tasks that are running longer or shorter than expected. Each time a task is run, we save the start time. When the task is complete we calculate the run time in seconds.

These statistics are not set until the task complete. Until then they are zero. The run time change is a positive or negative integer representing the percent change in run length for this task relative to the average at the time was started. A value of 25 means the task ran 25% longer than average, and a value of -10 means the task was 10% faster.

See “Task Types” on page 293 for details on the information that might be on the Task Results page.

To terminate a currently running task, a task flagged as pending in the Date Complete column, right-click on the task and select **Terminate** from the drop-down menu. You are asked to confirm the termination.

To delete task results, right-click on a result and select **Delete**. Tasks that require a sign off can only be deleted by a user with the Signoff Administrator capability.

If a task was scheduled to run but no results were returned, go to the Scheduled Task tab to ensure that errors did not occur during the task execution.

Task Types

The task types are:

- Account Aggregation — scan all applications, discover users and entitlements on those applications, and then correlate those users and entitlements with roles.
  - See “Account Aggregation” on page 296.
- Account Group Aggregation — scans applications and aggregates account groups and application object types. These are then used for group certification (either permissions or membership) or for displaying group information in identity certifications.
  - See “Account Group Aggregation” on page 298.
- Activity Aggregation — scan all applications, discover activity on the applications, and then correlate that activity with identity cubes. This enables you to track and monitor all activity for possible policy violations.
  - See “Activity Aggregation” on page 299.
- Alert Aggregation — scan applications and aggregates alerts from a set of Alert Collectors. These are then used to generate alert actions.
  - See “Alert Aggregation” on page 300
- Alert Processor — process the aggregated alerts against the alert definitions and launch the appropriate action.
  - See “Alert Processor” on page 300
- Application Builder — create multiple IdentityIQ applications or update the attribute map of an existing IdentityIQ application.
  - “Application Builder” on page 302
- ArcSight Data Export — export data for HP ArcSight Database Connector to an external database table.
  - “ArcSight Data Export” on page 304
- Data Export — generate a de-normalized data report to export to an external database table.
  - “Data Export” on page 301
- Effective Access Indexing — generate an index of any indirect access that was granted through another object. For example a nested group, an unstructured target, or another role.
  - “Effective Access Indexing” on page 302
- Encrypted Data Synchronization Task —re-encrypt data with user-generated encryption key.
  - "Encrypted Data Synchronization Task" on page 307
- Entitlement Role Generator — scans the entitlements in the system and automatically generates a simple role and appropriates a profile for each one that it finds.
  - “Entitlement Role Generator” on page 307
- FIM Application Creator — automatically discover and create FIM Management Agent Applications.
  - “FIM Application Creator” on page 308
- IQService Public Key Exchange — change the public keys that are used for IQService communications
  - “IQService Public Key Exchange” on page 309
- ITIM Application Creator — inspect the IBM Tivoli Identity Manager (ITIM) and retrieve information about the ITIM services (applications). This task auto-generates an application for each service defined in ITIM.

  - “ITIM Application Creator” on page 309
- Identity IQ Cloud Gateway Synchronization — Synchronize the specified objects to the Cloud Gateway.
  - “IdentityIQ Cloud Gateway Synchronization” on page 310
- Identity Refresh — scan all applications, including the IdentityIQ application, to ensure that all identity information is up-to-date and accurate. Refresh identity scans are also used to detect and report on policy violations and trigger event certifications.
  - See “Identity Refresh” on page 310.
- Identity Request Maintenance — scan for completed Lifecycle Manager access requests.
  - See “Identity Request Maintenance” on page 315.
- Missing Managed Entitlements Scan — scan the selected application to create entitlement objects for items added after the application was last aggregated
  - “Missing Managed Entitlements Scan” on page 315
- Novell Application Creator — inspect the Novell IDM application and retrieve information about all connected applications.
  - See “Novell Application Creator” on page 315.
- OIM Application Creator — inspect the OIM application and retrieve information about all connected applications.
  - See “OIM Application Creator” on page 316.
- Policy Scan — runs policies against identity cubes and update identity score cards with any policy violations discovered.
  - See “Policy Scan” on page 316.
- Propagate Role Changes — refreshes identities who have an assigned role whose associated entitlements have changed.
  - “Propagate Role Changes” on page 317.
- Refresh Logical Accounts — is used to refresh composite accounts for all identities that could, potentially, have a composite account on the composite applications selected.
  - See “Refresh Logical Accounts” on page 319.
- Role Index Refresh — updates all role information and creates the indexes needed to perform role searches. You must run this task before performing any role searching.
  - “Role Index Refresh” on page 320
- Run Rule — runs the specified rule with name/value pairs.
  - “Run Rule” on page 320
- Sequential Task Launcher — launches the specified tasks in the order defined. This enables you to launch tasks that must be run sequentially in the proper order without having to schedule each separately based on estimated run times.
  - “Sequential Task Launcher” on page 320
- "System Maintenance" on page 321 — tasks designed to run in the background.
  - See “System Maintenance” on page 321.
- Target Aggregation — scan selected applications for activity targets.
  - See “Target Aggregation” on page 321.

See “Tasks Page” on page 283 for information on working with these task types.

All task types contain the following standard properties:

Table 46—Task Standard Properties

Field	Description
Name	The name of the task as defined when the task was created
Description	Brief description of the task.
Previous Result Action	Previous result actions determine how subsequent runs of this task react to existing task results. **Delete** — overwrite the previous task results with the new information. **Rename Old** — append a numeral to the name of the old task result. **Rename New** — append a numeral to the name of the new task result. **Cancel** — cancel the new run of the task if a task result with the same name exists.
Allow Concurrency	Enable two identical tasks to run at the same time. If enabled, allow concurreny appends a numeric value to the name of the task that started second. If disabled, the second task is cancelled and an exception sent to the requestor.
Require Signoff	Require sign off on the results of this task. Tasks that require sign off generate work items and email notifications that are assigned to the designated signers. Sign off decisions are retained with the task results for tracking purposes.
Host	A comma separated list of host names on which to run this task. If multiple hosts are specified, the task manager selects the first active host If there are no active hosts, or if an incorrect host name is given, the task terminates, and an error message is left in the result.
Number of Runs	The number of times this task has been run.
Average Run Time	The average time in seconds of task runs.
Reset Run Statistics	Reset the statistic if you reconfigure the task and expect the run times to change. When you reconfigure complex tasks like aggregation or refresh, you should consider resetting run statistics. For example, enabling provisioning in the refresh task can profoundly influence run time so statistics should not be diluted by the previous average before provisioning was enabled.
Email Task Alerts	
Email Notification	Select a frequency for email notification to be sent upon task completion. **Disable** — no email notification sent on task completion **Warning** — send an email notification if the task results in a warning **Failure** — send an email notification if the task fails **Always** — always send an email notification upon task completion
Email Notification Template	Select a notification email template from the drop-down list.

Table 46—Task Standard Properties

Field	Description
Email Recipients	The list of users to receive the task completion notification. Use the drop-down arrow to display all identities, or type the first few letters of a name. select names from the list.

Account Aggregation

Account Aggregation tasks scan all applications, discover users and entitlements on those applications, and, optionally correlates those users and entitlements with roles.

Identities that have changed since the last aggregation performed on an application are marked as needing refresh to increase the performance of identity refresh tasks. You can disable this function.

You can perform the correlation functions as part of this task or run account aggregation on all of the applications in your enterprise and then correlate the identity cubes with all of the aggregated information using an identity refresh task.

To perform aggregation on a composite application you must include the composite application and all of the applications that have accounts with which it is associated in the task definition.

Partitioning is available to speed the processing time for account aggregations and level the load on the machines running these tasks. Partitioning is used to break operations into multiple pieces, or partitions. Each partition is then placed in a global queue, and machines, or hosts, in a cluster compete to execute the partitions in the queue. Machines are added or removed from the cluster dynamically with automatic balancing. If a machine fails or is taken down while processing a partition, the partition is placed back into the queue and reassigned to a different machine. A single result object is shared by all partitions and is continually updated so you can monitor the overall progress of the partitioned operation. When all partitions have finished executing the result is marked complete. See, “Partitioning” on page 281.

Note: You must run the Target Aggregation task after this task is complete if you have activity targets set. This tasks removes all targets when it is run.

See “Target Aggregation” on page 321.

The information scanned and updated is determined by the following criteria when the task is created or edited. You can use any combination of options to build a task.

Table 47—Account Aggregator Options

Option	Description
Select an application to scan	The drop-down list of all applications.
Optionally select a rule to assign capabilities or perform other processing on new identities	If accounts are discovered that do not have matching identities in the IdentityIQ application, the rule specified here is used to create a new identity cube. These rules are created during configuration and deployment. Note: Click the “...” icon to launch the Rule Editor to make changes to your rules if needed.

Table 47—Account Aggregator Options

Option	Description
Refresh assigned and detected roles	Scan for newly assigned roles and update identity cubes.
Check active policies	Scan for policy violations and update identity cubes.
Check to updated existing identities, but not to create new identities if a match is not found	Only create links if they can be correlated to an existing identity.
Refresh the identity risk scorecards	Scan for risk score information and update identity risk score cards.
Maintain identity histories	Compare current identity cubes to existing identity cube history, snapshots, and create new snapshots if any changes are discovered.
Enable Delta Aggregation	Enable the connector to aggregate only those accounts that have changed since the last aggregation. This requires support by the connector.
Detect deleted accounts	Compare current aggregated accounts with the accounts previously aggregated report any deleted accounts. **Maximum deleted accounts**: This is the maximum number of accounts that can be flagged for deletion after an account aggregation. If this number is passed, no accounts are deleted from the application.
Refresh assigned scope	Refresh assigned scope based on changes discovered during the aggregation and correlation process.
Disable auto creation of scopes	Do not automatically assign scope to identities as part of this task.
Disable optimization of unchanged accounts	Use this option to force the aggregation of all accounts, changed or unchanged since the last aggregation.
Promote managed attributes	When enabled, any values for entitlement or permissions encountered while running the task automatically get promoted as managed attributes.
Disable auto-creation of applications	Do not automatically create application objects for multiplexed accounts.
Disable marking the identity as needing refresh	Disable marking only identities on which change was detected as need to be refreshed. All identities are included in subsequent identity refresh task.

Table 47—Account Aggregator Options

Option	Description
Enable Partitioning	Enable partitioning of this task across multiple hosts. **Note:** Partitioning is not supported for PE2 based connectors. Partitioning has to be configured on the applications and connectors before this option is valid.
Terminate when maximum number of errors is exceeded	Terminate after the specified number of errors occurs. If the database is available, the task result contains a message indicating that the task was terminated due to excessive errors. If the database is down, the task result cannot be persisted and the task might appear to remain in the pending state. **Maximum errors before termination** Number of errors to tolerate before terminating the task.
Sequential Execution - Terminate an Error	Force applications to aggregate in the listed order and stop the aggregation task if an error is encountered.
Actions to include in the task result	Select the actions performed as part of the aggregation task for which detailed information should be included in the task results. This task performs a number of individual actions on accounts and identity cubes during the aggregation and configuration processes. By default only the final results of the task are included in the task results report. To included detailed information on the actions performed as part of the task, select those actions from the list. **Correlate Manual** — identities with accounts that were manually correlated. These are not changed by the task. **Correlate Maintain** — correlation information has not changed since the last time this task ran. **Correlate New Account** — a new account was discovered for an existing identity and assigned. **Correlate Reassign** — an existing account was reassigned from one identity to another as part of the correlation process. **Create New Identity** — an account was discovered for an identity that did not exist in IdentityIQ. An identity cube was created for the new identity. **Ignore** — an account for a new identity was discovered, but a new identity cube was not created. This might occur if this tasks is configured to perform correlation only. **Remove Account** — an account discovered as part of a previous aggregation was not found during this aggregation. These accounts are removed from IdentityIQ.

Account Group Aggregation

An Account Group Aggregation task scans applications and aggregates account groups and application object types. These results are then used for group certification (either permissions or membership), for displaying group information in certifications, and for performing identity searches.

The information scanned and updated is determined by the following criteria when the task is created or edited. You can use any combination of options to build a task.

Table 48—Account Group Aggregation Options

Option	Description
Select applications to scan	The drop-down list of all applications configured to work with IdentityIQ.
Filter object types to scan	This option is only available for applications on which multiple application objects can exist. This option is not available if you select to scan more than one application. The list of all object types or account groups associated with the selected application. If nothing is selected, all object types and account groups are included. It might become important to scan object types separately if they share attributes.
Enable Delta Aggregation	Enable the connector to aggregate only those account groups or application objects that have changed since the last aggregation. This requires support by the connector.
Detect deleted account groups	Detect and delete any account group or application object that was deleted on the native application since the last aggregation task was run.
Automatically promote descriptions to this locale	The default locale for the description attribute of the account group or application object. This option is used if an existing description locale is not found.
Description attribute (default description)	The attribute that stores the description. This value defaults to the value `description` if this option is not set.

Activity Aggregation

Activity Aggregation tasks scan all applications, discover activity on the applications, and then correlate that activity with identity cubes. Using these tasks enables you to track and monitor activity within your enterprise.

The information scanned and updated is determined by the following criteria when the task is created or edited. You can use any combination of options to build a task.

Table 49—Activity Aggregator Options

Option	Description
Select an activity data source	The drop-down list of all activity data sources discovered by IdentityIQ. If no data source is selected, all available data sources are scanned as part of the task. You applications must be configured to support activity tracking.
Enable storage of the last activity position scanned on the data source	If enabled, the task marks the last activity scanned on the data source so that subsequent runs of this task begin scans at that mark instead of rescanning information. If this option is not enabled, each run of the task scans the entire data source.

Table 49—Activity Aggregator Options

Option	Description
Store uncorrelated activities so they can be re-scanned and correlated at a later time	If enabled, all activity discovered on the data source is stored, even if that activity does not correlate to a user in the application. Storing this activity enables you to add users and update their identity cubes without having to rescan your data sources.

Alert Aggregation

Alert Aggregation tasks scan applications and aggregates alerts from a set of Alert Collectors. These are then used to generate alert actions.

The information scanned and updated is determined by the following criteria when the task is created or edited. You can use any combination of options to build a task.

Table 50—Alert Aggregation Options

Option	Description
Select sources to scan	The drop-down list of all alert data sources discovered by IdentityIQ. If no data source is selected, all available data sources are scanned as part of the task. You applications must be configured to support alert tracking.
Enable Delta Aggregation	If enabled, the task only aggregates alerts that have occurred since the last run of this task. This option requires support from the connectors being scanned.
Process Alerts	If enabled, an Alert Processor task will launch as soon as this Alert Aggregation task is complete. If you select this option, you can limit the alert types processed in a comma separated list, or process all of the alerts collected.

Alert Processor

Alert Processor tasks process the aggregated alerts against the alert definitions and launch the appropriate action.

The information scanned and updated is determined by the following criteria when the task is created or edited. You can use any combination of options to build a task.

Table 51—Alert Processor Options

Option	Description
Optional filter string to constrain the alerts processed	If not set, all are processed.

Table 51—Alert Processor Options

Option	Description
Exclude alerts previously processed	Enable to exclude Alerts that were previously processed.
Optional filter string to constrain the alert definitions to match against alerts	If not set, all Alert Definitions are evaluated.
Enable Partitioning	Enable the task to split into partitions and run across multiple threads and hosts, if available.

Data Export

The Data Export task enables you to export IdentityIQ data to an external database. You can select to export any combination of identity, account, and certification data.

Before you can use the Data Export task, you must create the export database tables on your destination data source.

The task schedule user interface includes a button that generates a customized DDL which you can hand off to a database administrator for execution. Once the data source parameters are entered, click Generate Table Creation SQL.

Table 52—Data Export Options

Option	Description
Datasource Parameters	
Database	Select a database type from the drop-down list.
User Name	Enter the user name parameter of the database table.
Password	Enter the password of the database table.
Driver Class	Enter the driver class used for database.
URL	Enter the URL of the database.
Object Export Options	
Export Identities.	Export identity related data. You can perform a full or incremental export. Use the Export Filter field to apply any database filters.
Export Accounts.	Export account related data. You can perform a full or incremental export. Use the Export Filter field to apply any database filters.
Export Certifications.	Export certification related data. You can perform a full or incremental export. Use the Export Filter field to apply any database filters.

After you complete customizing your report options, click Save for later use or Save and Execute to save the report and run it immediately.

Effective Access Indexing

Effective Access is any indirect access that was granted through another object. For example a nested group, an unstructured target, or another role.

Table 53—Effective Access Index Options

Option	Description
Index Entitlement Targets	Include any effective entitlements associated with application that support effective access searching.
Index Role Targets	Include any effective roles associated with application that support effective access searching.
Index direct role permissions	Include any effective direct role permissions associated with application that support effective access searching.
Index unstructured targets	Include any targets.
Refresh Fulltext Index	Refresh the Fulltext index as part of this task.
Delete all current targets before indexing	Clear an existing Effective Access Index before running this task.

After you complete customizing your report options, click Save for later use or Save and Execute to save the report and run it immediately.

Application Builder

The Application Builder task lets you create multiple IdentityIQ applications, and update existing applications in bulk. The task also includes the ability to perform account and group aggregation for a host using the associated application. It can also export essential data about your existing applications.

The task accepts the inputs required to create or update applications from a `.csv` file. Sample `.csv` files for Linux-Direct and Windows-Local are provided with this task as examples of how input data can be defined. The sample files are located in the `WEB-INF/config` directory of your IdentityIQ installation. You can also use the task's **Read** option to create `.csv` files from your existing applications, to use as models for creating `.csv` files that support the create and update options.

By default, before creating or updating an application on IdentityIQ, a test connection is performed to ensure that the connector is performing correctly. To skip the Test Connection operation, use **Skip Test Connection** in the Application Builder options.

To enable logging for the Application Builder task, add this entry to the `log4j2.properties` file:

```
logger.ApplicationBuilderExecutor.name=sailpoint.task.ApplicationBuilderExecutor
logger.ApplicationBuilderExecutor.level=debug
```

Before using the task to update an existing application, it is recommended that you use the iiq console to export the application definition, in case you need to restore them to their original state.

When you use this task to **Update** an existing application, the update is partial; that is, the update operation can add new attribute definitions to an existing schema, as well as adding a new schema.

Use the account or group aggregation options to trigger a background aggregation task.

Working with Flexible Schemas and Provisioning Forms

The Application Builder task supports including XML definitions in your csv files if you need to create or update flexible account schemas, or provisioning forms. Refer to the sample `.csv` files provided with this task for examples of how a schema definition can be included in the `.csv` file. Sample files are provided in the `WEB-INF/config` directory for Linux-Direct and Windows-Local.

If your input file includes an XML definition of a Provisioning Form, be aware that importing a Provisioning Form definition in a create or update operation will replace all existing Provisioning Forms with the new form as defined in the `.csv`

Table 54— Attribute Builder Options

Option	Description
Application Type	Select an application type from the drop-down list. This is type of application you want to bulk-process. A single application builder task can only process applications of the same IdentityIQ-supported type, such as JDBC, Active Directory, or LDAP
Operation	Select an operation from the drop-down list. **Create** - create multiple applications by providing parameters in the `.csv` file in the specified format **Update** - update existing applications by providing parameters in the `.csv` file in the specified format **Read** - export existing applications to the `.csv` file format. Any existing exported files will be overwritten if the task is run again using the same filename. **Note: The Read operation reads the attribute map, account schema, and provisioning policy of an existing application present in IdentityIQ and exports it to the file path provided in CSV format. You must provide the application type and file path to which the file is to be exported before running the operation.**

File Path	The file path, including file name, for the `.csv` file. For the **Read** option, this is the path to the location and name of the file the task will create. For **Create** and **Update** options, this is the path to the file containing the data for creating or updating your applications; these files must be present on the application server or accessible within the network.
	
	Sample `.csv` files are provided in the `WEB-INF/config` directory for Linux-Direct and Windows-Local:
	`Application-builder_linux.csv`
	`Application-builder-windows-local.csv`

Account Aggregation	Executes the account aggregation task. The account aggregation task is triggered sequentially. The aggregation task will use the following format; the UID (unique identifier) is generated automatically: `<Application type> + <Account Aggregation> + <Current time stamp> + <UID>`

Table 54— Attribute Builder Options

Option	Description
Group Aggregation	Executes the group aggregation task. The group aggregation task is triggered sequentially. The aggregation task will use the following format; the UID (unique identifier) is generated automatically: `<Application type> + <Group Aggregation> + <Current time stamp> + <UID>`
Number of Applications per Aggregation Task	The number of application included in each aggregation task. Default: 10
Skip Test Connection	Skip the default test connection operation.

ArcSight Data Export

Export data for HP ArcSight Database Connector to an external database table.

The ArcSight data export task enables you to export IdentityIQ data to external tables.

Before you can use the ArcSight data export task, you must create the export databases on your destination data source.

The task schedule user interface includes a button that generates a customized DDL which you can hand off to a database administrator for execution. Once the data source parameters are entered, click Generate Table Creation SQL. The task adds the following tables in database:

Table 55—Tables in database

Tables	Description
sptr_arcsight_export	Table to maintain the task execution history.
sptr_arcsight_identity	Table contains exported data of Identity.
sptr_arcsight_audit_event	Table contains Audit Events information.

Table 56—ArcSight Data Export Options

Option	Description
Datasource Parameters	
Database	Select a database type from the drop-down list.
User Name	Enter the user name parameter of the database table.
Password	Enter the password of the database table.
Driver Class	Enter the driver class used for database.
URL	Enter the URL of the database.
Object Export Options	

Table 56—ArcSight Data Export Options

Option	Description
Export Identities	Export Identity related data in ArcSight tables. It provides the following options: Full: Exports all the records irrespective if they were exported earlier. Incremental: Exports only records that are updated since last run of this task. This option can even be selected when running the task for first time. When the task is running for first time, this option exports all records similar to the Full option.
Export Audits	Export Audit Events in ArcSight table. It provides the following options: Full: Exports all the records irrespective if they were exported earlier. Incremental: Exports only records that are updated since last run of this task. This option can even be selected when running the task for first time. When the task is running for first time, this option exports all records similar to the Full option.

After you complete customizing your report options, click Save for later use or Save and Execute to save the report and run it immediately.

Configuring HP ArcSight Task to populate host name or IP

The value of column application_host can be populated by adding a map with the value as arcsightAppNameHostMap as shown in the following example. The fieldThis is read from the map as explained below:

It is difficult to determine the host name or IP address of the account as the field is not constant in Application definition in IdentityIQ. Hence, customer can define a map in TaskDefinition and select the task added to export data in ArcSight table. The key in the map should be name of the application defined in IdentityIQ and value should be hostname, IP, or any string that ArcSight administrator understands.

To add the map:

1. Go to debug page, navigate to TaskDefinition and open the ArcSight task configured above.
2. Add the entry as key = Name of Application defined in IdentityIQ and value as the string to identify host of Account like Hostname or IP.
3. Save the task definition. For example:

```
  <entry key="arcsightAppNameHostMap">
    <value>
        <Map>
          <entry key="LinuxApp1" value="linux01.iiq.com"/>
          <entry key="LinuxApp2" value="127.15.19.21"/>
          <entry key="ADDirectApp" value="AD.iiq.com"/>
          <entry key="ServiceNowApp" value="https://iiq.service-now.com"/>
          <entry key="ACF2App" value="ACF2-Mainframe"/>
        </Map>
    </value>
</entry>
```

Note: If the application name is not defined in the map the host field is blank.

Following fields are added in export table:

Table 57—IdentityIQ sptr_arcsight_identity export table

Fields	Description
linkid	Primary key for Link table in IdentityIQ database. This field is copied from spt_link table id field and is the primary key for export table.
identityid	Primary key in Identity table. This field is copied from spt_Identity table.
modified_dt	Populates timestamp when the record is exported in export table. The field can be referred while configuring time based ArcSight database connector.
identity_display_name	Represents Display Name of Identity which is copied from spt_identity table field (display_name).
identity_firstname	Represents first name of Identity which is copied from spt_identity table field (firstname).
identity_lastname	Represents last name of Identity which is copied from spt_identity table field (lastname).
application_type	Populates the type of Account which is connected to the Identity like ActiveDirectory – Direct, ACF2 – Full, Box, Cloud Gateway, ServiceNow and so on.
application_host	The host name, IP, or any string which can be used by ArcSight administrator to identify the host of link/account uniquely. Customer can enter any string which can be sent to ArcSight to identify the host of link. This field can be populated as explained in “Configuring HP ArcSight Task to populate host name or IP” on page 305.
application_name	Populates the name of Application of the Account connected to the Identity.
link_display_name	The account connected to the identity which is copied from spt_link table, field display_name.
entitlements	Represents comma separated list of entitlements to the link of Identity.
risk_score	Represents the composite risk score of Identity.

Table 58—IdentityIQ sptr_arcsight_audit_event export table

Fields	Description
auditid	The audit ID which is primary key for the export Audit table. The field is copied from spt_audit_event table id field.
created_dt	Populates timestamp when the record is exported in export table. The field can be referred while configuring time based ArcSight database connector.
owner	Describes the Owner of the audit generated.
source	Provides more details to help ArcSight administrator determine the source of audit.
action	Describes the action taken on entity.

Table 58—IdentityIQ sptr_arcsight_audit_event export table

Fields	Description
target	Provides target details.
application	Describes the name of application the target belongs to.
account_name	The name of Account is populated in this field.
attribute_name	The name of attribute modified.
attribute_value	The value provided to the attribute.

Encrypted Data Synchronization Task

The Encrypted Data Synchronization Task is used to re-encrypt IdentityIQ data when a new custom encryption key is generated.

Table 59—Encrypted Data Synchronization Task Options

Option	Description
Disable Application Synchronization	Select this option to ignore encryption key synchronization against applications.
Disable Identity Synchronization	Select this option to ignore encryption key synchronization against identities.
Disable IntegrationConfig Synchronization	Select this option to ignore encryption key synchronization against IntegrationConfig objects.
Disable Attachment Synchronization	Select this option to ignore encryption key synchronization against attachments.
Convert Encrypted Identity Secrets to Hashing	Select this option to convert any encryption keys to use hashing.

Once you have completed customizing your report options, click Save for later use or Save and Execute to save the report and run it immediately

Entitlement Role Generator

The Entitlement Role Generator creates an Entitlement Role for every entitlement found in a specified application. Recommended role types are Entitlement or IT.

You can further refine creation by specifying an entitlement name or permission target so that only entitlements matching the specified criteria are used.

It is recommended to specify a template to be used to name the created roles. IdentityIQ uses Velocity templates. If no template is used, a generic name based on either the entitlement or role is created.

Table 60—Entitlement Role Generator Options

Option	Description
Applications	Select one or more applications from the drop-down list.
Type of Role to Create	Input the name of the role based on the specifications for your enterprise.
Enter the locale to check for descriptions. (If left blank the default Locale is used)	Enter the location of the role description.
Generate entitlements from attributes whose name starts with	Enter letters in the attribute name to filter the scan.
Generate entitlements from permissions whose target starts with	Enter letters in the permission name to filter the scan
Velocity template from which to generate entitlement role names. The template is always passed the applicationName parameter. The description, attributeName, attributeValue, permissionTarget, and/or permissionRights parameters are set when available.	Enter the Velocity template string.

FIM Application Creator

Run the FIM Application Creator task automatically discover and create FIM Management Agent Applications. This task auto-generates an application for each management agent defined in FIM. The applications generated by this task are added to the list of applications in IdentityIQ.

Table 61—ITIM Application Creator Options

Option	Description
Microsoft Forefront Identity Manager Applications	Select applications to inspect and from which applications should be generated based on the management agents found.
Native Object Types of Account	Specify the native object types of accounts created by this task.
Native Object Types of Group	Specify the native object types of groups created by this task.

IQService Public Key Exchange

Run the IQService Public Key task to change the public keys that are used for IQService communications.

Table 62—ITIM Application Creator Options

Option	Description
Select IQService based application(s)	Select IQService based applications on which to change the public keys.

ITIM Application Creator

Run the ITIM Application Creator task to inspect IBM Tivoli Identity Manager (ITIM) and retrieve information about the ITIM services (applications). This task auto-generates an application for each service defined in ITIM. Each ITIM application contains a list of services that are roughly equivalent to the list of applications maintained in IdentityIQ. The applications generated by this task are added to the list of applications in IdentityIQ.

Table 63—ITIM Application Creator Options

Option	Description
ITIM Applications	Select applications to inspect and from which applications should be generated based on the services found.
Generated application name prefix	Specify a prefix to append to any applications created by this task.
Generated application name suffix	Specify a suffix to append to any applications created by this task.

IdentityIQ Cloud Gateway Synchronization

IdentityIQ Cloud Gateway Synchronization tasks scan selected IdentityIQ applications for specified objects and synchronizes them with IdentityIQ Cloud applications. It is intended for use when IdentityIQ is not able to communicate directly with the managed system.

Table 64—IdentityIQ Cloud Gateway Synchronization

Option	Description
IdentityIQ Cloud Gateway application name	Select the name of the application to synchronize.
Applications hosted on the IdentityIQ Cloud Gateway	Select the name of the hosted cloud gateway application with which to synchronize the IdentityIQ application.
Rules to be executed on the IdentityIQ Cloud Gateway	Select which rules to execute against selected applications.

Identity Refresh

Refresh identity tasks scan all identities to ensure that all identity information is up-to-date and accurate. Refresh identity scans are also used to detect and report on policy violations and launch event certifications.

Incremental identity refresh can be configured to only refresh those identities on which information has changed since the last refresh was performed, to increase performance.

Note: Partitioning is disabled if you enable Mark dormant scopes after refresh or Refresh the group scorecards options.

Note: The Number of Refresh Threads option is not supported when partitioning is enabled.

Partitioning is available to speed the processing time for identity refresh tasks and level the load on the machines running these tasks. Partitioning is used to break operations into multiple pieces, or partitions. Each partition is then placed in a global queue, and machines, or hosts, in a cluster compete to execute the partitions in the queue. Machines are added or removed from the cluster dynamically with automatic balancing. If a machine fails or is taken down while processing a partition, the partition is placed back into the queue and reassigned to a different machine. A single result object is shared by all partitions and is continually updated so you can monitor the overall progress of the partitioned operation. When all partitions have finished executing the result is marked complete. See, “Partitioning” on page 281.

The information scanned and updated is determined by the following criteria when the task is created or edited. You can use any combination of options to build a task.

Table 65—Identity Refresh Options

Option	Description
Optional filter string to constrain the identities refreshed	A filtering string used to limit the number of identity cubes updated by this task. For example you can limit the refresh to one department within your enterprise, such as Finance, by entering: department == “Finance”
Optional list of group or population names to constrain the identities refreshed	A filtering string used to limit the number of identity cubes updated by this task. For example you can limit the refresh to one group or population within your enterprise.
Refresh identities whose last refresh date is before this date	Refresh any identities not refreshed since the date entered. Enter and date manually or click the **“...”** icon to display the calendar view. Use this to recover from a refresh that ended abnormally. For example, you start a refresh task and it runs for a day before stopping abnormally. After resolving the issue with the task, instead of repeating the refresh of all the identities that completed before the task stopped, you can only refresh the ones that were missed on the last refresh. Enter the approximate date the last refresh stopped and only refresh the remainder.
Refresh identities whose last refresh date is at least this number of hours ago	Enter the number of hours manually. Use this option to refresh identities that have not been refreshed recently. The time is in this option is relative rather than absolute. Instead of remembering a specific task launch date and typing that in each time you run the refresh task you can have just one task and run that repeatedly. For example you can run it for every thing more than an hour old.
Refresh identities whose last refresh date is within this number of hours	Enter the number of hours manually. Use this option to refresh identities that were refreshed recently. The primary use case for this is to refresh things that were recently touched by aggregation. For example, if you have several aggregation sources but those sources tend to touch different subsets of all identities, and you would rather not refresh the identities that were not touch be the last aggregation.

Table 65—Identity Refresh Options

Option	Description
Include modified identities in the refresh window	Refresh any identities modified within the specified time frames. There are two dates stored on each Identity, the date of last refresh and the date of last modification. The last refresh date is set whenever you run the refresh or aggregation tasks and the identity is changed in some way. The last modification date is set whenever you edit the identity in some way outside of a refresh or aggregation task, for example from a Lifecycle Manager workflow or a custom task. Use this option to refresh identities that were edited within a period of time, but not necessarily by the refresh task. For example, you might do a full refresh once a week but during the week people were adding or removing roles, changing extended identity attributes, doing manual correlation, or changing identities in some other way. Most of those cases have options to do a targeted refresh immediately after the change happens but this is not always the case and sometimes it is better to batch up a number of refreshes rather than have hundreds of individual refreshes occurring concurrently. If you ran the refresh task with one of the date-based options you would not necessarily pick up identities that were manually edited. If you want to include those select this option.
Refresh only identities marked as needing refresh during aggregation	Only refresh identities marked as needing refresh during the most recent aggregation task.
Do not reset the needing refresh marker after refresh	Do not clear the needing refresh marker set during aggregation. Use this option if you have multiple refresh tasks scheduled, such as entitlement and risk refresh. Then you can set the final refresh to clear the markers.
Exclude identities marked inactive	Exclude inactive identities from the refresh.
Refresh identity attributes	Update identity cubes with any changes made to the attributes used to define identities.
Refresh Identity Entitlements for all links	Refresh any account attribute mark as an entitlement in the application schema. This process is resource intensive as it refreshes all entitlement values for all links.
Refresh manager status	Update all identity cubes in which the manager status has changed. For example, if a user was promoted to manager in their department, their identity cube would be updated by this task.
Refresh assigned and detected roles and promote additional entitlements	Update any assigned or detected role assignments that have change since the last time this task was run. Any additional entitlements found in this refresh are promoted during this task.

Table 65—Identity Refresh Options

Option	Description
Provision assignments	Provision any assigned roles and entitlements detected since the last time this task was run.
Disable deprovisioning of deassigned roles	Prevents assigned roles from being deprovisioned after they have been deassigned.
Refresh role metadata for each identity	Update information about the identity's relationship to their role. For example, information regarding whether or not an identity has all the roles required by the given role. Note: This option must be selected in order to generate Role Statistics.
Enable manual account selection	Sent Account Selection Notification emails to users with more than one account on any application where the system cannot determine the provisioning account. By default, no provisioning is done in this case.
Synchronize Attributes	Provision identity mapping targets if their value has changed.
Refresh the identity risk scorecards	Update Identity Risk Scores with any information discovered by the scan performed by this task.
Maintain identity histories	Update the identity history by creating a snapshot of any identities with information that has changed since the last refresh.
Refresh the group scorecards	Update Group Risk Scores with any information discovered by the scan performed by this task. Partitioning is disabled if you select this option.
Clean up groups definitions that are no longer referenced	Delete un-referenced group definitions. This option is only supported if it is selected in conjunction with the Refresh the group score card option and they run in the same task.
Check active policies	Scan for active policies and apply those policies to the identities included in the task.
Keep previous violations	Maintain a history of violations that are no longer active.
A comma separated list of specific policy names. When set this overrides the default policies	Scan for and apply only those policies included in this list to the identities included in the task.
Refresh assigned scope	Refresh assigned scope based on changes discovered.
Disable auto creation of scopes	Do not automatically assign scope to identities as part of this task.

Table 65—Identity Refresh Options

Option	Description
Mark dormant scopes after refresh	Mark scopes that are not assigned to any identities as dormant. Partitioning is disabled if you select this option.
Process Events	Enable event certifications. Use the snapshots created during aggregation to approximate the previous state of the identities at the beginning of the refresh. This copied identity is compared to the updated identity to determine if event certifications are launched.
Refresh logical application links	Scan for changes to composite applications and refresh the link information.
Promote managed attributes	When enabled, any values for entitlement or permissions encountered while running the task automatically get promoted as managed attributes
Number of Refresh Threads	Specify the number of concurrent threads used during task processing. The number of threads should not exceed 10. This option is not supported whit partitioning enabled.
Always launch the workflow (even if the usual triggers don't apply)	Launch a workflow for each identity even if no identity triggers or provisioning policy questions apply.
Enable the generation of work items for unmanaged parts of the provisioning plan	Create work items for role entitlements that are not managed by available connectors or provisioning integration modules so the appropriate action can be taken.
Disable connector lookup of managers that do not correlate	Disable the default MANAGER_LOOKUP feature and stop the automatic lookup/bootstrap of the manager account at the connector level.
Enable partitioning	Enable partitioning of this task across multiple hosts. Partitioning has to be configured before this option is valid.

Identity Request Maintenance

The Identity Request Maintenance task scans all Lifecycle Manager access requests to ensure that all identity change requests were provisioned.

Table 66—Identity Request Maintenance Options

Option	Description
Max age (in days) for Identity Request objects	The maximum number of days that an identity request object (AccessRequest) is stored in the IdentityIQ database before it is removed. Set this according to your policy on how long access request details are required. The default is zero (0), which indicates that they are stored forever.
Verify provisioning for requests?	Scan for provisioning requests which have been verified.
Number of days to attempt to verify the request with the Identity model before failing.	The number of days the task attempts to scan for verified access requests before reporting a failure. When a timeout occurs, any item not verified is left in the non-finished provisioning state, either Committed or Pending, and the overall request is marked Partially Complete if any item succeeded. If no item succeed the entire request is marked failed. Set this value based on the type of connectors and their expected provisioning times. The default setting is continuous checking forever.

Missing Managed Entitlements Scan

Missing Managed Entitlement Scan tasks scan the selected application and create any entitlement objects for items added after the application was last aggregated.

Select the applications to include in the scan. At least one application must be specified. Click the arrow to the right of the suggestion field to display a list of all applications, or enter a few letters in the field to display a list of applications that start with that letter string.

This task returns a list of entitlement names, values, and the application on which they were detected.

Novell Application Creator

Run the Novell Application Creator task to inspect Novell IDM applications and retrieve information about the applications to which they are connected. This task generates an IdentityIQ application for applications connected to the Novell IDM application specified. The applications generated by this task are added to the list of applications in IdentityIQ.

Table 67—Novell Application Creator Options

Option	Description
Novell IDM Application	Select a Novell IDM application to inspect and from which applications should be generated.
Novell Connected Applications List	Specify the connected applications for which application should be created in IdentityIQ.

OIM Application Creator

Run the OIM Application Creator task to inspect Oracle Identity Manager applications and retrieve information about the applications to which they are connected. This task generates an IdentityIQ application for applications connected to the OIM application specified. The applications generated by this task are added to the list of applications in IdentityIQ.

Table 68—Novell Application Creator Options

Option	Description
OIM Application	Select an OIM application to inspect and from which applications should be generated.

Policy Scan

The Policy task type is used to run policies against identity cubes and update identity score cards with any policy violations discovered. IdentityIQ provides the Check Active Policies task as a global policy update task.

The information scanned and updated is determined by the following criteria when the task is created or edited. You can use any combination of options to build a task.

Table 69—Policy Scan Options

Option	Description
Optional filter string to constrain the identities refreshed	A filtering string used to limit the number of identity cubes updated by this task. For example you can limit the refresh to one department within your enterprise, such as Finance, by entering: department == “Finance”
Optional list of group or population names to constrain the identities refreshed	Use this list to further limit the number of identities included in this policy scan.
Apply all active policies	Scan for active policies and apply those policies to the identities included in the task.
A comma separated list of specific policy names. When set, this overrides the default policies	Scan for and apply only those policies included in this list to the identities included in the task.

Propagate Role Changes

The Propagate Role Changes task updates any identities that have assigned roles whose associated entitlements have changed. This is the only task that can propagate the removal of entitlements from an assigned role.

Table 70—Propagate Role Changes Options

Option	Description
Number of minutes task should run	The number of minutes for the task to run. The task stops only after finishing current event processing.
Check active policies	Scan for active policies and apply those to the identities included in the task.
Keep previous violations	Mark old policies as inactive but do not delete them.
A comma separated list of specific policy names. When set, this overrides the default policies.	Scan for and apply only those policies included in this list to the identities included in this task.
Enable Partition	Allow the task to split into partitions and run across multiple threads and hosts, if available.

- Once you have completed customizing your report options, click Save for later use or Save and Execute to save the report and run it immediately.
- After executing the task, the Task Result page displays the following output:
- **Number of Identity Updates**: displays the total number of Identity updates propagated. It is different than number of Identities updated, since multiple role events include some common identities and are counted multiple times, for each role event.
- **Number of Events Processed**: displays the total number of role events propagated. This is not the number of role modifications but the role change events in the queue. As single role modification results in multiple role change events in the queue.
- **Number of Events Pending:** displays the total number of pending role change events in the queue. If timeout is not defined, Role Propagation task completes only after propagating all the events. If timeout is defined, there could be pending events in the queue even after successful completion of this task.
- **Number of Events with No Impacted Identities:** displays the total number of events which are not impacting on connected identities. This event count is based on those bundles which are not directly assigned to the identities.
-
- Role change events are propagated sequentially and are not consolidated to cancel out redundant changes.
- If Refresh Identity Task is run before Role propagation task, and if it adds any entitlement as part of role changes, processing of role change event through Role Propagation Task would be redundant.
- In case of retry status, the transaction would be marked as failed and role propagation task would be stopped.
- While adding an entitlement, if account is missing, transaction would be marked as failed and role propagation task would be stopped. User has to run Refresh Identity task to resolve this.
- While processing an event, if the following exception is from target system, the task would remain blocked until the events are successfully processed.
- `mandatory group cannot be removed`

- This issue can be resolved by deleting the event from the database.
  - When Role Propagation task is under execution, if user creates events in database, these events would not be considered by the current task. These events would be considered for the next task execution.

Refresh Logical Accounts

The Refresh Logical Accounts task type is used to refresh composite accounts for all identities that could, potentially, have a logical account on the applications selected. This refresh occurs without performing aggregation on the logical or tiered applications containing the links.

A logical account rule is run on each identity that has a logical link, or a link on the primary tier application of the logical application. If no primary tier has been defined, the rule is run on all identities that have an account on any of the tier applications.

The information scanned and updated is determined by the following criteria when the task is created or edited. You can use any combination of options to build a task.

Table 71—Refresh Logical Accounts Options

Option	Description
Logical Applications	Select composite applications to refresh from the drop down-list.
Refresh identities whose last refresh date is before this date	Refresh any identities that have not been refreshed since the date entered. Enter and date manually or click the **"..."** icon to display the calendar view.
Refresh all application account attributes	Perform an aggregation of identity information on each application and update the account attributes on each identity as required. Selecting this attribute initiates a full application aggregation for each identity included in this task. This might impact the performance of IdentityIQ.
Refresh identity attributes	Update identity cubes with any changes made to the attributes used to define identities.
Refresh manager status	Update all identity cubes in which the manager status has changed. For example, if a user was promoted to manager in their department, their identity cube would be updated by this task.
Refresh the identity risk scorecards	Update Identity Risk Scores with any information discovered by the scan performed by this task.
Maintain identity histories	Update the identity history by creating a snapshot of any identities with information that has changed since the last refresh.
Refresh the group scorecards	Update Group Risk Scores with any information discovered by the scan performed by this task.
Apply all active policies	Scan for active policies and apply those policies to the identities included in the task.

Table 71—Refresh Logical Accounts Options

Option	Description
A comma separated list of specific policy names. When set this overrides the default policies	Scan for and apply only those policies included in this list to the identities included in the task.
Number of Refresh Threads	Number of threads to use simultaneously while running this task.

Role Index Refresh

A role index refresh task updates all role information and creates the indexes needed to perform role searches. You must run this task before performing any role searching.

Run Rule

A task used to run an arbitrary rule with a series of name/value pairs.

You must have to configure some return statement as string. From your code, you have to return some meaningful string to the task. In your task definition declare:

```
<Returns>
 <Argument name="tskSuccess" type="string">
   <Prompt>Task  Result:</Prompt>
 </Argument>
</Returns>
```

And in your code:

```
String tskSuccess = "failed";
if ( Do some condition check here) {
//Do something;
String tskSuccess = "Success";
}
return tskSuccess;
```

The rule is expected to return a string value representing its status. Any string other than `Success` results in a failed task result.

Sequential Task Launcher

A sequential task launcher initiates the specified tasks in the order defined. This enables you to run tasks sequentially without having to schedule each separately based on estimated run times.

Table 72—Sequential Task Launcher Options

Option	Description
Enter the list of tasks you would like to run. Tasks are run in the order that they are entered.	Select the tasks you would like to run and the order in which they should run.
Print log statements to indicate which tasks have been completed.	Select to print log statements so the sequential tasks can be tracked.
Cease execution if one of the executing tasks encounters an error.	Select to stop the sequential task if one of the tasks in the list fails. If this option is not selected the task continues in order.

System Maintenance

SailPoint provides System Maintenance tasks with the IdentityIQ application, the Work Item Expiration Scanner, Mitigation Expiration Scanner, System Maintenance, System Maintenance Object Pruner, Role Overlap Analysis, and the Synchronize Roles task. These tasks are configured, by default, to run in the background of the application and update score card, application, and role information as needed.

The Work Item Expiration Scanner checks for work items that were assigned but have not been completed by the set expiration date.

The Mitigation Expiration Scanner checks for roles or entitlements for which the exceptions allowed during certification have expired.

The Perform Maintenance task prunes identity snapshots, task results, access request attachments, and certifications, escalates orphaned work items, and performs other background maintenance tasks.

The System Maintenance Object Pruner prunes objects in batches to improve performance. This task is not part of the System Maintenance task pruning operations and is run independently when necessary. This task is always run with partitioning enabled.

The Role Overlap Analysis performs impact analysis on a specified role. The task result name is annotated with the name of the selected role so you can tell multiple analysis results apart.

The Synchronize Roles task synchronizes IdentityIQ roles with the roles on the identity management systems that are configured to work through a provisioning provider.

Target Aggregation

A target aggregation task scans applications and aggregates activity targets from those applications. These targets are then used for activity monitoring and risk assessment.

Select the applications to include in the scan. If no applications are specified, all applications are included. Click the arrow to the right of the suggestion field to display a list of all target sources.

How to Complete Task Work Items

Task work items are generated by task that require sign off on the results they create, and those sign off request that are forwarded by a designated signer. Sign off request are displayed on your Home Page and you are notified through an email when the work item is created.

Sign off decisions are retained with the task results for tracking purposes.

1. Click on a sign off type work item to display the sign off request.
2. Review the work item information in the Summary section.
3. Review the Comments section for any information associated with this work item.
   Refresh
4. Click **Click to View Task Results** in the Details sections to display the Task Results page.
5. Click **Return to Work Item** when you have completed your review of the task results.
6. Click on an action button to open the associated comments dialog and conclude this work session.

   If you sign off on, or reject the sign off request, the status of the task results is updated to reflect that decision. You must specify a recipient If you forward the work item.

Chapter 20: Role Management

This chapter contains the following sections:

- Role Modeling — used to create and maintain the roles that define your enterprise. See “Role Modeling” on page 323.
- Multiple Role and Account Assignment — roles can be assigned to the same identity multiple times, and roles can be applied to multiple accounts on the same application. See “Multiple Role and Account Assignment” on page 352.
- Automated Propagation of Role Changes to Role Members — any changes to the role or delete a role, that change would propagate to all identities that are currently assigned to the said role. See “Automated Propagation of Role Changes to Role Members” on page 355.

Role Modeling

Role modeling is used to create and maintain the roles that define your enterprise. These roles are used to categorize and manage users based on job function. Roles also provide a translation between business and IT functions, ease the provisioning and the request process for new access, simplify auditing, and the access review and certification process.

Roles are an important part of any identity control system. Roles enable business managers to make more accurate decisions and to make an appropriate trade-off between business benefits and risks. Roles make it easier to translate business process rules into technical IT controls. Roles enable better visibility into IT data so that results and metrics can be understood and approved by business managers and executives.

Role mining enables you to create new roles within IdentityIQ by analyzing data within the system using pattern-matching algorithms. IdentityIQ supports role mining to create both business and IT roles. Business roles typically model how users are grouped by business function, including functional hierarchies, project teams, or geographic location. IT roles typically model how application entitlements (or permissions) are logically grouped for streamlined access.

Business role mining within IdentityIQ facilitates the creation of organizational groupings based on identity attributes – for example department, cost center or job title. The business role mining supports multiple configuration options to assist users in generating new roles. After the mining task is completed, the new roles are added to the Role Viewer where they can be modified as necessary.

IdentityIQ also supports the creation of roles based on the mining of entitlements within the enterprise. These roles typically model the IT privileges required to perform a specific function within an application or other target system. Using a configurable algorithm, IdentityIQ searches for access patterns to determine logical groupings of entitlements.

When you define roles based on entitlements from the applications being monitored by IdentityIQ, the aggregation and correlation process discovers the entitlements, matches them to the roles you defined, and assigns those roles to the users who have those entitlements. If you create a hierarchical structure of roles using the inheritance function of the Role Viewer, users are assigned the lowest level role discovered during aggregation. For example, if role A is a member of role B, and role B is a member of role C, and an identity is discovered that is assigned all of the entitlements that defined roles C, B, and A, they are assign role A. Assigning the lowest level role enables operations such as certifications to be performed on one role instead of on each entitlement assigned to the user.

Role type is used to configure roles to perform different functions within your business model. For example, type might be used to control inheritance or automatic assignment of roles. Role types are configured on the System Setup page.

Role management also uses the concept of permissions to enable you to grant users permission to certain roles without assigning them the role or incorporating it in their role hierarchy. For example, while a non-IT user with a business-type role might need access to the entitlements contained within an IT-type role, they probably do not need to have that role assigned to them or included as part of their hierarchal role structure.

Role archiving enables you to store versions of roles that have changed over time. This function enables you to roll-back to previous versions of the role if necessary. If roll approval is required in your enterprise, role roll-backs also require approval. Role archiving is controlled through business processes and is enabled during the configuration of the IdentityIQ product.

Role activation events enable you to use business processes to automatically activate or deactivate roles based on dates specified in the role modeler. Role activation business processes can be configured to automatically refresh identities to include or exclude the impacted roles.

Granted IdentityIQ user rights enables you to associate specific IdentityIQ capabilities and scopes to roles. Those capabilities and scopes are then granted to identities when they are assigned the role and the Identity Cube Refresh task is run with the **Provision assigned roles option** selected. By default this function is disabled in IdentityIQ and must be turned on during the deployment and configuration process.

The Role Management page includes the following:

- "Role Search Tab" on page 333
- "Entitlement Analysis" on page 336
- "IT Role Mining" on page 338
- "Business Role Mining" on page 340
- "Working with the Role Manager" on page 345

Role Viewer Tab

Note: The Role Navigation panel can display roles that are outside of your assigned scope. You cannot edit those roles.

The Role Navigation panel of the Role Viewer tab displays your existing roles. The list of roles can be organized in a top down, bottom up, or grid format. The grid shows a simple list of roles in alphabetic order. If you expand a role in the Top Down view you see the roles that are members of the expanded role. If you expand a role in the Bottom Up view you see the roles in which the expanded role is a member. Use filtering to locate specific roles in the Top Down and Bottom Up views.

Click the arrow icon on the top, right side to contract or expand the Role Navigation panel. Contracting the panel provides more screen space to view role details in the Role Information panel.

Click a role to display detailed information in the Role Information panel of the Role Viewer.

If approval and impact analysis are active, roles and profiles that have changes pending approval or are undergoing impact analysis are displayed with a red square surrounding their icon. Role analysis and role approval are an important part of the overall role life-cycle management. Role analytics and approval for new, modified, or rolled-back roles are controlled thought business processes configured for your implementation of IdentityIQ.

Inactive roles that are not pending approval or analysis are displayed with a gray icon.

Click **Add** to display the Role Editor page and define a new role. Right-click an existing role and select **Clone** to create a new role based on the existing one. See “Role Editor Page” on page 327

To delete a role, right-click the role and select **Delete**. You must then confirm the deletion request.

The Role Information panel contains all of the information associated with the selected role. Some of the sections listed in the table might not be available for all role types. If there is information associated with a role that is not supported by the assigned role type, the information is displayed with a warning message.

Roles in which activation rules are enabled display a notice in the upper right-hand corner of the information panel containing activation or deactivation information.

Table 73—Role Viewer - Information Panel Descriptions

Field Name	Field Description
Name	The name of the role.
Display Name	The name to be used throughout IdentityIQ.
Owner	The owner assigned to the role.
Scope	The scope of this role. Scope is used to determine the objects to which a user has access. If scoping is active, identities can only see objects that they created or that are within the scopes they control. **Note: The scope option is only displayed if the scope feature is enabled.**
Type	The type of role being displayed. Role type definitions are customizable and created as part of the configuration process.
Description	A short description of the role.
Extended Attributes	Any extended role attributes configured for your enterprise and marked as searchable are displayed with the role information. For example, Identity Attribute, Date Attribute, Rule Attribute.

Table 73—Role Viewer - Information Panel Descriptions

Field Name	Field Description
Role Statistics	The Role Statistics panel displays detailed statistical information on the users and entitlements a given role. Click each applicable category to view a window containing item-specific statistical information. Available IdentityIQ categories include the following: Members - Number of Identities assigned the role. Click to view a grid displaying those identities. Members with Additional Entitlements - Number of Identities that have entitlements which are not permitted or required by this role or any other role they have been assigned. This applies to Business Roles provided by IdentityIQ, not to custom roles. Members with Missing Required Roles - Number of Identities that are missing roles which are required by this one. This applies to Business Roles provided by IdentityIQ, not to custom roles. Identities Detected - Number of Identities whose entitlements indicate that they have this role. Click to view a grid displaying those identities. This applies to IT and Entitlement Roles provided by IdentityIQ, not to custom roles. Identities Detected to be Exceptions - Number of Identities whose entitlements indicate that they have this role, even though they have not been assigned any roles that permit or require this one. Click to view a grid displaying those identities. This applies to IT and Entitlement Roles provided by IdentityIQ, not to custom roles. Provisioned Entitlements - Number of Entitlements that would be provisioned if this role were to be assigned to and/or required by a new Identity. This applies to Business, IT, and Entitlement Roles provided by IdentityIQ, not to custom roles. Permitted Entitlements - Number of Entitlements that would be provisioned in order for an Identity to match all roles permitted by this one. This applies to Business Roles provided by IdentityIQ, not to custom roles. Click the Refresh button at the bottom of the panel of each role you wish to view the statistics. -OR- Run the Refresh Role Scorecard task to populate and display the statistical data by default on all roles. Note: The “Refresh role metadata” option must be selected in the Refresh Identity Cubes task in order for Role Statistics panel to display any information.
Scheduled Events	The events scheduled for this role. Activate — the date on which the role becomes active. Deactivate — the date on which the date is to be deactivated.
Archived roles	Previous, or different, versions of this role. If archiving is active, each time a change is made to a role definition a version of the role is stored. This enables you to roll-back to previous versions if required.

Table 73—Role Viewer - Information Panel Descriptions

Field Name	Field Description
Assignment Rule	The rule used to automatically assign roles to identities during a correlation process. Roles assigned either manually on the identities pages or through an assignment rule are considered Assigned Roles throughout IdentityIQ.
Inherited Roles	The roles in which this role is a member.
Permitted Roles	Roles to which users have access if they are assigned this role.
Required Roles	The roles to which the user must have access if they are to be assigned this role.
Entitlements	The rules and permissions (targets and rights) that define the profiles contained within the role. The entitlements are grouped by application.
Inherited Entitlements	The entitlement details for the entitlements that define the roles to which this role is a member. The included entitlements are grouped by application.
Granted IdentityIQ User Rights	The IdentityIQ capabilities and scopes associated with role. These rights are granted to the identities to whom this role is assigned. **Note: These capabilities and scopes are not assigned until a Identity Cube Refresh task is run with the Provision assigned roles option selected.**

The Role Viewer tab enables you to work with the following IdentityIQ components:

- Roles — See “Role Editor Page” on page 327
- Archived Roles — See "Role Editor - Archived Role Panel" on page 330
- Profiles — See "Role Editor - Edit Entitlement Panel" on page 330

Role Editor Page

Use the Role Editor to define the roles that make up your enterprise. A role is a collection of roles or profiles that enable an identity to perform certain operations. For example, one role might enable an identity to request a purchase order and another might enable an identity to approve purchase requests. Use roles to monitor identity entitlements, identify policy violations, and compile identity risk scores to enable you to maintain compliance.

Note: When adding new roles, the list of attributes changes to reflect the currently selected role type. When editing a role, if the role type changes, any attributes from the original role are preserved and the user is prompted with the warning message “This attribute does not apply to the current role type”.

Click **Submit** to save the changes made on this page. Click **Submit with Impact Analysis** to save the changes and create an impact analysis report. the report provides details on the impact these changes have on the rest of your product implementation and statistics on the amount of overlap between the new role and existing roles. If the approval business process is active, an approval work item is sent to the role owner and the role changes are inactive until the approval is complete. See “How to Approve Role Changes” on page 351 and "How to Perform Impact Analysis" on page 351.

Roles that are awaiting approval are displayed with a red square around the role icon. You can further edit roles with approval or analysis pending, but a notice displays at the top of the page alerting you that “An approval or impact analysis work item is pending on this role.” If you change and submit a role with changes pending, the original work item is deleted and replaced with a work item containing the latest changes. A role with changes pending approval displays the original, unchanged, role information on the Role Information panel, but the latest,

changed, information on the Role Editor page. This enables you to view the role as it currently exists in the Role Information panel, but ensures that you do not duplicate changes on the Role Edit page.

See “How to Create or Edit a Role From the Role Management Page” on page 345 for information on how to work with roles.

The Role Editor panel contains all of the information associated with the selected role. Some of the sections listed in the table might not be available for all role types. If there is information associated with a role that is not supported by the assigned role type, the information is displayed with a warning message.

Table 74—Role Management - Role Editor Field Descriptions

Field Name	Description
Name	The name of the role.
Display Name	The name to be used throughout IdentityIQ.
Type	The type of role. For example, organizational, business, or IT. Role type definitions are customizable and created as part of the configuration process.
Owner	Enter a valid user or workgroup. Typing the first few letters of a name displays a list of all of the user and workgroup names in the system containing that letter combination. You can select from the displayed list.
Scope	Select a scope from the drop-down list. Only scopes that you control are displayed in the list. Scope is used to determine the objects to which a user has access. If scoping is active, identities can only see objects that they created or that are within the scopes they control.
Description	A brief description of the role. **Note: This description is displayed with the role throughout IdentityIQ and should be as intuitive as possible.** **Note: You must Save the description before changing languages to enter another description.** Use the language selector to enter description in multiple languages. The drop-down list displays any languages supported by your instance of IdentityIQ. The description displayed throughout the product is dependent on the language associated with the user’s browser. If only one description is entered, that is the description used by default.
Enable Activity Monitoring	Activate this feature to track activity for any user who is assigned this role. If activity monitoring is not available on the selected application, the Activity Monitoring Enabled check-box is replaced by the following note: This application does not currently have activity monitoring configured.
Provision both profiles and policies	Provision any changes to either profiles or policies associated with this role.

Table 74—Role Management - Role Editor Field Descriptions

Field Name	Description
Allow multiple application accounts	Enables a role to specify its own target account, or create a new account, during a role request, even if it is required by another role and included in that roles required roles list. If this option is not enabled, required roles are assigned to the same account as the top-level role.
Enable multiple assignments	**Note: This option is not available if either multiple assignments are not enabled, or if they are universally enabled.** Enables a role to be assigned to the same identity multiple times. This option is only available on assignable role types.
Disable	Disable the role so that it is no longer available in your application. Disabled roles names appear gray in the Role Navigation panel.
Custom or Extended Role Attributes	Any extended role attributes configured for your enterprise and marked as searchable are displayed with the role information.
Scheduled Events	The activation events scheduled for the role. Activation events use business processes to automatically activate or deactivate roles based on the dates specified in the Add New Event dialog.
Assignment Rule	A rule used to automatically assign roles to identities during a correlation process. Assignment rules can be created using: **Match List** — only identities whose criteria match that specified in the list. The criteria is configured using the tools provided. Add identity attributes, application attributes and application permissions. Customize further by creating attribute groups to which this assignment rule applies. Note: If Is Null is selected, the associated value text box is disabled. When the is null match is processed, the term matches users on the chosen application who have a null value for that attribute/permission. **Filter** — a custom database query for role creation. **Script** — a custom script for role creation. **Rule** — select an existing rule from the drop-down list. Note: Click the "..." icon to launch the Rule Editor to make changes to your rules if needed. **Population** — select an existing population and assign this role to identities in that population.
Permitted Roles	Roles to which users have access if they are assigned this role.
Required Roles	The roles to which an identity must have access before this role can operate properly.
Inherited Roles	The roles in which this role is a member.
Entitlements	Detailed information about the entitlements that are contained in the role. Use this panel to create new entitlements or edit or delete existing entitlements. Mouse over the information icon to display the description of an entitlement.

Table 74—Role Management - Role Editor Field Descriptions

Field Name	Description
Provisioning Policy	A list of provisioning policies associated with this role. Use this panel to add, edit, or delete provisioning policies.
Granted IdentityIQ User Rights	Use this panel to specify the IdentityIQ capabilities and scopes associated with role. These rights are granted to the identities to whom this role is assigned. **Note: These capabilities and scopes are not assigned until a Identity Cube Refresh task is run with the Provision assigned roles option selected.**

Role Editor - Archived Role Panel

Click an archived role to display the Archived Role panel and view the details of the archived role and determine the proper version for this roll-back.

Click **Roll Back to Archive Role** to return to the Role Editor page. Use the action buttons on the bottom of the page to complete the procedure. If approval is required on role changes it is required when a role is rolled back to a previous version.

Role Editor - Edit Entitlement Panel

Use the Edit Entitlement panel to define the profiles that are included in the role. A profile is a set of entitlements on an application. An entitlement is either a specific value for an account attribute, most commonly group membership, or a permission. Profiles are not shared between roles.

Click **Submit** to save changes or add the profile to the role.

Note: The simple view might not be available for all roles.

There are two options for adding entitlements to a role, the **Simple View** or the **Advanced View**. The simple view eliminates the need to create attribute rules to locate entitlements and provides a drop-down list of the entitlement configured for selection for each application. See “How to Create or Edit a Profile” on page 347 for information on how to work with profiles.

The Entitlement Editor panel contains the following information:

Table 75—Role Editor - Edit Entitlement Panel Simple View Field Descriptions

Field Name	Description
Application	The application associated with the account attributes or permissions for this profile.
Account Attribute	The value of the account attribute, most commonly group membership.
Select Entitlement	Specify as many entitlements as required for this role.

Table 76—Role Editor - Edit Entitlement Panel Advanced View Field Descriptions

Field Name	Description
Description	A brief description of the profile. **Note: This description is displayed with the role throughout the product and should be as intuitive as possible.**

Table 76—Role Editor - Edit Entitlement Panel Advanced View Field Descriptions

Field Name	Description
Application	The application associated with the account attributes or permissions for this profile.
Attribute Rules: Attribute rules are made up of filters that can be grouped and controlled using AND\OR operations. The attribute rules associated with a profile can be as simple or complex as needed. The Add a Filter box is used to create the individual filters, the Filter(s) box is used to view and manipulate the existing filters. See "How to Create or Edit a Profile" on page 347.	
Field	The attribute associated with the attribute filter. The drop-down list contains all attributes configured for the selected application. Applications are configured on the Configure Application page. See "Configure Applications" on page 95.
Search Type	The qualifier associated with the attribute value. **Multi Valued attributes** — contains all, is null, is not null **Long, Int, Date** — All except contains all and is like — equals, is less than, is greater than, is greater than or equal to, is less than or equal to, is in, is null, is not null, is not equal **Boolean** — equal, is not equal to, is null, is not null **Permission** — equals, is not equal, is in, is null, is not null **Everything else** — All operations except contains all — is like, equals, is less than, is greater than, is greater than or equal to, is less than or equal to, is in, is null, is not null, is not equal
Value	**Note: This field is not available for unary operations.** The value of the attribute. When available, select an entitlement from the drop-down list.
Ignore Case	Specifies if case should be a factor when comparing entitlements defined for profiles with those assigned to users. During identity correlation, the entitlements defined in profiles are compared with entitlements assigned to users to determine roles and additional entitlements for certifications. This field is not available for unary operations.
Operation	The operation used to control the interaction between the filters.
Permissions:	
Rights	The rights associated with this profile on the target attribute. For example, create, read, update, delete, execute. Use the Shift and Ctrl keys to select multiple rights from the list.
Target	The target attribute for this permission.

Role Editor - Provisioning Policy Editor Panel

Provisioning policies define the fields required for a role to be provisioned, often including a default value or script/rule for calculating a value. With a provisioning policy in place, when a role is requested and a field cannot be calculated by the system, the user must input specified criteria into a generated form before the request can be completed.

See "How to Create or Edit a Provisioning Policy" on page 333 for information on how to work with provisioning policies.

The Provisioning Policy Editor panel contains the following information:

Table 77—Role Editor - Provisioning Policy Editor Field Descriptions

Field Name	Description
Edit Provisioning Policy Fields Panel Use the Edit Provisioning Policy Fields panel to customize the look and function of the form fields generated from the provisioning policy.	
Name	The name of the field.
Display Name	The name displayed for the field in the form generated by the provisioning policy.
Help Text	The text you wish to appear when hovering the mouse over the help icon.
Type	Select the type of field from the drop-down list. Choose from the following: Boolean — true or false values field Date — calendar date field Integer — only numerical values field Long — similar to integer but is used for large numerical values Identity — specific identity in IdentityIQ field Secret — hidden text field String — text field
Multi Valued	Choose this to have more than one selectable value in this field of the generated form. Click the plus sign to add another value.
Read Only	Determine how the read only value is derived: Value — value based on the selection from the drop-down list Rule — value is based on a specified rule Script — value is determined by the execution of a script
Hidden	Determine how the hidden value is derived: Value — value based on the selection from the drop-down list Rule — value is based on a specified rule Script — value is determined by the execution of a script
Owner	The owner of the provisioning policy. This is determined by selecting from the following: None — no owner is assigned to this provisioning policy. Application Owner — identity assigned as owner of the application in which the provisioning policy resides. Role Owner — identity assigned as owner of the role in which the provisioning policy resides. Rule — use a rule to determine the owner of this provisioning policy. Script — use a script to determine the owner of this provisioning policy
Required	Choose whether or not to have the completion of this field a requirement for submitting the form.
Review Required	Choose whether or not to require the person who is approving the workflow item to approve this field.
Refresh Form on Change	Select this option to have the form associated with this policy refresh to reflex changes to this policy.

Table 77—Role Editor - Provisioning Policy Editor Field Descriptions

Field Name	Description
Display Only	Set this field as display only.
Authoritative	Boolean that specifies whether the field value should completely replace the current value rather than be merged with it; applicable only for multi-valued attributes
Value	Determine how the value is derived. Select from the following: Literal — value is based on the information you provide Rule — value is based on a specified rule Script — value is determined by the execution of a script
Value	The value displayed in the field of the generated form before editing. Choose from the following: None — the field is blank Literal — value is based on the information you provide Rule — value is based on a specified rule Script — value is determined by the execution of a script
Validation	Gives the ability to specify a script or rule for validating the user's value. For example, a script that validates that a password is 8 characters or longer.

How to Create or Edit a Provisioning Policy

To Create or Edit a Provisioning Policy:

1. Access the Provisioning Policy panel from the Role Editor page.
2. Click an existing provisioning policy to edit or click Add Provisioning Policy to create a new one.
3. Edit the provisioning policy information.
4. Optional: Add or delete provisioning policy fields.
   See "Role Management - Role Editor Field Descriptions" on page 328 for descriptions of the fields in each section.
5. Select fields to include in the form.
6. Click **Save** to return to the Role Editor.

Role Search Tab

Use the Role Search tab to generate searches on the roles. These searches can be used to locate roles by name, owner, type, or status. You can also search for roles by the number of users to whom they are assigned, either manually or through role assignment rules, the number of entitlements they contain, their risk score weight, their association to other roles, the last time they were assigned or certified, or any combination of that criteria.

For example, you can identify roles that were created but are not being used by searching for setting **Detected Total** and **Assigned Total** to less than one (1).

Note: The Refresh Role Indexes task must have run at least once before a roles search can yield results.

Searches yielding helpful results can be saved for your reuse, or saved as reports. Saving a search as a report enables scheduling of the search on an on-going basis for monitoring and tracking purposes. See “Reporting” on page 407.

The search fields are inclusive, only actions matching values specified in all fields are returned with the results.

Search criteria is used to narrow the result set for a search. Not entering information or making a selection in a search criteria field implies that all possible choices should be included. For example, if you do not enter an type in the **Type** field, events with any action type are included.

The Role Search tab contains the following information:

Table 78—Role Management - Role Search Criteria

Criteria	Description
Saved Searches:	
Search Name	The names of past searches that you have saved for reuse. **Note: These Saved Searches are only available for your use.**
Loaded Saved Search:	
The name and description of the saved query with which you are working.	
Run Search	Run the search with the criteria displayed on the current page. **Note: If you have modified the criteria of the Loaded Saved Search, the modified criteria is used for the search.**
Clear Search	Unload the Loaded Saved Search and clear all query options.
Delete Search	Delete the specified Loaded Saved Query.
Role Attributes:	
Name	Enter a role name on which to search. Entering a string of characters returns all roles with that string in their name that your controlled scopes enable you to view. For example, if you enter `admin` the search returns information for the roles System Administrator, SysAdmin, and Administrative Assistant.
Owner	Enter the role owner on which to search. Click the arrow to the right of the suggestion field to display a list of all role owners, or enter a few letters in the field to display a list of role owners whose names start with that letter string.
Type	Select the role type on which to search. For example, IT, Organizational, or Business. Role types are defined for your enterprise during the role modeling process.
Status	Select the status of the roles to include in the search, Enabled or Disabled.
Detected Total	Specify an upper or lower limit to the number of identities by whom this role can be detected and still be included in the search results. Detected roles are roles that are automatically assigned to identities based on the entitlements to which they have access. For example, to search for roles that were not detected by any identity during correlation, select Less Than from the drop-down list and type `1` in the empty field. This search returns all roles that were not automatically assigned to at least one identity.

Table 78—Role Management - Role Search Criteria

Criteria	Description
Assigned Total	Specify an upper or lower limit for the number of identities to whom this role can be assigned and still be included in the search results. Assigned roles are roles that were manually assigned to an identity by someone with role assignment authority or through a role assignment rule. For example, to search for roles that were not assigned to any identity, select Less Than from the drop-down list and type `1` in the empty field. This search returns all roles that were not manually assigned to at least one identity.
Entitlement Total	Specify an upper or low limit to the number of entitlements a role can contain and still be included in the search results. For example, if you select Less Than and type `3`, the search returns roles that contain two (2), one (1), or zero (0) entitlements.
Risk Score Weight	Specify an upper or lower limit for risk score weight that can be assigned to a role for it to be included in the search results. For example, you can specify a Greater Than value to search for high-risk roles, or you can specify a Less Than value to search for roles that were created with a risk score weight that is too low for their type. In the second example, if your enterprise has a policy that requires that all IT-type roles have a risk score weight of 100, you can select IT from the **Type** drop-down list, select Less Than from the **Risk Score Weight** drop-down list, and type 100 in the empty field to return all IT-type roles with a risk score weight less than 100.
Associated To Another Role	Include only roles that are associated with at least one other role or only roles that are not associated with any other role. **True** — include only roles that are associated with at least one other role. **False** — include only roles that are not associated with any other roles.
Filter By: Date	
Date Type	Select a state with which to relate the dates specified: Last Membership Certification — the date the last role membership certification was performed. Last Composition Certification — the date the last role composition certification was performed. Last Assigned — the date the role was last assigned to an identity.
Start Date	Specify a beginning date for this search. The search returns information pertaining to any action performed on or after the date specified.
End Date	Specify an end date for this search. The search returns information pertaining to any action performed on or before the date specified.
Fields to Display:	

Table 78—Role Management - Role Search Criteria

Criteria	Description
Fields to Display	Specify the information displayed on the Role Search Results page associated with this search. Each field defines a column on the results table. **Note: You must select at least one field to display on the results page.**

Role Search Results

The Role Search Results panel displays all of the roles that match the criteria specified in your search. The results are dependent on the **Fields to Display** list on the Role Search tab. From the Role Search Results panel you can export your search results to file and save the search criteria for use in the future.

Searches Options:

Use the drop-down list above the Role Search Results panel to save search criteria for use in future searches:

- **Save Search** — save the search for your own use. A list of saved searches is displayed at the top of the search tab each time you log in.
- **Save Search as Report** — searches saved as reports are added to your list of reports and can be scheduled to run on an on-going basis See “Reporting” on page 407.

Export Searches:

Use the buttons on the top, right of the Role Search Results dialog to export the search results to file for archiving and auditing purposes. The search results can be exported to an Adobe PDF or Microsoft Excel format.

Entitlement Analysis

IdentityIQ supports the creation of roles based on the mining of entitlements within the enterprise. These roles typically model the IT privileges required to perform a specific function within an application or other target system. Using a configurable algorithm, IdentityIQ searches for access patterns to determine logical groupings of entitlements.

Entitlement analysis enables you to search for entitlements based on specific application and identity information or by populations defined within your deployment of IdentityIQ. This feature enables you to create meaningful roles without having to remember every entitlement on every application or be familiar with the access assigned to each employee in your enterprise.

Entitlement Analysis also enables you to analyze the entitlement information collected to further refine the roles you are creating before saving.

Performing Entitlement Analysis involves three distinct phases:

- Searching for entitlements
- Analyze the search results
- Creating roles

Search for Entitlements:

1. Access the Entitlement Analysis tab from the Role Management page.
2. Select the applications on which to search for entitlements.
   Enter the fist letters of an application name to display a suggestion list, or click the arrow to the right of the field to display a list of all the applications to which you have access.

3. *Optional:* Narrow your entitlement search using the Identity Attribute fields or a list of populations. Use the **Search by Attribute** or **Search by Populations** radio buttons to switch between the options. The Identity Attribute fields displayed are dependent on the identity attributes defined during configuration. Populations are defined from the Advanced Analytics, Identity Search Results page.
4. Click **Search** to begin the entitlement mining based on the specified criteria.

Analyze the Search Results:

The search returns the following information:

Note: The search only returns those entitlements based on account or group attributes, not those based on permissions.

Table 79—Role Management IT - Role Analysis Search Results Descriptions

Column	Description
Search Parameters:	
Attribute	The criteria used to define this search. For example, Application, Last Name, Population, or Manager.
Filter Type	The type of filter applied to the search criteria. For example, Equal or Like.
Value	The value entered in the search field.
Only show percentages above: Use the slider to limit the results displayed in the table based on the percentage of the population to which the results apply. For example, if you are only interested in entitlements that apply to at least forty percent (40%) of the population searched, click the slider and move it to that percentage, or type the percentage in the field to the right.	
Entitlement Information: Click a value to display a list of all identities to whom that entitlement is assigned.	
Name	The name of the attribute from which this entitlement was derived. Attributes used to define entitlements are specified during configuration.
Value	The value assigned to the attribute. Click a value to expand a list of users to whom the entitlement is assigned.
Percent of Population	The number of identities assigned to that value of that attribute on this application expressed as a percentage of all identities that have an account on the application.

Use the results to analyze the entitlements that exist within your enterprise. The Group and Analyze feature enables you to group entitlements within an application and generate results based on that group. This feature enables you to see how assigning multiple entitlements to a role can impact access within the application.

To group and analyze, select multiple entitlements and click **Group and Analyze**. The results are displayed below the entitlements table. Click a group to see the details for the entitlements within. You can perform analysis multiple times on entitlements or on the groups created.

Save the Profile:

When you are satisfied with the information you have mined and analyzed, click **Create Role**. You must enter an name for the new role, optionally a type and description, and click **Save** to return to the Role Viewer.

Additional Information

From the Role Editor you can add additional profiles, edit the role or save the role and return to the Role Viewer. See “Role Editor Page” on page 327.

Role Mining

Role Mining is used to create roles based on specified criteria in an existing enterprise. IdentityIQ separates role mining into the following categories:

- "IT Role Mining" on page 338
- "Business Role Mining" on page 340

The IT Role Mining panel generates roles in bulk. The population of identities from which to mine can be restricted by IPOP or by String, boolean, or integer attributes (multi-valued are not supported at this time).

The entitlements from which roles are generated are defined on a by-application basis. When an application is added to the mining analysis, all of its entitlements are added to a box to the right. Users can prevent the entitlements from being considered in the analysis by clicking the “x” next to them.

The population size is restricted by the defined identity population as well as the applications under consideration. The current population size is presented along with a warning that mining details are not available for large populations.

You can restrict the roles that are generated by specifying a minimum number of identities and entitlements per role.

Select IT Role Mining or Business Role Mining from the Create New drop-down list to create and launch a new role mining task. Alternatively, you can select an existing template from the Role Mining Template panel and use the predefined criteria in your role mining task.

Note: Names are required when creating role mining templates. When you edit an existing template, you are given the choice to either change the existing template or create a new template. If you create a new template you are require to give it a new name.

IT Role Mining

IT Role Mining creates roles based on the mining of entitlements within the enterprise. These roles typically model the IT privileges required to perform a specific function within an application or other target system. Using a configurable algorithm, IdentityIQ searches for access patterns to determine logical groupings of entitlements.

The mining task generates or updates a single IT role with entitlements that are mined from a user population specified by groups, applications, or an identity filter. A threshold percentage limits the entitlements that are added to those held by a percentage of the population that exceeds the threshold.

Create New IT Role Using Role Mining

Use the Create New drop-down list at the top-right corner of the page and select IT Role Mining. Input your mining criteria in the IT Role Mining panel. View "Role Management - IT Role Mining Field Descriptions" on page 339 for details about the IT Role Mining panel.

Table 80—Role Management - IT Role Mining Field Descriptions

Field Name	Description
Owner	Enter a valid user or workgroup. Typing the first few letters of a name displays a list of all of the user and workgroup names in the system containing that letter combination. You can select from the displayed list.
Identities to Mine	Search By Attributes – Input the attribute data to target specific identity criteria used in the role mining task. Search By Population – Select a population on which the role mining task is run. Note: Selecting a population automatically filters the applications to those included in the selected population.
Applications to Mine	Specify the application(s) on which to focus the mining task.
Entitlements to Exclude	Select any entitlements that are associated with the application to exclude in the role mining task. All other entitlements are used as part of the role mining criteria.
The size of the population to be mined is currently X identities	The variable value of the total number of identities used in the role mining task based on the current mining criteria.
Minimum Identities per Role	Specify the minimum number of Identities, who meet the role mining criteria, that are required to create this role.
Minimum Entitlements per Role	Specify the minimum number of entitlements, which meet the role mining criteria, that are required to create this role.
Maximum Groups to Mine	Specify the maximum number of groups (candidate roles), which can be generated using this role mining criteria. Note: The role mining task fails if the number of candidate roles discovered exceeds the number specified in this field.

Once you have entered your criteria, click Save to save your selections as an IT Role Mining template. Click Save and Execute to save the template and run the role mining task. Enter the name of your role mining template then click OK.

Use An Existing IT Role Mining Template

Use or edit an existing IT Role Mining template to generate a role based on previous criteria by clicking a template name in the Role Mining Templates panel on the Role Mining tab.

Click **View Latest Mining Results to view the results of the most recent mining task for this template**.

Any changes to the template are saved for this template unless the template name is changed. Once you have entered your criteria, click Save to save your selections. Click Save and Execute to save the template and run the role mining task. Executed mining tasks appear on the Role Mining Results tab.

Note: Names are required when creating role mining templates. When you edit an existing template, you are given the choice to either change the existing template or create a new template. If you create a new template you are require to give it a new name.

Business Role Mining

Business role mining within IdentityIQ facilitates the creation of organizational groupings based on identity attributes – for example department, cost center or job title. The business role mining supports multiple configuration options to assist users in generating new roles. The criteria used to generate the business role can be saved as a template for future use. After the mining task is completed, the new roles are added to the Role Viewer where they can be modified as necessary.

The Business Role Mining panel generates roles from identity attributes and entitlements. The generated roles are either organized into a hierarchy based on identity attributes of the users from which the roles are mined or they are generated in a flattened manner. From there they are moved into either an existing container role or one that was newly created.

Entitlement mining is optionally performed on the generated business roles. These entitlements are either directly attached to those business roles or place in newly created IT roles that are then added to the business roles' Permits or Requires lists.

Once you have entered your criteria, click Save to save your selections as a Business Role Mining template. Click Save and Execute to save the template and run the role mining task. Enter the name of your role mining template then click OK. When the task is launched a success message dialog is displayed.

If you perform role mining on the same role consecutive times, the process does not modify owner, assigned scope, description, type, selector, or the disabled attributes on consecutive runs. Sub roles can be added on consecutive runs, but not removed. Mining for entitlements does not change. The process mines and associates entitlements. If a role is enabled and mining is run again, the role remains enabled, and entitlements can be granted with no approval process. If a role is disabled before the repeated mining is run, the role remains disabled.

To review the results of the mining task, click **View Latest Mining Results**. See “Role Mining Results” on page 342.

The roles generated by the mining task are displayed on the Role Viewer tab.

Once the roles are created and active they can be used just like any other roles.

Note: Roles created through business role mining are disabled by default.

To clear the role mining form, click Reset Mining Form.

Table 81—Role Management - Business Role Mining Field Descriptions

Field Name	Description
General Settings:	
Name	The name of the business role mining routine. The name created here is used to identify the settings used in the event the same role mining routine is reused in the future.
Compute Population Statistics	Compute statistics for the mined roles and display them in the task result.
Perform Analysis Only (no roles are generated)	Perform the role mining for analysis purpose only. No roles are generated when this mining is complete. See the results of the task on the Task Results tab of the Tasks page.
Hierarchical Settings:	

Table 81—Role Management - Business Role Mining Field Descriptions

Field Name	Description
Generate a New Root Container Role	Generate a container role into which all generated roles should be placed.
Specify an Existing Root Container Role	Select an existing role into which all the newly generated roles should be place.
Generate a Role Hierarchy from the Identity Mining Attributes	Generate a role hierarchy. Each attribute generates its own level in the hierarchy, and that level contains the roles whose names match the values for that given attribute.
Ordered Identity Mining Attributes	Arrange the list of attributes used to order the hierarchy of the generated roles. Users are assigned the role based on this list's ordering. For example if the list order is 1. Region, 2. Location, 3. Department then all users in the same department for a given location in a given region are assigned that role.
Role Settings:	
Type of Business Roles to Generate	Type of role generated by the task. Note: This option is hidden when the “Perform Analysis Only” is selected on the business role mining page.
Owner	Enter a valid user. Typing the first few letters of a name displays a list of all of the user names in the system containing that letter combination. You can select from the displayed list. Note: This option is hidden when the “Perform Analysis Only” is selected on the business role mining page.
Minimum Number of Users per Role	Minimum number of users who must the mining criteria before a role is generated.
Naming Algorithm	The filter-based naming algorithm concatenates all the attributes, separated by periods, to generate role names. The generic UID naming algorithm generates random role names. Note: This option is hidden when the “Perform Analysis Only” is selected on the business role mining page.
Prefix to Apply to Generated Role Names	Prefix to add to the generated role names. Note: This option is hidden when the “Perform Analysis Only” is selected on the business role mining page.
Disable Generated Roles	Disable all newly generated roles upon creation. This enables you to review and modify the roles if necessary before they are available for use.
IT Settings:	

Table 81—Role Management - Business Role Mining Field Descriptions

Field Name	Description
Mine for Entitlements on Generated Business Roles	Mine for entitlements as part of this task.
Attach Mined Profiles directly to Business Functional Roles	Attach mined profiles directly to the generated roles. If this option is not selected new IT roles are created to hold the entitlements and these IT roles are added to the generated roles' Permits or Requires list based on the selection below.
Type of IT Roles to Generate	Type of role that is generated to hold the entitlements.
Business Roles' Relationship to Mined IT Roles	Determines if the newly created IT roles are added to the generated roles' Permits or Requires list.
Entitlement Source Applications	Applications to mine for entitlements.
Percentage Threshold for Inclusion of an Entitlement	Specify the minimum inclusion threshold that an entitlement must meet before it is included in the role.

Use An Existing Business Role Mining Template

Use or edit an existing Business Role Mining template to generate a role based on previous criteria by clicking a template name in the Role Mining Templates panel on the Role Mining tab.

Click **View Latest Mining Results to view the results of the most recent mining task for this template**.

Any changes to the template are saved for this template unless the template name is changed. Once you have entered your criteria, click Save to save your selections. Click Save and Execute to save the template and run the role mining task. Executed mining tasks appear on the Role Mining Results tab.

Note: Names are required when creating role mining templates. When you edit an existing template, you are given the choice to either change the existing template or create a new template. If you create a new template you are require to give it a new name.

Role Mining Results

The Role Mining Results tab displays a table containing information about the role mining tasks run in IdentityIQ. Use the filtering tools to narrow down the viewable results by name, start / end date and result. Click a line item in the table to view the details of the mining result.

Right-click a line item to open a sub menu with different options depending on the role mining type. Business Role mining sub-menu options include View Results and Delete. IT Role Mining sub-menu options include View Results, Export to CSV, and Delete.

Table 82—Role Management - Role Mining Results Field Descriptions

Field Name	Description
Name	The name of the role mining template used for the task.
Date Complete	The date the role mining task completed.
Result	The result of the role mining task. Note: Click the refresh button at the bottom of the panel if the task status is "Pending". Right-click the task and select Delete to remove it from the Role Mining Results tab.
Owner	The identity named as owner of the role mining template.
Type	The type of role mining task.

Viewing the information and actions available on the role mining result details varies depending on the role mining type.

IT Role Mining Results Details

The IT Role Mining Results Details page displays a table containing a visual representation of the available unique roles generated based on the criteria used in the role mining task. Click a line item to highlight that row. Right-click the row to bring up a sub-menu from which you can select either View Group Summary, Create Role, or View Population. Click View List of Mining Results to return to the previous page.

Group Summary

The Group Summary window displays a quick view of the application and entitlements which make up that group.

Create Role

The Create Role window displays information about the role and its entitlements which were generated by the role mining task. Additional changes can be made here prior to committing to the role creation.

Table 83—IT Role Mining Results - Create Role Field Descriptions

Field Name	Description
Name	Input the name of the role being created.
Owner	The owner of the role being created.
Scope	Select a scope from the drop-down list. Only scopes that you control are displayed in the list. Scope is used to determine the objects to which a user has access. If scoping is active, identities can only see objects that they created or that are within the scopes they control.
Container Role	Select a container role from the drop down list in which to have the created role placed.
Description	Enter a brief description of the role.

Table 83—IT Role Mining Results - Create Role Field Descriptions

Field Name	Description
Direct Entitlements	Displays the entitlements that were mined as a result of the role mining criteria entered. Click the "X" icon to remove any entitlements. Note: No entitlements can be added. Entitlements can only be removed from the list. At least one entitlement must be included to successfully create a role.
Inherited Roles	Select from the drop-down list the roles, if any, in which this role is a member.
Entitlements from Inherited Roles	Displays the entitlements included in the inherited role. Click the "X" icon to remove any entitlements.

Click Save to complete the role creation or Cancel to close the window. The new role is available on the Role Viewer tab.

View Population

The View Population window displays information about the identities in IdentityIQ which match the criteria used by the role mining task. The information displayed in this table is defined when IdentityIQ is configured for your enterprise. By default the table displays Name, First Name, Last Name and Manager. Use the drop-down list at the top of the window to filter the results to display identities that match the criteria exclusively or those that match but have additional entitlements.

Business Role Mining Results Details

Click a Business Role Mining type line item to open the Latest Mining Results window for that mining task. The window displays detailed information on the roles generated based on the criteria used in the role mining task.

Table 84—Business Role Mining Results - Latest Mining Results Window Field Descriptions

Field Name	Description
Details	
Name	The name of the role which was created.
Type	The type of the role which was created.
Description	A brief description of the role which was created.
Status	Current status of the role mining task.
Started By	Displays the name of the person that launched the role mining task.
Started	Displays the date and time on which the mining task was started.
Completed	Displays the date and time on which the mining task was completed.
Business Role Mining Attributes	
Attribute	Displays information regarding the following topics: Identity Mining attributes — attributes selected in the mining criteria. Roles mined — total number of roles mined based on the provided mining criteria. Roles updated — number of roles updated as a result of the latest mining task. Coverage of mined roles — displays the percentage of comparative roles used in the mining task based off of the mining criteria.

Working with the Role Manager

Use the following sections to work with roles in the Role Manager. These sections enable you to create and edit roles and profiles, perform role analysis, and approve new or modified roles.

How to Create or Edit a Role From the Role Management Page

Use the following procedure to edit existing roles or create new roles. Roles can also be created from certifications and role mining.

Use the approval function to open approval work items for role owners. See “How to Approve Role Changes” on page 351.

Use the impact analysis function to create a report that provides details on the impact these changes can have on the rest of your product implementation. See "How to Perform Impact Analysis" on page 351.

Procedure

1. Access Role Management.
   Click or mouse over the Define tab and select **Roles**.
2. Click a role to edit.
   — OR — can
   Select Add to create a new role.
3. Enter the role information. This information is used throughout the product.
   Name — A descriptive name of this role.

   Note: Role names with single quotation marks, double quotation marks, or commas are not supported.
   Type — The type of role being created. For example, organizational, business, or IT.
   Role type definitions are customizable and created as part of the configuration process.

   Owner — The name of the owner for this role. Entering the first few letters of a name displays a select list of valid users and workgroups with names starting with those letters. Select a name from the list.

   Description — A detailed description of the role.
4. *Optional:* Define activation events for the role being created.

   Note: Only one activation or deactivation event can be defined at a time.
   a. Click **Add Event** to display the Add New Event dialog.
   b. Manually enter a date or click the calendar icon to select a date.
   c. Select Activate or Deactivate from the **Action** drop-down list.
   d. Click **Save** to return to the Role Editor page.

   Select an event and click Delete to remove the event.

5. *Optional:* Define an assignment rule for the role being created.

   - **Match List** — define a list of entitlements to determine role assignment.
     For attributes select an attribute from the drop-down list and type a value.
     For permissions, type the name (target) and value (right).

   Note: **If Null is selected, the associated value text box is disabled. When the is null match is processed, the term matches users on the chosen application who have a null value for that attribute/permission.**

   - **Filter** — enter a custom XML database query to define user for this role.
   - **Script** — enter a custom script for role assignment. Scripts are similar to rules, but the source is stored with the role and can be edited from this page.
   - **Rule** — select an existing rule from the drop-down list.
   - **Population** — select a population from the list. Members of that population are assigned the role. Populations are generated as the results of identity searches.

6. *Optional:* Click **Modify Permitted Roles** in the Permitted Roles panel and modify the list of roles permitted by this role.
   a. Enter the first few letters of a role name in the **Select a role** field and select a role from the selection list.
   b. Click **Add** to add the role to the membership list.
      Add as many roles as required.
   c. Click **Save**.
7. *Optional:* Click **Modify Required Roles** in the Required Roles panel and modify the list of roles required by this role.
   a. Enter the first few letters of a role name in the **Select a role** field and select a role from the selection list.
   b. Click **Add** to add the role to the membership list.
      Add as many roles as required.
   c. Click **Save**.
8. *Optional:* Click **Modify Inheritance** in the Inherited Roles panel and modify the list of roles of which this role is a member. This role inherits entitlements from any role to which it is a member.
   a. Enter the first few letters of a role name in the **Select a role** field and select a role from the selection list.
   b. Click **Add** to add the role to the inheritance list.
      Add as many roles as required.
   c. Click **Save**.
9. Create new profiles or edit existing profiles from the Entitlements panel.
   Profiles created for this role are inherited by any role that is a member of this role.
10. Optional: Click **Add Provisioning Policy** in the Provisioning Policy panel.

11. Take one of the following actions:

    - Click **Submit** to save the role or, if the approval work flow is active, open an approval work item for the specified role owner.
      The approval feature is only available if the work flow was activated during configuration.
    - Click **Submit with Impact Analysis** to create a report that provides details on the impact these changes can have on the rest of your product implementation and open an approval work item if the approval work flow is active.
    - Click **Check Policy Conflicts** to display any policy violations created by changes made on this page. Policy checking is only available if impact analysis has been run.

Additional Information

To work with profiles associated with a role see:

- "How to Create or Edit a Profile" on page 347
- "How to Approve Role Changes" on page 351

How to Create a Role From a Role Creation Request

Use the following procedure to create roles from role creation request work items. Role creation request work items can be generated through the certification process.

Note: Approval is only required if the approval work flow is active. If approval is not required roles are added directly from the Create Role dialog.

Procedure

To create a new role from a role creation request, do the following:

1. Click the work item requesting the role in your inbox.
2. Review the information in the work item and do one of the following:
    - **Forward** — forward the work item to another authorized user to make the decision on the role. Optionally add comments on the **Forward Comments** dialog.
    - **Reject** — reject the proposed role. Optionally add comments on the Rejection Comments dialog.
    - **Approve** — continue with step 3 to proceed with the approval process.
3. *Optional:* Edit the name of the role.
4. Optional: Edit the owner of the role.
   Entering the first few letters of a name or workgroup displays a select list of valid IdentityIQ users and workgroups with names starting with those letters. Select a name from the list.
5. *Optional:* Edit or enter a description of the role being created.
6. Click **Approve** to display the **Approval Comments** dialog.
7. Add comments if they are required and click **Approve** to create this role.

How to Create or Edit a Profile

A profile is a set of entitlements on a specific application. An entitlement is either a specific value for an account attribute, most commonly group membership, or a permission. Profiles are specific to one role.

To Edit a Profile:

1. Access the Entitlement panel from the Role Editor page.
2. Edit the entitlement information.

   Note: If you change the application with which this profile is associated, all entitlements that you have created are removed.

3. Add or delete attribute rules and permissions.
   See "Role Editor - Edit Entitlement Panel" on page 330 for descriptions of the fields in each section.
4. Click **Save** to return to the Role Editor.

To Create a Profile:

- Create a new profile — See “How to Create a New Profile” on page 349.
- Create a profile using entitlement mining — “How to Create a Profile Using Entitlement Analysis” on page 348.

How to Create a Profile Using Entitlement Analysis

IdentityIQ supports the creation of roles based on the mining of entitlements within the enterprise. These roles typically model the IT privileges required to perform a specific function within an application or other target system. Using a configurable algorithm, IdentityIQ searches for access patterns to determine logical groupings of entitlements.

Entitlement analysis enables you to search for entitlements based on specific application and identity information. This feature enables you to create meaningful profiles without having to remember every entitlement on every application, or be familiar with the access assigned to each employee in your enterprise.

Entitlement mining also enables you to analyze the entitlement information collected to further refine the profiles you are creating before saving.

Procedure

Creating a profile using entitlement analysis actually involves three distinct phases:

- Searching for entitlements
- Analyze the search results
- Saving the profile

Search for Entitlements:

1. Access the Create Profile from Entitlement Analysis panel.
   Click Create in the Profiles panel of the Role Editor and select New Profile From Entitlement. Profiles can only be added within a role. See “How to Create or Edit a Role From the Role Management Page” on page 345.
2. Select the application on which to search for entitlements.
3. *Optional:* Narrow your entitlement search using the Identity Attribute fields.
   The Identity Attribute fields displayed are dependent on the identity attributes defined during configuration.
4. Click **Search** to begin the role analysis based on the specified criteria.

Analyze the Search Results:

The search returns the following information:

Note: The entitlement analysis search only returns those entitlements based on account or group attributes, not those based on permissions.

Table 85—Entitlement Mining Search Results Descriptions

Column	Description
Search Parameters:	
Attribute	The criteria used to define this search. For example, Application, Last Name, or Manager.
Filter Type	The type of filter applied to the search criteria. For example, Equal or Like.
Value	The value entered in the search field.
Only show percentages above: Use the slider to limit the results displayed in the table based on the percentage of the population to which the results apply. For example, if you are only interested in entitlements that apply to at least forty percent (40%) of the population searched, click the slider and move it to that percentage, or type the percentage in the field to the right.	
Entitlement Information: Click a value to display a list of all identities to whom that entitlement is assigned.	
Name	The name of the attribute from which this entitlement was derived. Attributes used to define entitlements are specified during configuration.
Value	The value assigned to the attribute. Click a value to expand a list of users to whom the entitlement is assigned.
Percent of Population	The number of identities assigned to that value of that attribute on this application expressed as a percentage of all identities that have an account on the application.

Use these results to analyze the entitlements that exist within your enterprise. The Group and Analyze feature enables you to group entitlements within an application and generate results based on that group. This feature enables you to see how assigning multiple entitlements to a profile can impact access within the application.

To group and analyze, select multiple entitlements and click **Group and Analyze**. The results are displayed below the entitlements table. Click a group to see the details for the entitlements within. You can perform analysis multiple times on entitlements or on the groups created.

Save the Profile:

When you are satisfied with the information you have mined and analyzed, click **Create Profile**. You must enter an name for the new profile, optionally a description, and click **Save** to return to the Role Editor.

Additional Information

From the Role Editor you can add additional profiles, edit the role or save the role and return to the Role Viewer. See “Role Editor Page” on page 327.

How to Create a New Profile

Use one of the following procedures to create a new profile.

P**rocedure 1**

Note: Click Simple View if you are in the advance view. The Simple View might not be available in all roles.

1. Click **Add** in the Entitlements Panel.

2. Select the application on which to apply this profile from the **Application** suggestion list.
   Enter the first few letters of an application name and select the application from the suggest list.
3. Select an account attribute and then an entitlement from the drop-down lists.
4. Click **Save** to return to the Role Editor.

Procedure 2

1. Click **Advanced View** in the Entitlements Panel.
2. Click **Create** in the Profiles panel of the Role Editor and select **New Profile**. Profiles can only be added within a role. See “How to Create or Edit a Role From the Role Management Page” on page 345.
3. Enter a description for the profile.
4. Select the application on which to apply this profile from the **Application** suggestion list.
   Enter the first few letters of an application name and select the application from the suggest list.
5. Add **Attribute Rules** and **Permissions** to the profile. To use the filter, see "Creating Attribute Rules" on page 350. For an explanation of the permission options, see "Creating Attribute Permissions" on page 350.
6. Click **Save** to return to the Role Editor.

Additional Information

From the Role Editor you can add additional profiles, edit the role or save the role and return to the Role Modeler page. See “Role Editor Page” on page 327.

Creating Attribute Rules

Use the Attribute Rules function to add and combine filters to define your profiles. Apply qualifiers to attributes within filters to limit the values returned and then use grouping and AND\OR operations to create the rules that make up the profile.

Add a Filter:

Create the filters that make up the attribute rules.

- **Field** — select an attribute value from the drop-down list. This list contains all of the attributes mapped from the selected application.
- **Search Type** — the qualifier to associate with the value, for example equals or like.
- **Value** — the value of the attribute.
- **Ignore Case** — specifies if case should be factored into the query.

Filter(s):

The Operations drop-down list enables you to specify AND/OR relationships between the filters in the list. You can use multiple layers of filter grouping containing AND\OR operations to create complex attribute rules. For example, you can create an attribute rule that returns all users who are in payroll OR human resource AND located in Chicago.

Creating Attribute Permissions

Use the permissions panel to add permissions to the profile. Permissions define rights on targets on the application. Select rights from the rights lists, for example, create, read, update, delete, execute, and specify the target attribute in the Target field. Use the Shift and Ctrl keys to select multiple rights.

How to Approve Role Changes

When roles are created or edited, they might require approval from the designated owner before they become active. Work items are created and sent to the owners when approval is necessary. Use this procedure to review and approve or reject role changes.

Role analysis and role approval are an important part of the overall role life-cycle management. Role analytics and approval, both for new or modified roles are controlled thought business processes configured for your implementation of IdentityIQ.

Procedure

1. Click an approval work item in your Inbox on the to display the Approval page.
2. Review the summary information of the work item.
3. Review the comments associate with the work item and, optionally, add comments.
4. Review the details sections.
   Modification approvals — review the changes in the Modified Role or Modified Profile table and make a decision.

   Creation approvals — review the information in the New Role or New Profile panel, make the necessary modifications, and make a decision. Some of the information is read only.
5. Click **Review Pending Changes** to display the Role Editor and review the changes proposed for the role.
6. Make a decision.
   Approve — approve the creation or modification. Add comments if needed and confirm the approve on the Approval Comments dialog.

   Reject — reject the request for approval on the creation or modification. Add comments if needed and confirm the rejection on the Rejection Comments dialog.

   Forward — forward the approval work item to another user. Entering the first few letters of a name displays a select list of valid users with names starting with those letters. Select a name from the list. Add comments if needed.

   Cancel — cancel the work you have done on the work item and return to the.

How to Perform Impact Analysis

Use the impact analysis function to create a report that provides details on the impact these changes can have on the rest of your product implementation.

When you click the **Submit with Impact Analysis** from the Role Editor, the changes are rolled into a work item that is assigned to you, an analysis task is launched, and a link is created inside the work item that points to the task results. You can navigate from the work item to the task result to check on the status of the task as it is running.

Impact analysis can also be performed from the Task page using the Role Overlap Analysis tasks. Overlap analysis returns information on the following overlap facets:

- Attributes — overlap between extended attributes and a some system attributes
- Local Assignment — overlap between assignment rules and profiles defined directly on the role (not inherited)
- Hierarchal Assignment — overlap between both local and inherited assignment rules and profiles
- Local Provisioning — overlap between provisioning side effects defined directly on the role
- Hierarchical Provisioning — overlap between both local and inherited provisioning side effects

Note: The Assignment and Provisioning numbers are the same for simple roles. However, the numbers are different when there are manually written provisioning plans. The numbers are also different when the profiles use OR terms because provisioning only picks the first terms using OR.

Procedure

1. Click an impact analysis work item in your inbox on the to display the Role Approval page.
2. Review the summary information of the work item.
3. Review the comments associate with the work item and, optionally, add comments.
4. Review the details of the changes being analyzed by the impact analysis task associated with the work item.
5. Click **Click to view analysis task results** to display the task results page containing the actual impact information obtained by the task.
6. Review the impact information and click **Return to Work Item** to return to the work item and make a decision on the request.
7. Make a decision.
   Approve — apply the creation or modification based on the content of the impact analysis task results.
   Reject — discard any changes made to the role based on the impact analysis results.
   Forward — forward the impact analysis work item to another user. Entering the first few letters of a name displays a select list of valid users with names starting with those letters. Select a name from the list. Add comments if needed.
   Cancel — cancel the work you have done on the work item and return to the.

Multiple Role and Account Assignment

IdentityIQ allows roles to be assigned to an identity more than once and applied to different sets of accounts associated with the identity. A second feature allows a role assignment to apply to multiple accounts on the same application.

Multiple Role Assignment

A system and a role-specific option allows a role to be assigned to an identity more than once and have the associated entitlements apply to different accounts.

The model that is used to persist role assignment on an identity includes the accounts to which the role assignment is provisioned. This model is referred to as target account memory. The role assignment can also contain an assignment note that describes why the assignment exists. The assignment note is useful for

differentiating multiple assignments. For example, you can have one assignment with a note of Standard Account and a second assignment with a note of Privileged Account.

When a role is assigned, the applicable accounts are selected automatically using rules or through an interactive user interface. The selection of accounts can optionally be a directive to create a new account. Account selection rules can be defined on a role that can contain entitlements that can be provisioned from profiles to automate the selection of applicable accounts. There can be a general rule for the role as well as a rule for every application included in the role profiles.

For Lifecycle Manager access requests, the requestor is prompted, if they are required by the configuration settings, to select the accounts to use for the request. This occurs if multiple accounts already exist on the relevant applications or IdentityIQ is configured to allow a new account to be created and account selection rules did not automatically select the appropriate accounts. The requestor can enter an assignment note during account selection.

When role assignment rules are processed during the Identity Refresh task, the default behavior is to skip any role provisioning that does not explicitly define the target account and to report the number of times provisioning was skipped. The Identity Refresh task can be configured to create required account selection work items if appropriate account selection rules are not defined, but care should be to be taken to ensure that this does not create an inordinate number of work items. To prevent the need for manual interaction, the best practice is to have completely defined account selection rules for all profiles associated with rule-based role assignments where multiple role assignment is allowed.

Details about the accounts that an assigned role applies to and the optional assignment note are displayed in the appropriate user interfaces including: Entitlements tab of the View Identity page, Certifications pages, Lifecycle Manager Current Access, Lifecycle Manager approval work items, and Manage Access Request details. Additionally, these user interfaces have a role listed multiple times if the role is assigned more than once.

Multiple Application Accounts in an Assignment

In a standard role assignment, a role can provision to no more than one account on a specific application. If the role hierarchy contains more than one role that targets the same application, the entitlements for the assigned role are all provisioned on the same account.

An option can be specified on any role that can be contained on a permitted or required list of another role, or any role that contains entitlements that can be provisioned from profiles or any role that contains a provisioning policy, that allows the entitlements in that role to be provisioned to a selected account or to a newly created account. If there is more than one role that can be provisioned that uses this option in the assigned role hierarchy, a different target account (including creating a new account) can be selected for each role.

Role Detection

All detected roles store information about the accounts and entitlements that fulfilled the detection. Detection recognizes and persists if a detected role was part of an assignment. For example, explicitly requested in a Lifecycle Manager access request.

A role can be detected more than once if there are role assignments targeting different accounts on the same application. For example, if assigned role A and assigned role B both have required role R, but different target accounts were selected for A and B, there are two detections of R. One for the accounts selected for A and one for the accounts selected for B. This model is necessary to accurately show which accounts and entitlements are included in each role assignment.

Hard and Soft Permitted Roles

A **hard permitted role** is a role that is requested through IdentityIQ. A **soft permitted role** is a role that is discovered through aggregation and entitlement correlation, but was not explicitly requested or provisioned using IdentityIQ.

When a role that contains hard permitted roles is unassigned and de-provisioned, the hard permitted roles is also de-provisioned if there are no other dependencies on those roles. If a role containing soft permitted roles is unassigned and de-provisioned, the soft permitted roles are not de-provisioned.

Identity Role Assignments

Role assignments have an assignment id that is used to uniquely refer to the assignment. The user interface does not display this assignment id, but any code that references an assignment needs to use the id to keep a reference from being ambiguous.

When a permitted role is requested through Lifecycle Manager, IdentityIQ records the request in the RoleAssignment model by placing a nested RoleAssignment for the permitted role inside the RoleAssignment for the assigned role. This process defines a hard permitted role.

The identity assignedRoles and assignedBundleSummary attributes are a unique list of roles, and if a role is assigned multiple times, the role is in this list only one time. The identity roleAssignments attribute can contain multiple items for the same role if the role is assigned multiple times.

Existing methods on the identity object related to role assignments remain for backward compatibility, but are marked deprecated and can return incomplete results if multiple assignments are enabled.

Provisioning Plans

If multiple assignments are enabled and exist, a provisioning plan to modify assignments must specify an assignment id to prevent ambiguity. When an assignment is being added and the intention is to create a second assignment, a special assignment id token of new is used.

A single attribute request can contain a list of roles that are to have their assignments changed. When multiple assignments are enabled and exist, each role must be contained in a separate attribute request so that an assignment id can be specified.

The provisioner remains backward compatible and continues to process provisioning plans without assignment ids or role lists.

If multiple assignments are enabled, it is imperative that provisioning plans are well formed and include the correct data to impact the desired change.

When multiple assignments for the role exist, a provisioning plan that includes a request to remove a role assignment by name without an assignment id removes one indeterminate role assignment. When an assignment for a role already exists, a provisioning plan that includes a request to add a role assignment without an assignment id or a new token selects one indeterminate role assignment and provision any missing entitlements.

Automated Propagation of Role Changes to Role Members

When managing role model in IdentityIQ any changes to the role or delete a role, would propagate to all identities that are currently assigned to the said role. This allows you to use role model as a true authoritative source for requested access. To allow propagation of role changes, you must set a configuration flag in System Setup.

If the above flag is set, role changes are provisioned. Change propagates to all identities that have that role and any revokes or additions that need to take place occurs. Examples of role changes include:

- role requirements changes, such as adding or removing an entitlement
- role Inheritance changes, such as disabling or enabling role
- changes to the list of required roles are needed

Propagation of changes are set to a **RoleChangeEvent** table and picked up by **Propagate Role Changes** task based on a created time.

Note: When a role is deleted, it is marked for deletion and the user is unable to make edits to the role. When a role is deleted, all entitlements assigned to that role are deleted.

Chapter 21: Define Policies

Policies are comprised of rules used to enforce any policies, separation of duty, activity or risk, defined within your enterprise. Policies are defined and used to monitor for identities that are in violation of those policies. For example, a separation of duties policy (SOD) can disallow one identity from requesting and approving purchase orders. An activity policy can disallow an identity with the Human Resource role from updating the payroll application even though the identity has view access to that application.

Custom policies are any policies that were created outside of IdentityIQ to meet special needs of your enterprise. You cannot create a custom policy from inside the product. Use the Edit Policy page to view information about a custom policy. However, changes made here do not impact the performance of the policy.

Note: The Define Policy pages require IdentityIQ administrative capabilities.

To access Policies, from the dash board go to **Setup > Policies**.

This chapter has the following topics:

- Policies Page — View and define policies
- Edit Policy Page — Create or edit policies
- Policy Rules — Create or edit policy rules
- Working with Policies — How-to tasks

Policies Page

Use the Policies page to view existing policies and to define policies for your enterprise. To limit the number of policies displayed in the table, use the filtering options. You can filter by policy name and policy type. Enter a letter or partial name in the **Policy Names** field to display any policies with names beginning with that letter pattern.

Before you make a policy active in your production environment, you can run a simulation to test the enabled rules that are defined in the policy. See “Policy Simulation” on page 360.

Table 86—Policies Page Column Definitions

Column Name	Description
Name	The name of the policy assigned when it was defined.

Table 86—Policies Page Column Definitions

Column Name	Description
Type	The type of policy. **SOD** – separation of duties policies ensure that identities are not assigned conflicting roles. **Entitlement SOD** — separation of duties policies ensure that identities are not assigned conflicting entitlements. **EffectiveEntitlementSOD** — ensure that identities are not assigned conflicting entitlements indirectly, through other objects. **Activity** — ensure that users are not accessing sensitive application if they should not or when they should not. **Account** — ensure that an identity does not have multiple accounts on an application. **Risk** — ensure that users are not exceeding the maximum risk threshold set for your enterprise. **Advanced** — custom policies created using match lists, filters, scripts, rules, or populations.
Description	A brief description of the policy as entered when it was defined.
State	The status of the policy. **Active** — the policy is currently being used. **Inactive** — the policy is not being used.

Edit Policy Page

Use the Edit Policy page to create new policies and to edit existing policies. The Edit Policy page contains the following information:

Table 87—Edit Policy Page Field Description

Field Name	Description
Name	A descriptive name of this policy. This is the name that displays on the Policies page.
Owner	The owner of the policy. If the notification option is enabled as part of policy and identity refresh tasks, the policy owner receives an email notification for each violation of the policy by default. Entering the first letter, or letters, of a name or workgroup displays a selection list of valid users and workgroups with names containing that letter string.
Policy Violation Owner	Use to assign an owner to the violation, not just to the policy. You can also assign owners to each individual rule that makes up the policy. If you assign an owner at the rule level it overrides the policy-level violation owner. **Note: Click the "..." icon to launch the Rule Editor to make changes to your rules if needed.** If the notification option is enabled, only the owner receives a work item, the observers only receive email notifications

Table 87—Edit Policy Page Field Description

Field Name	Description
Scope	The scope for this policy. If scope is assigned, only the owner of the policy and users who control the designated scope can see this policy on the Policies page. The scope assigned to the policy does not impact the way violations are displayed, reported, or monitored. Depending on configuration settings, objects with no scope assigned might be visible to all users with the correct capabilities.
Description	A brief description of the policy and its use in your organization. **Note: You must Save the description before changing languages to enter another description.** Use the language selector to enter description in multiple languages. The drop-down list displays any languages supported by your instance of IdentityIQ. The description displayed throughout the product is dependent on the language associated with the user's browser. If only one description is entered, that is the description used by default.
Violation formatting rule	Select a violation rule from the drop-down list. Violation formatting rules are defined when your system is configured. **Note: Click the "..." icon to launch the Rule Editor to make changes to your rules if needed.**
Violation business process	Select a business process from the drop-down list. A business process specified for the entire policy is overwritten by any business process specified as part of a policy rule on the Edit Rule pages. business processes can be use to define how violation work items are assigned or how to handle the violation based on decision made on the work item.
State	The state of the policy: **Active** — use the policy to monitor roles or activity. **Inactive** — do not use the policy to monitor role or activity at this time.
Send Alerts	Activate to display the Alert Properties section. You can set alerts to be sent by email and a work item opened each time a violation is detected.
Alert Properties: Not all of the alert property options are visible initially. This section expands as options are activated.	
Initial Notification Email	The email template used for the initial notification of the policy violation and work item assignment.

Table 87—Edit Policy Page Field Description

Field Name	Description
Escalation	Specify a level of escalation for this policy. **None** — after the initial alert no further messages are sent and the work item is never escalated. **Send Reminders** — email reminders are sent periodically until the work item is complete. **Reminders then Escalation** — email reminders are sent periodically until the work item is complete or, if the work item is not completed in a timely manner, the work item is escalated. **Escalation only** — the work item is escalated after a specified time period with no notifications or warning being sent.
Open Work Item	Select to automatically generate a work item for this violation.
Days Before First Reminder	The number of days after which the first email reminder is sent.
Reminder Frequency	The number of days, or interval, between email reminders being sent.
Reminder Email Template	Template used to format the reminder email. If none is selected, a system default is used.
Reminders Before Escalation	Maximum number of reminders to send before escalation begins. If this field is set to zero, no reminders are sent and escalation begins immediately.
Escalation Owner Rule	The rule used to determine the new owner of the escalated work item.
Escalation Email	Template used to format the escalation email.
Observers	Identities to whom the email notifications and work items are sent. Enter the first letter, or letters, of an identity name to display the suggest list or click the arrow to the right of the field to display all identities and select from the list. Select as many observers as required.
Rule Table	A list of the rules contained in this policy and a description of each. Click on a rule to access the edit rule pages. Account and Risk policies do not have a separate rule page.

Policy Simulation

Note: Policy simulation runs a background task that iterates over all identities to determine if a policy violation occurs for the rule or policy. This process can be time consuming and resource intensive depending on the complexity of the policy definition and the number of identities and accounts.

Before you make a policy active in your production environment, you can run a simulation for:

- All enabled rules in policy — Click **Run Simulation** next to the **Cancel** button. To view the number of violations, click **View Simulation**.
- A single rule in a policy with multiple rules — Click the **Run Simulation** link next to the rule. To view the number of violations, click the **View Simulation** link.

When you run a simulation on a policy, the policy is saved and the test is run for all the enabled rules. The rule or rules are disabled and the status of the policy is changed to **Inactive**. To activate the policy, you must edit the policy, change the state to **Active** and save the changes to the policy.

Note: **Before testing the rule, make sure the names of rules are unique in a policy. When you run a simulation for a single the rule, only the rule is disabled. The state of the policy is NOT changed. When you run a simulation for all the enabled rules in a policy, the state of the policy is changed to inactive. To activate the policy, you must change the state to Active and save the changes to the policy.**

To work with the rules for each policy type, see “Policy Rules” on page 361.

Policy Rules

Rules are used to enforce policies. Violations on each rule in a policy, when detected, are stored in the identity cube. These violations also appear on identity score cards and enable you to identify high-risk employees and respond. You can configure policy violations to trigger a business process that immediately sends email notifications and generates work items when a violation is detected. Policy violations can be managed through certifications or through the policy violations page.

You can use the simulation option to simulate the policy rule before you make it active in your production environment. See "Policy Simulation" on page 360.

This section has the following topics:

- Edit SOD Rule Page
- Edit Activity Rule Page
- Edit Advanced Policy Rule Page

Edit SOD Rule Page

Use the Edit SOD Rule page to define new rules for separation of duty polices or edit existing rules. Rules are used to monitor roles or entitlements for conflicts of interest. This enables you to identify high-risk employees and take the appropriate action as needed.

To create or Edit a policy, see “How to Create or Edit a Separation of Duty Policy” on page 367.

To access the Edit SOD Rule Page, navigate to **Define > Policies**, select the **SOD Policy** and then scroll down to the bottom of the page. Select an existing rule from the table or click **Create New Rule.** The following information is displayed on an Edit SOD Rule page:

Table 88—Edit SOD Rule Page Field Descriptions

Field Name	Description
Summary	A brief summary of this rule. This information is displayed in the **Rules** column of the Rules table on the Edit Policy page.
Description	A brief description of the rule.

Table 88—Edit SOD Rule Page Field Descriptions

Field Name	Description
Policy Violation Owner	Use to assign an owner to the violation, not just to the policy. You can also assign owners to each individual rule that makes up the policy. If you assign an owner at the rule level it overrides the policy-level violation owner. Note: Click the "..." icon to launch the Rule Editor to make changes to your rules if needed. If the notification option is enabled, only the owner receives a work item, the observers only receive email notifications
Violation formatting rule	Select a violation rule from the drop-down list. Violation formatting rules are defined when your system is configured.
Violation business process	Select a business process from the drop-down list. A business process specified for the entire policy is overwritten by any business process specified as part of a policy rule on the Edit Rule pages.
Disabled	Enable or disable the policy
Compensating Control	A textual description of exceptions or compensating factors that apply to this rule. For example, certain policies or rules might not apply to users at the executive level in your organization. **Note: This field is for documentation purposes only. Information entered here does not impact risk scoring associated with this rule or the reporting of policy violations.**
Correction Advice	Text entered in this field is displayed if a violation of this policy appears on a certification request and is selected for revocation. Use this field to enter information that can be used by a certifier to make the correct revocation decision.
Role SOD Rules:	
Any of these roles/entitlements conflict with any of these roles/entitlements	The lists of conflicting roles that define this rule. If an identity is assigned ANY of the roles from the **Any of these** table and ANY of the roles from the **conflict with any of these** table, they are in violation of this rule and their risk score card reflects that violation. Each table can contain multiple items, but if a user has even one role in each list it is a violation of the policy.
Entitlement SOD Rule:	
First Entitlement Set Second Entitlement Set	The list of conflicting entitlements that define this rule. Add identity attributes or account attributes and permissions to create lists of conflicting entitlements. Use the **Or/And** drop-down list to determine if an identity has to match all of the items in the list or just one to be in violation of this policy.
Effective Entitlement SOD Rule:	
First Entitlement Set Second Entitlement Set	The list of conflicting entitlements that define this rule. Add identity attributes, account attributes and permissions, and target permissions to create lists of conflicting entitlements. Use the **Or/And** drop-down list to determine if an identity has to match all of the items in the list or just one to be in violation of this policy.

Table 88—Edit SOD Rule Page Field Descriptions

Field Name	Description
Run or View Simulation	Use the simulation option to simulate the policy rule before you make it active in your production environment. **Note: Before testing the rule, make sure the names of rules are unique in a policy. When you run a simulation for a single the rule, only the rule is disabled. The state of the policy is NOT changed.** **When you run a simulation for all the enabled rules in a policy, the state of the policy is changed to inactive. To activate the policy, you must change the state to Active and save the changes to the policy.**

Edit Activity Rule Page

Use the Edit Activity Policy Rule page to define new rules for activity polices or edit existing rules. Rules are used to monitor the activities performed by users within your enterprise.

To create or Edit a policy, see "How to Create or Edit an Activity Policy" on page 367.

To access the Edit Activity Rule Page, navigate to **Define > Policies**, select the **Activity Policy** and then scroll down to the bottom of the page. Select an existing rule from the table or click **Create New Rule.** The following information is displayed on the Edit Activity Policy Rule page:

Table 89—Edit Activity Policy Rule Page Field Descriptions

Field Name	Description
Activity Rule:	
Summary	A brief summary of this rule. This information is displayed in the **Rules** column of the Rules table on the Edit Policy page.
Description	A brief description of the rule.
Policy Violation Owner	Use to assign an owner to the violation, not just to the policy. You can also assign owners to each individual rule that makes up the policy. If you assign an owner at the rule level it overrides the policy-level violation owner. **Note: Click the "..." icon to launch the Rule Editor to make changes to your rules if needed.** If the notification option is enabled, only the owner receives a work item, the observers only receive email notifications
Violation formatting rule	Select a violation rule from the drop-down list. Violation formatting rules are defined when your system is configured.
Violation business process	Select a business process from the drop-down list. A business process specified for the entire policy is overwritten by any business process specified as part of a policy rule on the Edit Rule pages.
Disabled	Enable or disable the policy.

Table 89—Edit Activity Policy Rule Page Field Descriptions

Field Name	Description
Compensating Control	A textual description of exceptions or compensating factors that apply to this rule. For example, certain policies or rules might not apply to users at the executive level in your organization. **Note: This field is for documentation purposes only. Information entered here does not impact risk scoring associated with this rule or the reporting of policy violations.**
Corrective Advice	Text entered in this field is displayed if a violation of this policy appears on a certification request and is selected for revocation. Use this field to enter information that can be used by a certifier to make the correct revocation decision.
Identity Filters: **Enable you to identify which types of identities should be considered when scanning activities for violations of this policy.** **These filters can be grouped and controlled using AND\OR operations and be as simple or complex as needed.** **The Add a Filter box is used to create the individual filters, the Filter(s) box is used to view and manipulate the existing filters**.	
Operation	The operation used to control the interaction between the filters.
Field	A distinguishing characteristic associated with the identity type for which you are searching. The drop-down list contains all of the categories by which identities can be differentiated.
Search Type	The qualifier associated with the attribute value. For example, equals or is like. The choices in this drop-down list are dependent on the Field specified.
Value	The value of the attribute.
Ignore Case	Specifies if case should be a factor when scanning for the value specified.
Activity Filters: **Enable you to select which types of activities should be considered violations of this policy. You can also choose Time Periods in order to define when this activity is considered a violation of this policy.**	
Time Periods	The time periods during which the activity is in violation of the policy. For example, if some one is logging into a sensitive application on the weekends or during non-office hours it might be a violation. The time periods are configured during the deployment of IdentityIQ.
Operation	The operation used to control the interaction between the filters.
Field	A distinguishing characteristic associated with the action for which you are searching. For example, start or end date, or the data source on which the action occurred.
Search Type	The qualifier associated with the field value. For example, `equals` or `is like`. The choices in this drop-down list are dependent on the Field specified.
Value	The value of the attribute.
Ignore Case	Specifies if case should be a factor when scanning for the value specified.

Table 89—Edit Activity Policy Rule Page Field Descriptions

Field Name	Description
Run or View Simulation	Use the simulation option to simulate the policy rule before you make it active in your production environment. **Note: Before testing the rule, make sure the names of rules are unique in a policy. When you run a simulation for a single the rule, only the rule is disabled. The state of the policy is NOT changed.** **When you run a simulation for all the enabled rules in a policy, the state of the policy is changed to inactive. To activate the policy, you must change the state to Active and save the changes to the policy.**

Edit Advanced Policy Rule Page

Use the Edit Advance Rule page to define new rules for advanced polices or edit existing rules. Advanced rules are used to create advanced, custom, violation monitoring based on a variety of entitlement, filters, scripts, rules, and populations.

To create or Edit a policy, see "How to Create or Edit an Advanced Policy" on page 368.

The following information is displayed on the Edit Advanced Rule page:

Table 90—Edit Advanced Policy Rule Page Field Descriptions

Field Name	Description
Activity Rule:	
Summary	A brief summary of this rule. This information is displayed in the **Rules** column of the Rules table on the Edit Policy page.
Description	A brief description of the rule and its use in your organization.
Violation formatting rule	Select a violation rule from the drop-down list. Violation formatting rules are defined when your system is configured.
Violation business process	Select a business process from the drop-down list. A business process specified for the entire policy is overwritten by any business process specified as part of a policy rule on the Edit Rule pages.
Disabled	Enable or disable the policy.
Compensating Control	A textual description of exceptions or compensating factors that apply to this rule. For example, certain policies or rules might not apply to users at the executive level in your organization. **Note: This field is for documentation purposes only. Information entered here does not impact risk scoring associated with this rule or the reporting of policy violations.**
Corrective Advice	Text entered in this field is displayed if a violation of this policy appears on a certification request and is selected for revocation. Use this field to enter information that can be used by a certifier to make the correct revocation decision.

Table 90—Edit Advanced Policy Rule Page Field Descriptions

Field Name	Description
Selection Method: **The selection method used when scanning for and assigning policy violations.**	
Match List	A list of entitlements that define a policy violation. An identity that is assigned the entitlements in this list is in violation of this policy.
Filter	A custom filter (XML database query) used to define a rule for this policy.
Script	A custom script used to define a rule for this policy.
Rule	The rule selected from the rules list.
Run or View Simulation	Use the simulation option to simulate the policy rule before you make it active in your production environment. **Note: Before testing the rule, make sure the names of rules are unique in a policy. When you run a simulation for a single the rule, only the rule is disabled. The state of the policy is NOT changed.** **When you run a simulation for all the enabled rules in a policy, the state of the policy is changed to inactive. To activate the policy, you must change the state to Active and save the changes to the policy.**

Working with Policies

To create a new policy, use the **Create new policy** drop-down menu. Select a type from the drop-down menu to display the Edit Policy page. To work with an existing policy, click on that policy row in the table or right-click on the policy and select **Edit** from the drop-down menu.

To remove a policy, right-click on the policy and select **Delete** from the drop-down menu.

This section has the following topics:

- How to Create or Edit a Risk Policy
- How to Create or Edit an Account Policy
- How to Create or Edit a Separation of Duty Policy
- How to Create or Edit an Activity Policy
- How to Create or Edit an Advanced Policy

How to Create or Edit a Risk Policy

Use the SailPoint-provided risk policy to set a maximum risk threshold for identities before they are considered in violation of your compliance standards. From the Policies page, click the risk policy in the Policies table to display the Edit Policy page and enter the **Composite score threshold**.

See “Policies Page” on page 357 and “Edit Policy Page” on page 358

You can create additional risk policies, but only one is operational within IdentityIQ at any time.

How to Create or Edit an Account Policy

Use the SailPoint provided account policy to ensure that no identities have multiple accounts on any of the applications within your enterprise. Use the Edit Policy page to activate the account policy and add information such as a name and owner.

See “Policies Page” on page 357 and “Edit Policy Page” on page 358

How to Create or Edit a Separation of Duty Policy

Policies are created using the Edit Policy and Edit SOD Rule pages. Use this procedure to create new policies.

Procedure

1. Click or mouse over the Define tab and select **Policies**.
2. ***Optional***: Use the filtering options to limit the number of policies displayed in the table.
   You can filter by both policy name and policy type.
3. Select either Role SOD or Entitlement SOD from the **Create new policy** drop-down list or click on an existing policy to display the Edit Policy page.
4. Enter the policy information.
   See "Edit Policy Page" on page 358 for detail description of the Edit Policy page.
5. Right-click on a rule or select **Create New Rule** to display the Edit SOD Rule page.
6. Enter the SOD Rule information in the top portion of the page. See "Edit SOD Rule Page" on page 361 for detailed descriptions of those fields.
7. Do one of the following: Select a role from the Add Role drop-down list below the Any of these roles table.
   a. Select a role from the Add Role drop-down list below the Any of these roles table.
   b. Select a role from the Add Role drop-down list below the conflict with any of these roles table.

   The drop-down list contains all of the roles defined for your organization. You can enter as many roles as are needed to build this rule.

   — OR —

   a. Select an application and use the Add Attribute or Add Permission buttons to build the First Entitlement Set.
   b. Select an application and use the Add Attribute or Add Permission buttons to build the Second Entitlement Set.

      For attributes select an attribute from the drop-down list and type a value.
      For permissions, type the name (target) and value (right).
      You can enter as many attributes and permissions as needed to build this rule.
8. Click **Done** to return to the Edit Policy page.
9. Repeat steps 5 thru 8 until all of the rules needed for this policy have been added or modified.
10. Click **Save** to save the policy and return to the Policies page.

How to Create or Edit an Activity Policy

Policies are created using the Edit Policy and Edit Activity Policy Rule pages. Use this procedure to create new policies.

Procedure

1. Click or mouse over the Define tab and select **Policies**.
2. **Optional**: Use the filtering options to limit the number of policies displayed in the table.
   You can filter by both policy name and policy type.
3. Select Activity Policy from the **Create new policy** drop-down list or click on an existing policy to display the Edit Policy page.
4. Enter the policy information. See "Edit Policy Page" on page 358 for detail description of the Edit Policy page.
5. Click on a rule or **Create New Rule** to display the Edit Activity Policy Rule page.
6. Enter the Activity Policy Rule information in the top portion of the page. See "Edit Activity Rule Page" on page 363 for detailed descriptions of those fields.
7. Create the filters necessary to identify the identity and activity types that should be considered when performing the policy scans for this violation.
   Use the Identity Filters and Activity Filters panels to add and combine filters for use in the policy. Apply qualifiers to filters to limit the values returned and then use grouping, AND\OR operations, and time periods to create the rules that make up the policy.

   Add a Filter:

   Create the filters that make up the rules.

   - **Field** — select an attribute value from the drop-down list.
   - **Search Type** — the qualifier to associate with the value, for example equals or like.
   - **Value** — the value of the field selected.
   - **Ignore Case** — specifies if case should be factored into the query.

   Filter(s):

   The Operations drop-down list enables you to specify AND/OR relationships between the filters in the list. Select multiple filters and group them to create sub-filters and use multiple layers of filter grouping to create complex rules.

   Click view/edit filter source to display an editable text version of the filter.

   See the online help or the IdentityIQ *User's Guide* for details on using the advanced filtering functions.
8. Click **Done** to save the new policy and return to the Edit Policies page.

How to Create or Edit an Advanced Policy

Policies are created using the Edit Policy and Edit Activity Policy Rule pages. Use this procedure to create new policies.

Procedure

1. Click or mouse over the Define tab and select **Policies**.
2. ***Optional***: Use the filtering options to limit the number of policies displayed in the table.
   You can filter by both policy name and policy type.
3. Select Advanced Policy from the **Create new policy** drop-down list or click on an existing policy to display the Edit Policy page.
4. Enter the policy information.

See "Edit Policy Page" on page 358 for detail description of the Edit Policy page.

5. Click **Create New Rule** or right-click on an existing rule to display the Edit Advanced Rule page.
6. Enter the Advanced Rule information in the top portion of the page. See "Edit Advanced Policy Rule Page" on page 365 for detailed descriptions of those fields.
7. Select a method by which to generate this rule:
   - **Match List** — define a list of entitlements to determine the rule.
     For attributes select an attribute from the drop-down list and type a value.
     For permissions, type the name (target) and value (right).
   - **Filter** — enter a custom XML database query to define user for this rule.
   - **Script** — enter a custom script to define the rule. Scripts are similar to rules, but the source is stored with the policy and can be edited from this page.
   - **Rule** — select an existing rule from the drop-down list.
   - **Population** — select a population from the list. Any identity that matches the criteria defined for the population displayed is in violation of this policy.
8. Click **Done** to save the new policy and return to the Edit Policies page.

Chapter 22: Work Items

The Work Items page provides a central location where you can view and manage work items that are assigned to you or to a workgroup of which you are a member. A work item is anything that requires an action before it is completed. Work items can be entire processes, such as access reviews, or any piece of a process, such as the approval of one entitlement for one user on one application.

Work Items on the Home Page

When a work item is created and you have a Notifications card on your Home page, the Notifications card displays the number of work items assigned to you. To access the work items, click the card or go to the navigation bar and click **My Work -> Work Items**.

To manage work items, refer to the following:

- "Work Item Administration" on page 371
- "Work Item Archive" on page 372

Work Item Administration

Note: To edit priorities in IdentityIQ, the “Allow priority editing on work items” setting must be selected on the Work Item tab IdentityIQ Configuration page located under the gear icon.

If a work item is created for a user who is no longer active in IdentityIQ, it is forwarded to the manager or supervisor for that user. If no manager is listed, the work item is assigned to the IdentityIQ administrator. Use escalation rules to determine the proper escalation path for orphaned work items. Escalation rules are created and set during the configuration and implementation of the product. Orphaned work items are discovered and identified during the Perform Maintenance task.

Use **Sort By** to customize the sort order of the work item list, or use the newest to oldest icon to flip the list.

Use **Filter** to limit the number of work items displayed, or search on a specific work item using the search field.

Click **View Archive** to see a list of completed work items.

Use the **Show...** drop-down list to select the work items you would like to see.

The Work Items page displays the following information:

Table 91—Work Item Page

Column Name	Description
Priority	Specifies the priority level to which the work item was designated. Use the drop-down list and edit the priority level. This edit is visible in the Work Items Manager and Inbox of the identity to whom the work item is assigned, as well the outbox of the person that assigned the work item.
Type	The type of work item.
Name	The name of the work item.
Created	The date the work item was assigned.
ID	Identification number assigned to the work item.

Table 91—Work Item Page

Column Name	Description
Owner	The name of the identity who has purview over the work item.
Requestor	The name of the user who assigned this work item to you.

Click the information icon to see the Details dialog containing work item and identity details, as well as any forwarding history associated to the work item.

Note: Work items can only be assigned if the assignee of the work item is a member of the same workgroup as the person who is assigning the work item.

Click the forward icon to open the **Forward Work Item** dialog.

The Manage Work Items table includes the following types of work items:

Work Item Archive

Use the Work Item Archive page to view completed work items. Only work items types that are configured in System Setup can be viewed on the Work Item Archive page. To access the system settings for Work Items, navigate to the IdentityIQ **Configuration > Work Items** tab under the gear icon.

Click the drop-down list to specify if your table displays all work items assigned to you and any groups to which you belong, only your own, personal work items or only the work items assigned to a selected workgroup.

To customize the information displayed in the Work Item Archive table, mouse over one of the header rows, click the drop-down arrow to reveal the sub-menu and select the desired columns from the Columns pop-out menu.

Click a line item launch the View Work Item page which displays detailed information about the work item.

Table 92—Work Item Archive Column Descriptions

Column Name	Description
ID	Identification number assigned to the work item.
Name	The name of the work item.
Type	The type of work item.
Requestor	The name of the user who assigned this work item to you.
Workgroup	Displays the workgroup to which this work item is assigned if applicable.
Owner	The name of the identity who has purview over the work item.
Completed By	The date the work item was completed
Created	The date the work item was assigned.
Modified	The date changes, if any, were made to the work item.
Archived	The date the work item was archived.
Priority	Specifies the priority level to which the work item was designated. Use the drop-down list and edit the priority level. This edit is visible in the Work Items Manager and Inbox of the identity to whom the work item is assigned, as well the outbox of the person that assigned the work item.

Table 92—Work Item Archive Column Descriptions

Column Name	Description
Access Request ID	Identification number designated for the Lifecycle Manager access request.

Chapter 23: IdentityIQ Console

The IdentityIQ Console is the command line utility for interfacing with IdentityIQ. This document lists the console commands and their descriptions.

Launching the Console

The IdentityIQ console requires the System Administrator capability.

By default, the console tries to authenticate with the default user/password `admin`/`admin`. If authentication fails, you are prompted for a user name and password. Specify the user name and password on the command line.

For example:
iiq `console -u amy.cox -p` IdentityIQ

the console prompts for user input if either the user or password are omitted.

Note: Authentication is disabled if there are no identities. This case is encountered during IdentityIQ setup, before init.xml is imported.

The IdentityIQ Console (iiq console) is launched by executing the iiq`.bat` file found in the *`Installation Directory`*`/WEB-INF/bin` directory. From a command prompt, launch the console with the command as shown for each operating system type:

Table 93—Console Launch Commands

Operating System	Command
Windows	iiq `console`
Unix	`./`iiq `console -j`

Note: The -j option turns on the JLine Java library for handling console input, enabling some ease-of-use functions such as command history recall. Command history recall is enabled in Windows without this library, so this parameter is not required in the Windows environment.

The > prompt character indicates that the console is running and ready to accept commands.

Viewing the List of Command

The **help** command displays a list of all commands available in the console along with a short description of each. At the command prompt, enter `help` or`?` to see this full list of available commands.

Table 94—List of Console Commands

Command	Description
?	Display command help
help	Display command help
echo	Display a line of text

Table 94—List of Console Commands

Command	Description
quit	Quit the shell (same as exit)
exit	Exit the shell (same as quit)
source	Execute a file of commands
properties	Display system properties
time	Show how much time a command takes to run
xtimes	Run a command x times
about	Show application configuration information
threads	Show active threads
logConfig	Reload log4j configuration
Objects	
dtd	Create dtd
summary	Summarize objects
classes	List available classes
list	List objects
count	Count objects
get	View an object
checkout	Checkout an object to a file
checkin	Checkin an object from a file
delete	Delete an object
rollback	Rollback to a previous version
rename	Rename an object
import	Import objects from a file
importManagedAttributes	Import managed attribute definitions from a CSV file
export	Export objects to a file
exportManagedAttributes	Export managed attribute definitions to a CSV file
exportJasper	Exports only the jasperReport xml contained in a JasperTemplate object.
Identities	
identities	Llist identities
snapshot	Create an identity snapshot
score	Refresh compliance scores
listLocks	List all class locks
breakLocks	Break all class locks
Tasks	

Table 94—List of Console Commands

Command	Description
tasks	Display scheduled tasks
run	Launch a background task
runTaskWithArguments	Launch a task synchronously with arguments
terminate	Terminate a background task
terminateOrphans	Detect and terminate orphaned tasks
restart	Restart a failed task if possible
Certifications	
certify	Generate an access certification report
cancelCertify	Cancel an access certification report
archiveCertification	Archive and delete an access certification report
decompressCertification	Decompress an access certification archive
Groups	
refreshFactories	Refresh group factories (but not groups)
refreshGroups	Refresh groups (but not factories)
showGroup	Show identities in a group
Workflow	
workflow	Start a generic workflow
validate	Validate workflow definition
workItem	Describe a work item
approve	Approve a work item
reject	Reject a work item
wftest	Run the workflow test harness
Tests	
rule	Run a rule
parse	Parse an XML file
warp	Parse an XML object and print the re-serialization
notify	Send an email
authenticate	Test authentication
simulateHistory	Simulate trend history
search	Run a simple query
textsearch	Run a full text search
certificationPhase	Transition a certification into a new phase
impact	Perform impact analysis

Table 94—List of Console Commands

Command	Description
event	Schedule an identity event
expire	Immediately expire a workitem that has an expiration configured. If the workitem is type Event it'll also push the event forward with the workflower
connectorDebug	Call one of the exposed connector methods using the specified application
encrypt	Encrypt a string.
sql	Execute a SQL statement
hql	Execute a search based on a Hibernate Query Language statement.
updateHql	Update the hql search.
date	Displays the current system date/time and its UTIME (universal time) value (Optional UTIME parameter causes the command to display the date/time corresponding to the provided UTIME value.)
shell	Escapes out to the command line and run the command specified.
meter	Toggles metering on and off; while metering is on, the console reports some timing statistics for each command executed. Meter information is displayed after the results of each command as it is executed.
compress	Compress the contents of a file to a string that can be included within an XML element.
uncompress	Return a compressed, Base64-encoded file to its uncompressed format.
clearEmailQueue	Remove any queued emails that have not been sent
provision	Evaluate a provisioning plan
lock	Lock an object
unlock	Break a lock on an object
showLock	Show lock details
clearCache	Clear the object cache
service	Service management
oconfig	Analyze ObjectConfigs

Command Syntax

The syntax for any console command that requires parameters can be determined by entering that command with no arguments.

```
> workflow
```

```
usage: workflow name [varfile]
```

Note: Command names are case sensitive and must be entered as shown in the command list. Parameters are not case sensitive.

Some commands take no arguments and execute if entered. This table contains a list of the commands that require no arguments.

Table 95—Console Commands That Do Not Require Arguments

Command	Action
? or **help**	Lists all available console commands
quit or **exit**	Exits the console shell
classes	Lists all classes
refreshGroups	Refresh group indexes (Optional group name or ID can be specified)
refreshFactories	Refresh set of GroupDefinitions for a GroupFactory (Optional factory name or ID can be specified)
logConfig	Reloads log4j configuration from log4j2.properties file
summary	Lists all classes and the count of objects of that class in the system
properties	Displays Java properties of the server where IdentityIQ is installed
about	Displays application configuration information
threads	Shows a list of active threads
tasks	Writes a list of all currently scheduled tasks, in a columnar layout, to the console (stdout)
identities	Writes the Name, Manager, Roles, and Links for each Identity in the system to the console (stdout)
date	Displays the current system date/time and its UTIME (universal time) value (Optional UTIME parameter causes the command to display the date/time corresponding to the provided UTIME value.)
status	Reports current running status of the task and request schedulers
meter	Toggles metering on and off; while metering is on, the console reports some timing statistics for each command executed. Meter information is displayed after the results of each command as it is executed.
clearEmailQueue	Deletes all queued but unsent email messages
clearCache	Clears the IdentityIQ object cache

Syntax for Redirecting Command Output

Most of the commands report data or error messages to the console or standard out (stdout) for the system. The output for any command can be redirected to a file by specifying `> filename` at the end of the command.

This example redirects the output from the **get** command to a file:

```
> get identity Adam.Kennedy > c:\output\AdamKennedyID.xml
```

Console Commands

In this document, the list of console commands is subdivided based on how frequently they are likely to be used in a production environment.

- **Commonly Used Commands** — commands that system administrators should know well and use frequently.
- **Less Commonly Used Commands** — commands that might need to be run periodically, such as when working with SailPoint Support to resolve an issue, but are not used regularly.
- **Seldom Used Commands** — developer commands that are rarely useful in a production environment.

Commonly Used Commands

This section lists and documents the syntax and actions of the most commonly used console commands.

Help and ?

These two commands list all the available console commands.

Syntax	`?` `help`

Examples	`> ?`
	
	`> help`

Result	Lists all commands available in the console

Exit and Quit

These two commands exit the console shell, returning the user to the operating system command prompt.

Syntax	`exit` `quit`

Examples	`> exit`
	
	`> quit`

Result	Exits console shell and returns user to the operating system command prompt

Source

The **source** command runs commands from a script file. The commands on each line in the file are executed by the console sequentially.

Syntax	`source` *`filename`*

Examples	`source c:\data\cmdfile.txt`
Result	Runs the console commands in the `c:\data\cmdfile.txt` file sequentially

List

The **list** command lists all objects of the specified class, constrained by any specified filter. If this command is specified without arguments, the command syntax is displayed, followed by a list of all available classes whose objects can be listed. This is helpful in locating objects within the system and in identifying object names to use as parameters on other commands.

Syntax	`list classname [filter]`
	`filter: xxx - names beginning with xxx`
	`xxx* - names beginning with xxx`
	`*xxx - names ending with xxx`
	`*xxx* - names containing xxx`

Examples	`> list application ent*`
Result	Lists all application objects whose names begin with ent

Get

The **get** command displays the XML representation of the named object.

Syntax	`get classname <objectname`re `or ID>`
Examples	`> get identity Adam.Kennedy`
Result	Displays the Adam.Kennedy Identity in XML format

Note: This command only displays the object to the console (stdout), it does not export the object. The output can be redirected to a file if the user has write access to the server's file system.

```
> get identity Adam.Kennedy > c:\output\AdamKennedyID.xml
```

Other alternatives for getting the XML representation of an object into a text file include:

- copying and pasting contents of this command's stdout into a text file
- retrieving the object's XML from the IdentityIQ Debug pages
- using the **checkout** command (described next) to write the XML representation of an object to a text file

Checkout

The **checkout** command writes a copy of the XML representation of the requested object to the specified filename. The file can be used for review or for moving objects from one environment to another, for example, from the user acceptance testing environment to production. Organizations doing custom development on rules, workflows, etc. might use **checkout** to extract any of these objects to a file for modification.

Syntax	`checkout class name <objectname or ID> file [-clean [=id,created…]]`
Examples	`> checkout rule "Cert Signoff Approver" certrule.xml`
Result	Writes a copy of the Cert Signoff Approver rule's XML representation to the file `certrule.xml`

The **-clean** option can be used to remove all values that do not transfer between IdentityIQ instances, such as created and modified dates as well as globally unique ID values (GUIDs). Specifying the **-clean** option with no qualifiers cleans the id, created, modified, and lastRefresh attributes. The **-clean** option can also be used to explicitly clear specific fields by name. The fields to clear must be listed in a comma separated list.

Checkin

The **checkin** command reads a file containing an object's XML representation and saves the object into the database. If the object is a workItem, the command invokes the workflower to process the workItem. If the object is a bundle (role) and the approve parameter is specified, a role approval workflow is launched. For all other object types, and for bundles that are submitted without the approve parameter, the object is saved into the database.

Note: The command's syntax parsing allows the approve parameter to be specified for any object but it only impacts the processing on Bundle objects.

Syntax	`checkin filename [approve]`

Examples	`> checkin newRole.xml approve`
	
	`> checkin bobSmithID.xml`

Result	First example saves Identity Bob Smith, as represented by the XML in bobSmithID.xml, into to the database; overwrites existing or adds new record Second example launches an approval workflow for the bundle object represented by the XML in newRole.xml

Note: If an Import file is specified as the input file for this command, only the first object in the file is checked in; the rest are ignored and a warning message is displayed to the console (stdout).

Delete

Note: This action cannot be undone and should be used with extreme caution and only in rare circumstances.

The **delete** command deletes the named object and removes all of its owned, or subordinate, objects. In a production environment, this is not recommended unless specifically directed by SailPoint Support.

Syntax	`delete classname <objectname or ID>`
Examples	`> delete identity bob.smith`

Result	Removes Identity Bob Smith and all of his associated objects from the system

Note: Wildcards can be used on the <*object name* or *ID*> argument:
— * - all objects of the specified class (use with extreme caution!)
— xxx - all objects whose name or ID contains xxx

Import

The **import** command imports objects into IdentityIQ from an XML file. This command can be used on a file that contains a Jasper report, a SailPoint import file, or an object of one of the standard object classes. The file contents are evaluated and processed based on the first tag in the file:

- *JasperReport*: Jasper report
- SailPoint: SailPoint import object; can contain multiple regular objects in one file as well as an ImportAction tag that directs how the contents of the file are processed, for example, merge, include, execute, logConfig.
- Anything else: assumed to be a single regular object

Syntax	`import [-noids] filename`

Examples	`> import init.xml`
	
	`> import -noids init.xml`

Result	The first example Imports the contents of the file `init.xml` into the IdentityIQ database. In the second example, all ID attributes are removed before parsing occurs. This action is a normal part of the initialization process for IdentityIQ.

Syntax	`import [-noids][-noroleevents] filename`

Examples	`> import -noids bundles.xml`
	
	`> import -noids -noroleevents bundles.xml`

Result	The first example allows user to import the events. It removes all ID attributes before parsing. The second example disables generation of role change events for role propagation. Select 'Allow Role Propagation' option from the Global Settings -> IdentityIQ Configuration -> Roles option in UI.

This is one of the most commonly used commands. Installations who manage their workflows and rules in an external source code control system, for example, use this command to bring changes to those objects into IdentityIQ once they have been modified in their external XML representations.

Export

The **export** command writes all objects of a given class to a specified filename. This is commonly used in gathering objects from IdentityIQ to deliver to SailPoint Support as resources in resolving tickets. It is also used for moving sets of objects between environments and for managing objects outside of IdentityIQ, such as storing workflows and rules in a source code control system.

More than one class can be exported at a time to the same file by specifying all the desired class names as arguments to the command. If the **export** command is specified without any class names, all objects of all classes are exported to the specified filename.

Syntax	`export [-clean[=id,created...]] filename [classname classname …]`
Examples	`> export -clean workflows.xml workflow` `> export IdLink.xml identity link`
Result	The first example exports the entire set of workflow objects from IdentityIQ to the file `workflows.xml`, removing values from the id, created, modified, and lastRefresh attributes. The second example exports all identities and links to a single file.

ListLocks

The **listLocks** command lists all locks held on any objects of the named class. At this time, Identity is the only class for which this command operates.

BreakLocks

Note: This command should be used with caution. Locks are useful in maintaining data integrity, and breaking them at the wrong time can potentially permit conflicting updates that can result in data corruption.

Note: The unlock command can be used to break a single lock whereas this command breaks all locks held on any object in the specified class.

If a process is holding a lock but is unable to perform the required action, the lock can cause problems in other processes' performance as well. The **breakLocks** command can be used to release locks forcibly. At this time, Identity is the only class for which this command functions.

Syntax	`breakLocks classname`
Examples	`> breakLocks identity`
Result	Releases all locks held on any identity object in the system and reports to the console the identity name, lock holder, and UTIME value for the lock date/time and the lock expiration date/time.

Rule

The **rule** command runs a rule defined in the system. The rule to run is specified as a command parameter. If any input variables must be passed to the rule, they must be entered in a variable file, specified as an XML Map. The file name is then also passed as a parameter to the command.

This command can be used for testing or executing existing system rules. It can also be used to run any beanshell code snippet against the IdentityIQ database. The code is created as a rule and loaded into the system and then executed from the console. Support uses rules like this for data cleanup.

Syntax	`rule <rulename or ID> [varfile]`
Examples	`> rule "Check Password Policy" c:\data\pwdParams.xml`
Result	Runs the Check Password Policy rule, passing its input variables through the file `c:\data\pwdParams.xml`

Parse

The **parse** command validates an XML file. If it is in valid form and its tags match the IdentityIQ DTD, it runs successfully and no information is printed to the console (`stdout`). If errors are encountered, a runtimeException is printed to the console describing the error.

Syntax	`parse filename`
Examples	`> parse c:\data\newWorkflow.xml`
Result	Validates the XML in the file `c:\data\newWorkflow.xml` and reports any errors to the console

Less Commonly Used Commands

These commands are not frequently used in a production environment. However, it is helpful to understand and be able to use them when the need arises.

DTD

The DTD command writes the IdentityIQ DTD (Document Type Definition) to the specified file.

Syntax	`dtd filename`
Examples	`> dtd c:\DTD\`IdentityIQ`.dtd`
Result	Writes the IdentityIQ DTD to the file `c:\DTD\`IdentityIQ`.dtd`

Classes

The **classes** command lists all classes accessible from the console. These are frequently used as parameters to other commands so this list can be helpful in entering correct arguments on those commands.

Syntax	`classes`
Examples	`> classes`
Result	Lists class names for all classes accessible to the console

Count

The **count** command returns a count of the objects of the specified class.

Syntax	`count` *`classname`*
Examples	`> count identity`
Result	Displays the count of Identity objects in the system

ImportManagedAttributes

The **importManagedAttributes** command is used to set managed attribute values, including localized descriptions, through a CSV file import. This can be used to update existing managedAttributes or to create new ones.

The filename can be specified with an absolute path or can be specified relative to the current working directory. These are the specific requirements for the import file contents:

- The first line in the file must be a comment line (starting with #) that contains the column names for the data records. Column names must be specified in a comma-separated format. All column names must match managedAttribute standard or extended attributes or specify a locale/supported language.
- Subsequent comment lines can be used to specify default values for attributes that are not contained in the data records. For example, if the whole file relates to a single application, the application name could be set as a default through a single comment line.
- Blank lines are permitted in the file, and are ignored, but cannot precede the first comment line.
- The data records must consist of comma-separated data values.

  Note: Only types Entitlement and Permission are valid. Group managedAttributes are stored in IdentityIQ as a subcategory of Entitlement type managedAttributes.
- Required attributes are type, application, attribute, and value. If type is not specified in the file, type Entitlement is assumed. The others three properties must be specified in the file. Type, application, and attribute can be specified through the data columns or with a single default value in extra comment lines. The value attribute must be in the data columns and cannot have a default value.
- The data values in the columns named to match supported languages should contain the description to use for that locale.

Example File Contents:

```
# value, displayName, En_US, owner=Jeff.Wilson
# application=AD
# attribute=MemberOf
# type=Entitlement
"CN=administrators,CN=Roles,DC=iiq,DC=com", Admins, "Administrators
group","CN=VPN,CN=Roles,DC=iiq,DC=com", VPN, "Remote Workers", Bob.Smith
```

Note: The test option causes the command to parse and validate the file without saving changes to the database.

Syntax	`importManagedAttributes` *`filename`* `[test]`
Examples	`> importManagedAttributes "c:\data\managedattributes.csv"`
Result	Imports managed attribute data from the file `c:\data\managedattrbutes.csv`, updating existing attributes or creating new ones from the data

ExportManagedAttributes

The **exportManagedAttributes** command exports either object properties or descriptions for managedAttributes to a CSV file. This is used to make mass changes to the managed attributes definitions or for collecting all the managed attributes in one file to review as a group.

Syntax	`exportManagedAttributes` *`filename`* `[`*`application`*`] [`*`language`*`]`
Examples	`> exportManagedAttributes "c:\data\AdamManAttrDesc.csv" ADAM en_US`
Result	Exports the managed attribute description on ADAM application to the file `c:\data\AdamManAttrDesc.csv`.

Application and language are both optional arguments and can be specified in either order. At most one application and one language can be specified at a time. If no application name is specified, managed attributes for all applications are exported. If a language is specified, only the core identifying properties of the managed attributes (type, application, attribute, value) and the descriptions for the specified locale are exported. If a language is not specified, all other object properties except descriptions are exported.

The file format generated by this command can be used in the **importManagedAttribute** command, so this command can be used to write values to a file for editing and reimporting. When an application name is specified on the export command, the application is not shown in the data rows but is specified as a default in the file header comments, as described and illustrated in the command details.

Run

The **run** command starts execution of a task that requires and accepts no arguments. Three optional parameters can be specified for this command: trace, profile, and sync.

- **Trace** — writes to the console (stdout) a trace of what happens as the task is run, depending on how the task's tracing code is written.
- **Profile** —displays the timing of certain phases of the task, the details displayed depend on the task's profile code.
- **Sync** — runs the task in synchronous execution mode, as opposed to scheduling it to run in background. If sync is specified, the control returns to the console only after the task has completed and any error messages are written to the console. If sync is not specified, the task is launched in the background and the results of the task are viewable in the taskResults object and are accessible from the console or from the user interface under **Setup** -> **Tasks** -> **Task Results**.

Syntax	`run` *`taskname`* `[trace] [profile] [sync]`

Examples	`> run "Refresh Risk Scores"`
Result	Runs the Refresh Risk Scores task in background

RunTaskWithArguments

The **runTaskWithArguments** command starts execution of a task that requires arguments. These tasks are always run in synchronous execution mode. This can only be used for tasks that accept arguments of simple data types; specifying an object as an argument is not possible here.

Syntax	`runTaskWithArguments` *`taskname`* `[arg1=val1,arg2=val2,…]`
Examples	`> runTaskWithArguments "Identity Refresh" refreshLinks=True,promoteAttributes=False`
Result	Runs the Identity Refresh task, refreshing Links for the Identities

Restart

The **restart** command restarts execution of a task that failed.

Syntax	`restart` *`taskResultName`*
Examples	`> run "Refresh Risk Scores"`
Result	Restarts the Refresh Risk Scores task in background if possible

RefreshFactories

The **refreshFactories** command can be specified with no arguments to refresh all group factories or with a specific GroupFactory name. Refreshing the group factory means identifying the group values that define each of the groups (or GroupDefinition objects). It does not refresh the list of Identities that make up each group or the statistics gathered for each group - just the list of groups themselves.

Syntax	`refreshFactories [<`*`factorname or ID`*`>]`
Examples	`> refreshFactories`
Result	Refreshes the GroupDefinition list associated with each GroupFactory in the system

RefreshGroups

The **refreshGroups** command refreshes the group indexes for all groups or for the specified group named as a command argument. The group indexes are collections of statistics for identities that are part of the group. The statistics include number of members, number of certifications due for certification owners in the group, number of certifications owned and completed on time by members of the group, and risk score information for members of the group. RefreshGroups only applies to GroupDefinitions that are indexed (attribute indexed="True").

Syntax	refreshGroups [*groupname* or *ID*]
Examples	> refreshGroups
Result	Refreshes index information for all indexed groups

ShowGroup

The **showGroup** command shows the membership (Identities) of the group named as an argument to the command.

Syntax	showGroup <*groupname* or *ID*>
Examples	> showGroup Finance
Result	Lists all Identities who are members of the Finance group.

Workflow

The **workflow** command launches the workflow specified as a command parameter. Input variables must be entered in a variable file to get passed to the workflow. A variable file is specified as an XML Map. The file name is then also passed as a parameter to the command. When the workflow is successfully launched, the XML for the workflowCase is printed to the console (stdout).

Syntax	workflow <*workflowname* or *ID*> [*varfile*]
Examples	> workflow "LCM Provisioning" c:\data\provFile.xml
Result	Runs the LCM Provisioning workflow, passing its input variables through the file c:\data\provFile.xml

The varfile should contain an attributes map like the example shown below. This example map passes an identity name and a provisioning plan object to the workflow.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Attributes PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<Attributes>
  <Map>
    <entry key="identityName" value="Adam.Kennedy"/>
    <entry key="plan">
      <value>
        <ProvisioningPlan>
          <AccountRequest application="IIQ" nativeIdentity="Adam.Kennedy" op="Modify">
            <AttributeRequest name="assignedRoles" op="Add" value="PRISM User">
              <Attributes>
                <Map>
                  <entry key="comments" value="req A"/>
                </Map>
              </Attributes>
            </AttributeRequest>
          </AccountRequest>
          <AccountRequest application="IIQ" nativeIdentity="ABC_12345" op="Modify">
```

```
            <AttributeRequest name="assignedRoles" op="Add" value="Test Role B2">
              <Attributes>
                <Map>
                  <entry key="comments" value="req B"/>
                </Map>
              </Attributes>
            </AttributeRequest>
          </AccountRequest>
        </ProvisioningPlan>
      </value>
    </entry>
   </Map>
</Attributes>
```

Validate

The **validate** command can validate a workflow or a rule. Input variables must be entered in the variable file to be passed to a workflow or rule. The variable file is specified as an XML map and the file name is passed as a parameter to the command. Validation errors are printed to the console (stdout).

Syntax	`validate <`*`rule`* `or` *`workflow`* `name or` *`ID`*`> [`*`varfile`*`]`
Examples	`> validate "LCM Provisioning" c:\data\provFile.xml`
Result	Validates the LCM Provisioning workflow, passes the input variables through `c:\data\provFile.xml`, and displays any validation errors

See "Workflow" on page 389 for an example of the varfile format.

Wftest

The **wftest** command is used to one or more workflows. The *WorkflowTestSuite* can be the name of a WorkflowTestSuite object or a file containing one.

Syntax	`wftest` *`WorkFlowTestSuite`* `name |` *`filename`*
Examples	`> wftest c:\test\workflowTest.xml name | c:\test\workflowTestOut.xml`
Result	Tests the workflows and sends the outcome to the `workflowTestOut` file.

SQL

The **sql** command executes a SQL statement. It can execute SQL specified inline with the command or it can read the SQL from a file. Only one SQL statement can be executed at a time. The output can be printed to the console (stdout) or redirected to a file. Select, update, and delete SQL statements can be executed. Update and delete actions cannot be undone.

Syntax	`sql` *`sqlStatement`* `| -f` *`inputFileName`*
Examples	`> sql "select * from sptr_identity" > c:\data\Identities.dat` `> sql -f c:\sql\SelectIdentities.sql`

Result	The first example executes the specified select statement and writes the results to `c:\data\Identities.dat`. The second example reads the SQL from `c:\sql\SelectIdentities.sql`, prints the SQL to the console (stdout), and displays the query results to the console (stdout).

Provision

The **provision** command processes the specified provisioning plan for the specified identity but does not save the information. This is used to test a connector or to test a provisioning plan. Errors are reported to the console. If the provisioning action would succeed, nothing is reported to the console.

Syntax	`Provision <identityname or ID> provisioningPlanFfilename`
Examples	`> provision Adam.Kennedy c:\data\provFile.xml`
Result	Tests the provisioning plan contained in `c:\data\provFile.xml` against Identity Adam.Kennedy and reports any exceptions to the console

The provisioning plan file should contain a provisioning plan in XML format. For example:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE ProvisioningPlan PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<ProvisioningPlan>
   <AccountRequest application="IIQ" nativeIdentity="ABC_12345" op="Modify">
     <AttributeRequest name="assignedRoles" op="Add" value="Test Role B1">
       <Attributes>
          <Map>
           <entry key="comments" value="req A"/>
         </Map>
       </Attributes>
     </AttributeRequest>
   </AccountRequest>
   <AccountRequest application="IIQ" nativeIdentity="ABC_12345" op="Modify">
     <AttributeRequest name="assignedRoles" op="Add" value="Test Role B2">
       <Attributes>
         <Map>
           <entry key="comments" value="req B"/>
         </Map>
       </Attributes>
     </AttributeRequest>
   </AccountRequest>
</ProvisioningPlan>
```

Lock

The **lock** command obtains a persistence lock on an object. The object's class and ID or name must be specified. By default, the lock is issued to the username Console, but a different username can be specified in the command's lockName parameter. The lock automatically expires after 5 minutes.

Syntax	`lock classname <objectID or name> [lockName]`

Examples	`> lock identity John.Smith`
Result	Obtains a persistence lock for Console on the identity record for John.Smith

Note: Identity objects are the only objects that can be locked. Attempts to specify a different object type in this command results in a syntax error exception.

Unlock

The **unlock** command releases the lock on an object. The object's class and ID or name must be specified. If the object is not locked, the message “Object is not locked” is displayed. If it is locked, the lock is released and the message “Lock has been broken” is displayed.

Syntax	`unlock` *`classname <objectID or name>`*
Examples	`> unlock identity John.Smith`
Result	Breaks the lock on the identity record for John.Smith

ShowLock

The **showLock** command lists the lock owner, locked date/time, and lock expiration date/time for a locked object. The object's class and ID or name must be specified to view its lock information. The message Object is not locked is displayed if the object is not currently locked. If the lock has expired, the lock information is shown but is prefaced with the message Lock has expired.

Syntax	`showLock` *`classname <objectID or name>`*
Examples	`> showLock identity John.Smith`
Result	Displays lock information (owner, date/time, expiration date/time) on the identity record for John.Smith or displays Object is not locked

Oconfig

The **oconfig** command list all extended attributes defined for each class that supports extended attributes. The list indicates the extended attribute numbers and corresponding attribute names on each class. Identity extended attributes which link to other Identities are stored separately in extended identity attribute fields, so those are listed in a separate Extended Identity Attributes sub-list under the Identity ObjectConfig. The objectConfig detail displays No attributes defined if no extended attributes are defined for a given class. An Object not found message is displayed if no objectConfig exists for the class.

Syntax	`oconfig`
Examples	`> oconfig`

Result	Displays each objectConfig and its extended attributes, numbered according to the extended attribute number that corresponds to each
	`ObjectConfig: Identity`
	`  1 region`
	`  2 Department`
	`  3 location`
	`  4 empId`
	`  5 jobtitle`
	`  Extended Identity Attributes:`
	`    1 regionOwner`
	`    2 locationOwner`
	`ObjectConfig: Link`
	`  1 inactive`
	`  2 service`
	`  3 privileged`
	`  4 lastLogin`
	`ObjectConfig: Application`
	`  1 DeployDate`
	`ObjectConfig: Bundle`
	`  No attributes defined`
	`ObjectConfig: ManagedAttribute`
	`  1 authorization`
	`  2 email`
	`  3 rank`
	`ObjectConfig: CertificationItem`
	`  Object not found`

TextSearch

The textsearch command enables command-line execution of full text searches as they are done through the LCM full text search. The class name must be either ManagedAttribute or Bundle, since those are the only indexed classes. This command searches for the specified string in the fullTextIndex created for the specified class and returns a map representation of the objects in which the string is found. If a filter attribute and value are specified, the search is further constrained to entries that correspond to that attribute name-value pair. The filter is always treated as an equals operation. The filterName must be an attribute that is indexed in the FullTextIndex object for the specified class.

Syntax	`textsearch` *`classname string`* [*`filterName filterValue`*]
Examples	`> textsearch Bundle manager type business`
Result	Returns a map of data values for each Bundle (role) of type=business that contains the string manager in any analyzed field. Analyzed fields in the Bundle FullTextIndex marked as analyzed=true.

Seldom Used Commands

These commands are used by developers for testing and are rarely used in a production environment.

Properties

The **properties** command displays system properties.

Syntax	`properties`
Examples	`> properties`
Result	Displays Java properties of the server on which IdentityIQ is installed

Time

The **time** command reports the duration of another command.

Syntax	`time` *`command`*
Examples	`> time run "refresh risk scores"`
Result	Initiates the **run** command and then indicates how much time it took to run. Most useful for long-running commands

Xtimes

The **xtimes** command repeats a single command as many times as specified in the first argument. This command is used for performance testing purposes. Running a command numerous time provides a more accurate indication of how long a process takes than running it once.

Syntax	`xtimes` *`x command`*
Examples	`> xtimes 3 run "refresh risk scores"`
Result	Runs the refresh risk scores task three times in a row

Note: This command can be combined with the time command to report timing statistics on the performance test. By specifying this command first (for example, xtimes 20 time run *taskname*), the time taken for each command run is reported. By specifying the time command first (for example, time xtimes 20 run *taskname***), the total time for all of the sequential runs is reported.**

About

The **about** command displays IdentityIQ's application configuration information.

Syntax	`about`
Examples	`> about`
Result	Lists application configuration specifics for the IdentityIQ instance (version, database, host, memory, etc.)

Threads

The **threads** command displays all active threads in the instance.

Syntax	`threads`
Examples	`> threads`
Result	Lists all active threads

LogConfig

The **logConfig** command reloads the log4j configuration into the instance.

Syntax	`logConfig`
Examples	`> logConfig`
Result	Reloads the log4j configuration from the `log4j2.properties` file

Summary

The **summary** command lists all classes and the count of objects of each class. Changes in these counts for some objects (for example, auditConfig) can indicate potential problems or areas of concern.

Syntax	`summary`
Examples	`> summary`
Result	Lists class name and count of objects for each class in the system

Rollback

The **rollback** command can undo a change to a role by restoring it from its BundleArchive object. BundleArchive objects are created when role archiving is enabled for IdentityIQ. Role archiving tracks changes made to a role by storing the pre-modification state in a BundleArchive object when the Bundle object is updated. This command only applies to the BundleArchive class.

Syntax	`rollback` *`classname`* `<`*`objectname`* `or` *`id`*`>`
Examples	`> rollback BundleArchive "Contractor-IT"`
Result	Restores the Contractor-IT role to the pre-modification state stored in its BundleArchive object

Rename

The **rename** command changes the name of an object from its existing name to the value specified by the *newname* parameter.

Syntax	`rename classname <objectname or ID> newname`
Examples	`> rename application ADAM ADAM-Production`
Result	Changes the name of the ADAM application to ADAM-Production

Note: The object can be found using its old Name or its ID value, but in either case, the *newname* value is used to update the Name attribute for the object.

ExportJasper

The **exportJasper** command creates a JasperReport XML file from a JasperTemplate object in IdentityIQ. Jasper Report is a third party user interface for report writing. JasperReport XML is not compatible with IdentityIQ's XML so the JasperReport XML is wrapped in a JasperTemplate object when saved in IdentityIQ. The JasperTemplate must be exported to create a file that can be used directly with the Jasper user interface before it can be reformatted.

Syntax	`exportJasper filename <JasperTemplateName or ID>`
Examples	`> exportJasper c:\data\AggResRpt.xml AggregationResults`
Result	Exports the Jasper XML from the AggregationResults JasperTemplate object into the file `c:\data\AggResRpt.xml`

Note: The import command can be used to re-import a JasperReport object into the database. The import wraps the XML in a JasperTemplate.

Identities

The **identities** command lists the Name, Manager, Roles, and Links for each identity in the system. By default, this information prints to the console (stdout) and can be difficult to read due to screen wrapping. If the output is redirected to a file, it is printed in the file in an easy-to-read style.

Syntax	`identities`
Examples	`> identities` `> identities > identities.txt`
Result	The first example writes the Name, Manager, Roles, and Links for each identity in the system to the console (stdout). The second example redirects that information to the file `identities.txt`.

Snapshot

The **snapshot** command takes a snapshot of the named identity as it exists at that moment and archives it in the database as an IdentitySnapshot object. This object provides a historical record of the state of Identity objects at various points in time. Automatic snapshotting can be enabled and configured to create IdentitySnapshot objects

at specified intervals or based on system activities (weekly, on aggregation change, etc.). The configuration of this feature can negatively impact system performance.

Syntax	`snapshot <identityname or ID>`
Examples	`> snapshot Alan.Bradley`
Result	Creates an IdentitySnapshot object for the identity Alan.Bradley, capturing his Identity Attributes, Roles (Bundles), Entitlements Outside Roles, Links, and Scorecard information at that moment in time

Score

The **score** command refreshes the identity score for the named identity and updates that score in the database. Score updates are more commonly executed through the IdentityIQ user interface.

Syntax	`score <identityname or ID>`
Examples	`> score Alan.Bradley`
Result	Recalculates the risk scores for Alan.Bradley and updates his Scorecard with the new risk scores

Tasks

The **tasks** command lists the Name, State, Next Execution, and Cron Strings for all currently scheduled tasks in the system.

Syntax	`tasks`
Examples	`> tasks`
Result	All currently scheduled tasks are written to the console (stdout)

TerminateOrphans

The **terminateOrphans** command sets the completion status of any open taskResult objects to Terminated. While tasks are running, their taskResults should be in a pending state, but occasionally task results can become orphaned and remain in this non-completed state when the task has finished (or has otherwise been terminated). This command can be used to clean up those orphaned taskResults but it must only be executed when there are no tasks running on the application server or the taskResults for actively running tasks are terminated along with any orphaned results.

This command requires no arguments for execution but an artificial argument please has been added to prevent accidentally running this command.

Syntax	`terminateOrphans please`
Examples	`> terminateOrphans please`

Result	Sets all open taskResults for the application server to the Terminated status

Certify

The **certify** command creates a manager or application certification. The certification is generated using the installation's default settings/parameters. This command is primarily used for testing purposes.

Syntax	`certify [managerName | application]`
Examples	`> certify Catherine.Simmons`
Result	Generates a manager certification for manager Catherine Simmons

Note: The command syntax help indicates that this command can generate an application owner certification when an application is specified as a command argument, but this feature has not been updated as the certification components of the product have changed over time. As a result, the application argument for this command is not currently usable.

CancelCertify

Note: This command is not recommended. Use the delete command to remove certification objects.

The **cancelCertify** command can be used to delete a certification object from the system.

Syntax	`cancelCertify <certificationName or ID>`
Examples	`> cancelCertify "Manager Access Review for William Moore"`
Result	Delete the named certification (the command fails if more than one certification object with the same name exists)

ArchiveCertification

The **archiveCertification** command archives the specified certification (creates a certificationArchive object) and deletes it as an active certification.

Syntax	`archiveCertification <certificationName or ID>`
Examples	`> archiveCertification "Manager Access Review for William Moore"`
Result	Creates a certificationArchive object and delete the certification from the system

DecompressCertification

The **decompressCertification** command retrieves the named certificationArchive object and prints it to the console (stdout) in the Certification object's XML format.

Syntax	`decompressCertification <certificationArchiveName or ID>`

Examples	`> decompressCertification "Manager Access Review for William Moore"`
Result	Prints the named certification archive to the console (stdout) in certification XML format

WorkItem

The **workItem** command displays certain details (Owner, Create Date, Expiration Date) for the specified workItem.

Note: This command requires the workItem ID or name value as an input parameter. The workItem ID value (a long hexadecimal number) is obtained using the IdentityIQ console's list workItem command. The workItem Name is not the descriptive name for the workitem, it is a numeric value assigned when the workItem is created. The value is found in the XML representation of each workItem through the Debug pages.

Syntax	`workItem <workItemID or Name>`
Examples	`> workItem 40288f0132b155ad0132b58a4e3f018e`
Result	Displays the Owner, Created Date, and Expiration Date for the specified workItem

Approve

The **approve** command sets the specified workItem to a Finished state (indicating it was approved), adds any specified completion comments to the workItem, and submits the workItem to the workflower to move it to the next appropriate stage.

Note: This command requires the workItem ID or name value as an input parameter. You can obtain the workItem ID value (a long hexadecimal number) using the IdentityIQ console list workItem command. The workItem Name is not the descriptive name for the workitem, it is a numeric value assigned when the workItem is created. The value is found in the XML representation of each workItem through the Debug pages.

Syntax	`approve <workItemID or Name> [comments]`
Examples	`> approve 40288f0132b155ad0132b58a4e3f018e "Access approved"`
Result	Marks the specified workItem as approved, adds the comment "Access approved" to the workItem's completion comments, and submits the workItem for evaluation of the next appropriate step (another approval, provisioning, etc.)

Reject

The **reject** command sets the specified workItem to a Rejected state, adds any specified completion comments to the workItem, and submits the workItem to the workflower to move it to the next appropriate stage.

Note: This command requires the workItem ID or name value as an input parameter. The workItem ID value (a long hexadecimal number) is obtained using the IdentityIQ console's list workItem command. The workItem Name is not the descriptive name for the workitem, it is a numeric value assigned when the workItem is created. The value is found in the XML representation of each workItem through the Debug pages.

Syntax	`reject <workItemID or name> [comments]`
Examples	`> reject 40288f0132b155ad0132b58a4e3f018e "Access conflicts with AP data entry entitlement"`
Result	Marks the specified workItem as rejected, adds the comment "Access conflicts with AP data entry entitlement" to the workItem's completion comments, and submits the workItem for evaluation of the next appropriate step (another approval, etc.)

Warp

The **warp** command parses an XML file to create an object and then displays the object's XML representation in the console (stdout). If it is not in valid form or its tags do not match the IdentityIQ DTD, a runtimeException is printed to the console describing the error.

Syntax	`warp filename`
Examples	`> warp c:\data\newWorkflow.xml`
Result	Parses the XML in the file `c:\data\newWorkflow.xml` and displays the XML representation of the object in the console, or reports any errors to the console

Notify

The **notify** command sends an email message to the specified identity using the email template specified. This command does not accept any other parameters that can be passed to the template, so it can only be used for templates whose messages do not rely on variable substitutions to build the content. This command is most often used for testing purposes.

Note: The toAddress argument can contain an identity name or ID or an email address. If it contains an identity name or ID, the email address is retrieved from the identity record.

Syntax	`notify <emailTemplateName or ID> toAddress`
Examples	`> notify Certification Alan.Bradley`
Result	Sends an email to Alan.Bradley's email address using the Certification email template

Authenticate

The **authenticate** command authenticates a username and password against the pass-through authentication source or the internal IdentityIQ records. No results are returned if the values are authenticated. If the password is incorrect or the user name cannot be found, an error message is displayed in the console (stdout).

Syntax	`authenticate username password`
Examples	`> authenticate Alan.Bradley s53n659#@5a!`

Result	Authenticates username Alan.Bradley and the provided password against the authentication source (pass-through or internal)

SimulateHistory

The **simulateHistory** command is used to generate a fake, randomly-generated group index or identity score history for one or more groups or identities. Used for generating test data in a development environment.

Syntax	`simulateHistory Identity|Group <`*`groupName` or `ID`*`>|<`*`identityName` or `ID`*`>|all`
Examples	`> simulateHistory Identity all` `> simulateHistory Group Finance`
Result	First example generates fake risk scorecards for all identities in the system Second example generates fake groupIndex information for the Finance group

Search

The **search** command looks up an object based on specified criteria, similar to a simplified SQL/HQL interface. A single class name is specified with a list of the attributes to display from that class. Following the `where` keyword, search filters can be specified in name value sets. All filter values are used in a like comparison. The record is returned if the record's field value contains the specified value string.

Syntax	`search `*`className`* `[`*`attributeName`*`…] where [`*`filter`*`…]` `filter: `*`attributeName value`*
Examples	`> search identity name manager.name region where name kat`
Result	Returns the name, manager's name, and region for all identities whose name contains the string kat. For example, records for Katherine.Jones, John.Kato, and Tammy.Erkatz are returned by this search

CertificationPhase

The **certificationPhase** command transitions the specified certification to the specified phase. This command fails if the certification is on or past the requested phase.

Note: The certification is advanced to the next enabled phase after the requested phase if the specified phase is not enabled for the certification. For example, if a certification has neither a Challenge nor a Remediation phase enabled but the command requests that it be advanced to the Challenge phase, the certification is advanced to the End phase.

Note: The certification is sequentially advanced through all enabled phases until it reaches or passes the requested phase. Any business logic that should occur during each phase transition (period enter rules, period end rules, etc.) is executed during the phase advancement.

Syntax	`certificationPhase <certificationName or ID> [Challenge | Remediation | End]`
Examples	`> certificationPhase "Catherine Simmons Access Review" Challenge`
Result	Advances the "Catherine Simmons Access Review" certification from its current phase (Active) to the Challenge phase. If this review is not configured for a Challenge phase, it is transitioned to the Remediation or End phase (depending on configuration).

Impact

The **impact** command reads an XML file containing a Bundle (role) object and performs role impact analysis for the role. The command parses the XML to its object form. Impact analysis is not performed if that object is not a Bundle.

Syntax	`impact filename`
Examples	`> impact c:\data\ContractorRole.xml`
Result	Performs role impact analysis for the Bundle object represented by the XML in `c:\data\ContractorRole.xml`

Event

The **event** command schedules a workflow to run, passing in an Identity name as an argument. By default, the workflow is scheduled 1 second after the command is issued, but a delay can be specified in seconds as a command argument.

Syntax	`event <identityName or ID> <workflowName or ID> [seconds]`
Examples	`> event Catherine.Simmons "Identity Refresh" 60`
Result	Schedules an Identity Refresh workflow to run for Catherine.Simmons 60 seconds after the command is issued

ConnectorDebug

The **connectorDebug** command is used to test a connector or troubleshoot application aggregation issues. Its method parameters determine what is tested and how.

Syntax	`connectorDebug <applicationName or ID> <method> [methodArgs…]`

The specific syntax for each of the "methods" is shown below.

Method	test

Purpose	Test whether a connection can be established with the application through its connector
Syntax	`connectorDebug <applicationName or ID> test`
Example	`> connectorDebug ADAM test`
Result	Returns “Test Succeeded” on success, reports an error in the console on failure.

Method	iterate
Purpose	Iterate through the application's account or group records
Syntax	`connectorDebug <applicationName or ID> iterate [account|group (default = account)] [-q (for "quiet mode")]`
Example	`> connectorDebug ADAM iterate -q` `> connectorDebug ADAM iterate account`
Result	First example iterates all account records natively in the ADAM application and returns only the count of iterated objects and how many milliseconds it took to run. Second example iterates account records natively in the ADAM application and returns a ResourceObject representation of each account to the console.

Method	get
Purpose	Test whether a connection can be established with the application through its connector
Syntax	`connectorDebug <applicationName or ID> get account|group nativeIdentity`
Example	`> connectorDebug ADAM get account` `“CN=Willie.Gomez,DC=sailpoint,DC=com”`
Result	Returns the XML representation of the ResourceObject for that nativeIdentity on the application

Method	auth
Purpose	Test pass-through authentication against the specified application (The featuresString in its application definition must contain AUTHENTICATION.)
Syntax	`connectorDebug <applicationName or ID> auth username password`
Example	`> connectorDebug ADAM auth administrator Pa$$w0rd`
Result	Returns “Authentication Successful” when user is authenticated or displays the exception message to the console if authentication fails

Encrypt

The **encrypt** command is used to encrypt a string. This command is generally only useful for test purposes. It can generate an encrypted password which can be passed in other console commands, for example, the **authenticate** command.

Syntax	`encrypt` *`string`*
Examples	`> encrypt MyPa$$w0rd`
Result	Returns the encrypted equivalent for the specified string

HQL

The **hql** command executes a search based on a Hibernate Query Language statement. The command syntax matches the sql command's syntax, but this command can select but not update data.

Syntax	`hql` *`hqlStatement`* `| -f` *`inputFileName`*

Examples	`> hql "select name, manager.name from Identity" > c:\data\Identities.dat`
	
	`> hql -f c:\hql\SelectIdentities.hql`

Result	The first example executes the specified HQL select statement and writes the results to the file `c:\data\Identities.dat`. The second example reads the HQL from the file `c:\hql\SelectIdentities.hql`, prints the HQL to the console (stdout), and displays the query results to the console (stdout).

Date

The **date** command shows the current date and time for the application server or the date and time value for a specified utime (universal time) value.

Syntax	`date [`*`utime`*`]`

Examples	`> date`
	
	`> date 1338820492484`

Result	The first example displays the command syntax and the current date/time and current UTIME value. The second example returns the date/time value for the specified UTIME value.

Shell

The **shell** command escapes out to the command line and runs the command specified. (This command does not work properly in a Windows environment but does work in UNIX.

Syntax	`shell` *`commandLine`*
Examples	`> shell ls`
Result	Lists the contents of the UNIX file system directory from which the console was run

Meter

The **meter** command toggles metering on or off. While metering is on, the console reports some timing statistics for each command executed. Meter information is displayed after the results of each command as it is executed.

Syntax	`meter`
Examples	`> meter`
Result	Toggles metering on and off. When turned on, all subsequently issued commands report timing statistics. Meter information displayed includes: number of calls, total number of milliseconds, maximum time for one call, minimum time for one call, and average time per call.

Compress

The **compress** command is designed to compress the contents of a file to a string that can be included within an XML element. It compresses the file and then encodes it to Base64 and writes that text to the specified output file. This resultant file can then be used in an XML element stored in the database. This has limited usefulness within IdentityIQ since no part of the application is designed to read these compressed strings, but custom rules can be used to process them as needed or they can simply be stored in the database to be retrieved and uncompressed for use by an external application at a later time.

Syntax	`compress` *`inputFilename outputFilename`*
Examples	`> compress file1.txt file2.txt`
Result	Compresses the contents of `file1.txt`, encodes that into Base64, and writes the resultant text string to `file2.txt`

Uncompress

The **uncompress** command functions in exactly the opposite way of the **compress** command, taking a compressed, Base64-encoded file and returning its uncompressed format.

Syntax	`uncompress` *`inputFilename outputFilename`*
Examples	`> compress file2.txt file3.txt`
Result	Reverses the compressing process to return the original, uncompressed version of the text, writing that to the file `file3.txt`

ClearEmailQueue

The **clearEmailQueue** command deletes all queued but unsent email messages from the IdentityIQ email queue. This includes any new messages that have not yet been sent and messages that have encountered problems that prevented successful delivery.

Syntax	`clearEmailQueue`
Examples	`> clearEmailQueue`
Result	Deletes all unsent emails from the email queue

ClearCache

The `clearCache` command removes objects from the Hibernate object cache. This can be used when debugging Hibernate issues.

Syntax	`clearCache`
Examples	`> clearCache`
Result	Clears the Hibernate object cache

Service

The **service** command provides information about the background services running in the console. The services include:

- Cache - periodically refreshes cached objects
- SMListener - listens for change events from PE2 change interceptors
- ResourceEvent - looks for change events added to a queue and processes them
- Heartbeat - maintains a Server object for each IdentityIQ instance and periodically updates it so you can tell if an instance is still running
- Task - the Quartz task scheduler
- Request - the IdentityIQ request processor - stopping the Request service also stops partitioned tasks

Syntax	`service list | start | stop | run`
Examples	`> service list`
Result	Lists background services running in the console

Reporting

This section contains the following information:

Chapter 24: Reports Introduction

Reports provide an at-a-glance view of the data in the IdentityIQ instance, which helps the organization manage system access and the compliance process. Out of the box, IdentityIQ includes a set of core reports in template form. Individual users and organizations can specify and save customized instances of these templates and run these reports on a scheduled or ad-hoc basis. Additionally, custom reports can be created to meet the needs of each customer.

IdentityIQ includes a reporting architecture that simplifies the process of creating custom reports. Basic reports can be created quickly through an XML specification. A variety of hooks are available for introducing more complex logic where it is needed to produce the desired report output. The standard report templates that are part of the product are modeled with this same XML specification structure and can serve as helpful examples of how custom reports should be structured.

This document explores the report interface and the process of specifying filters to create customized instances of the available report templates. It also describes the procedures required to create custom report templates that can also be used to create customized report instances with specific saved filters.

Report Terminology

This set of terms helps clarify the discussion of the options available for creating custom reports and customizing report templates by eliminating confusion around which option is which. The following terms are used:

- **Report Templates**: Out-of-the-box reports as provided with the standard IdentityIQ product. These report templates are on the Reports tab of the Reports window. The reports can be run directly or edited to create customized versions. These reports are also referred to as out-of-the-box reports, standard reports, or standard report templates.
- **Custom Reports**: Customer-specific reports developed by or specifically for a single customer through a custom Task Definition specification. These reports are on the Reports tab of the Reports window after they are saved in IdentityIQ. These report are also referred to as custom report templates.
- **Customized Report Instances** — User-specific report versions with pre-specified parameters. These reports are on the My Reports tab of the Reports window. Instances apply to out-of-the-box reports and custom reports. The terms customized report or instance can also be used in discussing these report specifications.

Chapter 25: Report Use and Customization

IdentityIQ includes a number of standard reports that are helpful in monitoring and managing compliance and provisioning activities. These reports can be run with or without filter specification. For example, the Uncorrelated Accounts Report can run with no filters and return the list of uncorrelated accounts for all applications in the system, or the report user can specify filters on the report to restrict the results to a subset of applications. The unfiltered, standard version of each report is accessible and able to be run from the Report window's Reports tab. When a user chooses to add filters, that report configuration is saved as a customized report instance on the My Reports tab.

To access the Reports page, from the Navigation menu bar, go to **Intelligence -> Reports**.

Reports Tab

The first tab visible on the Reports window is the My Reports page. However, the first time a user access the Reports page, the My Reports list is empty because My Reports only shows the customized reports you created and saved based on a report template. The second tab, the Reports tab, is where a new user must start interacting with reports.

The Reports tab lists all of the available report templates, grouped by report category. The out-of-the-box report categories are:

- Access Review and Certification Reports
- Account Group Reports
- Activity Reports
- Administration Reports
- Configured Resources Reports
- Identity and User Reports
- Policy Enforcement Reports
- Risk Reports
- Role Management Reports

For each report template, the report name and a brief description of its contents are shown. Reports can be run directly from this page or can be scheduled to run at some point in the future, either once or on a repeating, scheduled basis. Any report initiated (immediately or scheduled) from this page is run with no filters applied. In other words, the report runs for all system objects to which that report applies (all roles, all Identities, all policies, all access reviews, etc.).

To run a report with no filters, right click the report on this window and then choose **Execute** to run it once immediately or Schedule to set it up to run in the future or on a repeating basis.To completely remove the report from the system, click **Delete**.

Alternatively, filters and other specifications can be applied to a report and it can be saved as a new report instance with those parameters already in place. Click the report name in the list or right-click the report and choose **Save as New Report**. Both of these options open the Edit Report page that, displays the available filters and parameters for the report.

Edit Report Page

The Edit Report page allows the user to specify filters for the report, saving that configuration as a customized report instance for future re-use. In the new reporting architecture, every report specification is separated into multiple Sections. Every report specification includes a Standard Properties and Report Layout section as the first and last sections, respectively. Any parameters specific to a given report are specified in one or more sections between these two. Navigate between the "section" pages by clicking the desired page in the Sections list or by clicking the Next and Previous buttons at the bottom of the Edit Report page.

Standard Properties

This common set of properties applies to every report, so the Standard Properties page is presented as part of the Edit Report page for every report.

The table below lists the fields on the Standard Properties page and describes their usage.

Table 96—Reports Standard Properties

Field Name	Description	Required?
Name	The name for the report instance - shown on the **My Reports** window as the report's name and on the Report Results page when the report is run	Yes
Previous Result Action	Determines what is done with the results of previous runs of this report when it is run again **Rename Old** — renames existing report results by appending a numeric value to them **Rename New** — renames the run results of the new report by appending a numeric value to it **Delete** — deletes any previously generated report results for this report, replacing them with the new results **Cancel** — prevents the report from running if a previous result exists	Yes
Description	Brief description for the report - shown on the **My Reports** window as the report's description (defaults to the report template's description)	No
Allow Concurrency	Determines whether more than one instance of the report is allowed to be running at a time	No
Scope	Assigns a scope to the report which allows it to be seen and used by any user authorized to that scope; this is the easiest way to implement report sharing	No
Email Recipients	Names Identities to whom the report results should be emailed when it is generated; these users can view the report results but are not required by the system to act on them in any way	No

Table 96—Reports Standard Properties

Field Name	Description	Required?
Require Signoff	Determines whether anyone is required to sign off on the report; when this is selected, the **Signoff Properties** section of the form appears where the email template and signers are specified; a work item is created for each of these signers	No
Initial Notification Email	Specifies the email template for the email that is sent to notify the signers of the report work item created for them	Yes, if signoff required
Escalation	Specifies whether reminders should be sent or escalation should be enacted for the signoff work items. If either or both are selected, additional parameters specifying the timing of reminders/escalation are displayed and must be entered	Yes (though None is the default)
Signers	Specifies the Identities who must sign off on this report.These Identities receive a notification email and a work item is created.	Yes, if signoff required

Report Layout

The **Report Layout** page shows the columns that are available for inclusion on the report and the columns selected for inclusion on the report. It allows the user to select a sort field and a group field if desired. It also allows suppression of either the summary or the detail information as needed.

These are the fields on the Report Layout page, along with descriptions of their usage.

Table 97—Report Layout Fields

Property Name	Description	Required?
Sort By	Specifies the field by which the detail records in the report should be sorted; if a Group By field is also selected, grouping supersedes sorting and the records within each group are sorted by the Sort By field; only columns marked as sortable (sortable="true" in XML) are included in this selection list	No
Group By	Specifies the field by which the detail records in the report should be grouped; the group field value is displayed as a section header within the report body in the on-screen display of the report and in the PDF (not in the downloaded CSV format) ; only columns marked as sortable (sortable="true" in XML) are included in this selection list	No
Columns	Lists columns available for or included in the report; this section is split into two lists: the left list indicates available columns that are not selected for inclusion in the report body. The right list displays the columns that will be in the report in the order in which they will appear. Column names can be dragged from one side to the other and can be reordered within the list with drag-and-drop. The arrow buttons between the two lists can also be used to move columns around.	Yes
Disable Report Summary Display	Suppresses the summary section of the report output	No

Table 97—Report Layout Fields

Property Name	Description	Required?
Disable Report Details Display	Suppresses the detail section of the report output	No

Report-Specific Parameters

Most reports implement at least one page between the Standard Properties and Report Layout pages. These are named differently for each report, and some reports implement several pages while others include only one. These pages allow the user to specify filter parameters for the report instance. For example, the Uncorrelated Accounts Report contains one report-specific settings page called **Uncorrelated Accounts Parameters**; this page enables the user to specify the **Application** for which they want to see the a list of accounts that could not be correlated to existing Identities (from the authoritative application). If no application is selected in this filter, the report shows all uncorrelated accounts from all applications.

In some cases, the **Report Layout** column list will change based on the parameters set on the report-specific parameters pages. For example, the User Account Attributes Report can display account attributes on selected applications. If the application selected has attributes (for example, privileged or service accounts) which other applications don't have, when that application is selected for the report, those columns appear on the Report Layout page for optional inclusion in the report.

Saving and Executing Report Instances

Once a report has been customized with filter parameters, sorting and grouping specifications, and custom column selections or order, it can be saved as a My Reports report instance for future use/re-use. There are several save options available:

- **Save**: saves the report instance specification onto the My Reports tab under the name provided on the Standard Properties window
- **Save and Preview**: saves the report specification onto the My Reports tab and runs a preview of the report, which displays the summary section (unless suppressed) and the first page of detail results (20 records); allows the user to verify that the report shows the type of data they want to see; does not save the report results for later viewing
- **Save and Execute**: saves the report specification onto the My Reports tab and runs the report; the report results are saved to the database and can be recalled from the Report Results tab until they are deleted

Modifying through Preview Mode

The **Save and Preview** option enables the user to modify the report output layout, including rearranging columns, changing the detail sort order, and hiding either the summary or detail section. Any changes made in preview mode can be saved to the report specification, allowing Preview mode to function as an interactive method of reconfiguring the report output. A message at the top of the report prompts the user to choose whether to **Save Changes** or **Cancel Changes** when they alter the appearance of the report preview.

When a report is executed (as opposed to previewed), the final results cannot be reordered in the on-screen display of the report. The report can, however, be downloaded as a CSV and manipulated in a spreadsheet application if desired.

Reports without Preview Option

A few of the reports cannot be viewed in preview mode; this is because the data in these reports cannot be polled without fully executing the report. For example, the Identity Forwarding Report shows the forwarding user for all Identities who have one specified. Because the forwarding property is not searchable, these cannot be counted up front and the report cannot be previewed. In some cases, a report can be previewed unless certain options on it are selected. The Manager Access Review Report, for example, is available for previewing unless the Show Excluded Items option is selected. This option requires a union that cannot be done with the preview option, so previewing must be disabled. In these cases, a message is shown indicating that preview is not available when the user clicks **Save and Preview**.

My Reports Tab

When customized report instances are saved, they appear in the list for the user. From this page, any defined report instance can be opened, edited to change the instance's specifications, and saved with updates (with or without running the instance). Instances can also be used to create new report instances and can be run, scheduled, or deleted through their right-click menu options. The right-click menu options are:

- **Save as New Report** — Creates a new report instance based on an existing instance. Changes made are saved as a new report instance, leaving the existing one intact. This is particularly helpful when a report offers a large number of configuration parameters and a new instance is being created that changes only a few of the options (filters, sorts, etc.).
- **Edit** — Edits the existing report instance in place. This is the same as clicking the report instance in the list
- **Schedule** — Schedules the report instance for future (or repeated) execution.
- **Execute** — Runs the report instance immediately.
- **Delete** — Deletes the instance’s configuration from the system.

Scheduled Reports Tab

When a user chooses to schedule a report template or instance for execution (right-click -> **Schedule**), the New Schedule specification window is displayed. The user must specify a name for the scheduled version of the report and can optionally specify a description. The report execution is scheduled at the date and time entered by the user and can be set to run at various frequencies (once, hourly, daily, weekly, monthly, quarterly, or annually).

Reports that have been scheduled for future or repeated execution appear on the Scheduled Reports tab. Once the scheduled report has executed for the last time (for example, it was originally scheduled to run once in the future and that time arrived so it ran), it is removed from this list; only future-scheduled report executions are shown on this page.

Report Results Tab

The Report Results tab shows the completion status of all reports that have run and are currently available for viewing.

Click any report in the list to view its specific results. From the Report Result window, the report can be viewed on screen, downloaded as a PDF, or downloaded as a CSV file. In the new reporting architecture, the PDF and CSV forms of the report are created during report execution and saved to the database, making the download of either format fast and efficient.

XML Representation of Reports and Instances

Standard report templates are represented in the IdentityIQ object model as TaskDefinition objects. The XML representation of these can be seen through the IdentityIQ Debug pages by selecting Task Definition from the **Object Browser** list and searching for the report's name in the **Name** column.

When a report is saved as a customized instance, a new XML object is created in the system to represent its custom configuration. The instance's XML is far simpler than the template's because it references the template for most of the report generation details.

Details on these XML representations are explored further in the next chapter: *Developing Custom Reports*.

Chapter 26: Developing Custom Reports

IdentityIQ includes a reporting architecture that greatly simplifies the process of developing custom reports by allowing the developer to specify the report requirements in a TaskDefinition XML document. The executor uses IdentityIQ's Forms API to generate the UI form for parameter specification and creates the report output based on column con figurations specified in the TaskDefinition. The XML specifies the report's Standard Properties values, the report-specific parameters, the columns that are available for the report, how the data is retrieved for inclusion on the report, and how the report results are laid out in both the detail and summary sections.

The standard report templates provide some good examples of how to define reports through XML. Excerpts from these standard templates are used in this chapter to illustrate how to configure custom reports. Many of the excerpts come from the Uncorrelated Accounts Report, a fairly simple example that can be used to explore the basics of defining a custom report.

It might be helpful to examine the full XML for these reports to see the tags' usage in context as they are referenced and excerpted in this document. The reports can be viewed through the debug pages as described in XML Representation of Reports and Instances, or the entire set of TaskDefinition objects can be exported to a file through the iiq console and explored in a text editor. The console export command to write the system's TaskDefinition objects to a file is “export taskDefs.xml TaskDefinition” (where “taskDefs.xml” is the name of the file to which the objects are exported). Note that the file contains all tasks including reports because no filter available on the export command to select only a subset of objects of a given type.

Report as a TaskDefinition

In IdentityIQ, a report is essentially executed as a specialized task. The root element of a report is a <TaskDefinition> element.

```
<TaskDefinition executor="sailpoint.reporting.LiveReportExecutor" name="Uncorrelated
Accounts Report" progressMode="Percentage" resultAction="Rename" subType="Identity
and User Reports" template="true" type="LiveReport">
```

The type attribute for the TaskDefinition indicates that this is a report definition, and the executor specifies which class processes this task definition to run the report. The attributes of the TaskDefinition object and their purposes are described in the table below.

Table 98—TaskDefinition Type Attributes

Attribute	Usage
executor	Indicates the class to run the report. In previous version of IdentityIQ, custom reports required a custom report executor. The new architecture includes the “sailpoint.reporting.LiveReportExecutor”, which is always specified as the executor for any report of type “LiveReport,” including custom reports.

Table 98—TaskDefinition Type Attributes

Attribute	Usage
name	Name of the report template; shown on the **Reports** list as the report's **Name** **NOTE: When templates are edited, they must be saved as customized report instances, and the name value across all report templates and report instances must be unique. Therefore, the name attribute for a template is not displayed as the Name field's value on the Edit Report window's Standard Properties page since instances cannot be saved with this same name. The value entered in the Standard Properties page's Name field becomes the name value of the TaskDefinition XML for that instance.**
progressMode	Specifies how the executor updates progress while the report is being executed; most reports use Percentage. Possible values are: **None** — executor doesn't update progress (same as null, or not specifying) **String** — executor periodically updates progressString property of the result during execution **Percentage** — executor periodically updates the progress and percentageComplete properties of the result during execution
resultAction	Specifies the **PreviousResultAction** (shown on the Standard Properties page) - states how to manage the results from previous runs of this report when it is executed again. Possible values are: **Rename** — rename old report results by appending a numeric value (for example, Uncorrelated Accounts Report - 2) **RenameNew** — rename new report results by appending a numeric value **Cancel** — do not run the report when old report results still exist for the report (displays an error message indicating that a result from a previous execution of the report still exists) **Delete** — delete old report results when the report is executed again
subType	Indicates the report category to which this report belongs (sub-categories within the **Reports** and **My Reports** tabs); can be one of the out-of-the-box subTypes or a custom subType
template	Boolean indicating whether this is a report template (appears on the **Reports** tab) or a customized report instance (appears on **My Reports** tab); new custom reports should be set up as template=”true”
type	All reports using the new reporting architecture, including new custom reports, are of type “LiveReport”; pre-6.0 reports are of type “Report”

Note: When a report XML document is imported into IdentityIQ, two attributes - created and id - are generated by the system and saved as part of the TaskDefinition; these should not be specified in the XML by the report developer.

Elements within TaskDefinition

The TaskDefinition contains several nested elements that are used to define important information for the report. The TaskDefinition always contains an Attributes map and a Signature and usually contains a Description element and a list of RequiredRights.

Attributes Map

The **Attributes** map minimally must contain the report definition that described in the next section..

```
<Attributes>
   <Map>
     <entry key="report">
       <value>
         <LiveReport title="Uncorrelated Accounts Report">
             …
         </LiveReport>
       </value>
     </entry>
   </Map>
 </Attributes>
```

Other optional attributes in the attribute map include emailIdentities, reportSortBy, reportGroupBy, disableSummary, and disableDetail, as described in the Standard Forms for Report Specification section.

Signature

The Signature contains a map of Input attributes that names all of the parameters that can be specified for the report.

```
<Signature>
    <Inputs>
      <Argument multi="true" name="correlatedApps" type="Application">
        <Description>rept_input_uncorrelated_ident_report_correlated_apps
        </Description>
        <Prompt>report_input_correlated_apps</Prompt>
      </Argument>
      <Argument name="resultScope" type="Scope">
        <Description>rept_input_result_scope</Description>
      </Argument>
      <Argument multi="true" name="emailIdentities" type="Identity">
        <Description>rept_input_email_recips</Description>
      </Argument>
    </Inputs>
 </Signature>
```

When a customized report instance with pre-populated parameters is saved, those parameters are saved as a part of the instance's TaskDefinition in its Attributes map. In this example, a list of Applications and a list of Email Recipients have been saved for this report instance.

```
<TaskDefinition created="1344453735712" id="4028460238edaba4013907aff5200ec9"
modified="1344867911170" name="Uncorrelated Accounts" resultAction="Rename"
subType="Identity and User Reports" type="LiveReport">
 <Attributes>
    <Map>
       <entry key="correlatedApps">
        <value>
          <List>
            <String>4028460238edaba401372767b6eb0d70</String>
            <String>4028460238edaba40138edcfcd1102d2</String>
          </List>
        </value>
      </entry>
      <entry key="disableDetail" value="false"/>
      <entry key="disableSummary" value="false"/>
      <entry key="emailIdentities">
        <value>
          <List>
            <String>4028460238edaba40138edb3571e000d</String>
          </List>
        </value>
      </entry>
      <entry key="reportColumnOrder" value="username, firstName, lastName"/>
    </Map>
 </Attributes>
```

They are passed from the customized report instance to the associated report template at run-time through the taskDefinition input arguments specified in the <Signature> for the report.

Every report template's Signature should include input arguments for resultScope and emailIdentities, since these automatically appear on the Standard Properties window and are available for a user to specify on all reports. All other input arguments are report-specific and the remainder of the arguments in a custom report is specific to that report. The list of arguments should match the set of fields available for parameter specification on the report's Form. Report-specific arguments should include a name and type as attributes on the <Argument> element. If the argument is multi-valued, it should also include the attribute “multi=”true”.”

Application arguments can include a Description and Prompt element. When a custom form (in a <ReportForm> element) has been specified, these are ignored and can be omitted. However, if no custom form is specified, these report-specific input arguments are automatically rendered on a form page (titled Report Options) using the Prompt value as the field label and the Description value as the tool tip for the field. Both of these values can be specified as strings or as localizable message keys.

```
<Argument multi="true" name="correlatedApps" type="Application">
 <Description>rept_input_uncorrelated_ident_report_correlated_apps
 </Description>
```

```
 <Prompt>report_input_correlated_apps</Prompt>
```

The Account Group Members report is an example of a report that relies on this automatic form rendering for its report-specific filter options.

Description

The Description element is displayed as the Description for the report on both the Reports and My Reports lists and on the Edit Report window.

```
<Description>A detailed view of the uncorrelated user accounts in the
system.</Description>
```

Required Rights

The RequiredRights element specifies what system right(s) a user must have to be able to see and execute the report. The required rights are specified as references to one or more SPRight objects.

```
<RequiredRights>

  <Reference class="sailpoint.object.SPRight" id="4028460238ed9b8e0138ed9bc59d0054"
name="FullAccessUncorrelatedIdentitiesReport"/>

</RequiredRights>
```

Report Definition

The report, including its custom UI form or forms, its query specification, and its results contents and layout, is specified as a part of the TaskDefinition's attributes map with the attribute key “report”. The value for this attribute is a <LiveReport> element.

```
<Attributes>
    <Map>
      <entry key="report">
        <value>
          <LiveReport title="Uncorrelated Accounts Report">
```

Elements within <LiveReport> determine the report filters that can be specified by a report user, the query used to retrieve the data for the report, the layout of the report detail grid, and the contents and layout of the report's summary table and chart. The nested elements within <LiveReport> are:

Table 99—LiveReport Nested Elements

Element	Description
ReportForm	Determines how the report-specific parameters sections are presented to the user in the Edit Report window (see ReportForm: Collecting Report-Specific Parameters)
DataSource	Specifies how the data for the report details is retrieved from the database (see DataSource: Retrieving Report Data)
Columns	Lists columns available for inclusion in the report detail grid; also used in conjunction with DataSource to determine which data elements are retrieved in the query (see Columns/ReportColumnConfig: Report Grid Presentation)

Table 99—LiveReport Nested Elements

Element	Description
ReportSummary	Describes the Summary section of the report - information included, query to retrieve it, layout and labels for presentation of it (see ReportSummary: Summary Table)
Chart	Defines the graph or chart displayed in the Summary section for the report (see Chart: Report Graph)

Each of these elements is explored in detail in the next sections.

ReportForm: Collecting Report-Specific Parameters

Most reports allow users to specify filters that constrain the contents of the generated report. Report-specific parameters are collected from the report user through a custom form, referenced through a ReportForm element in the report definition. The form must be specified as a separate XML document and imported into IdentityIQ. The Form object is described in the Report Forms section at the end of this document.

The ReportForm element references the form like this:

```
<ReportForm>

<Reference class="sailpoint.object.Form" id="4028460238edaba40138edb36b330010"
name="Uncorrelated Account Report Custom Fields"/>

</ReportForm>
```

Standard Forms for Report Specification

The referenced ReportForm is presented to the user in the Edit Report window between the two standard form pages that are part of every report's specification: Standard Properties and Report Layout. Those two standard pages are rendered based on a Form object called Report Skeleton using values specified in the report's TaskDefinition XML. These tables indicate which TaskDefinition elements and attributes determine the values for fields on the Standard Properties and Report Layout pages.

Standard Properties Field	TaskDefinition Source
Name	If editing a customized report instance, <TaskDefinition> name attribute `<TaskDefinition name="Uncorrelated Accounts - Financials" … >` If creating a new report instance based on a template, none (not populated)
Previous Result Action	`<TaskDefinition> resultAction attribute<TaskDefinition name="Uncorrelated Accounts Report" resultAction="Rename" … >`
Description	<Description> element
Scope*	resultScope entry in Attributes map (value contains ID of selected scope) `<entry key="resultScope" value="2c9082ee38e813a20138e934eb210146"/>`

Standard Properties Field	TaskDefinition Source
Email Recipients*	emailIdentities entry in Attributes map (List contains Identity ID values) `<entry key=”emailIdentities”> <value> <List> <String>4028460238edaba40138edb35653000b</String> </List> </value> </entry>` Usually specified in instance XML instead of template XML
Allow Concurrency	`<TaskDefinition> concurrent attribute (concurrent=”true”)<TaskDefinition concurrent="true" name="Uncorrelated Accounts Report" … >`
Require Signoff*	<SignoffConfig> element `<SignoffConfig> <WorkItemConfig created="1344962495866" escalationStyle="none" id="4028460238edaba401392603057a1464"> <NotificationEmailTemplateRef><Reference class="sailpoint.object.EmailTemplate" id="4028460238ed9b8e0138ed9bd8690106" name="Default Report Template"/> </NotificationEmailTemplateRef> <Owners><Reference class="sailpoint.object.Identity" id="4028460238edaba40138edb36b33016d" name="Aaron.Nichols"/> </Owners> </WorkItemConfig> </SignoffConfig>`

Report Layout Field	TaskDefinition Source
Sort By*	reportSortBy entry in Attributes map `<entry key="reportSortBy" value="accountGroupDisplayName"/>`
Sort Ascending*	reportSortAsc entry in attributes map `<entry key="reportSortAsc"> <value> <Boolean>true</Boolean> </value> </entry>`
Group By*	reportGroupBy entry in Attributes map `<entry key="reportGroupBy" value="application"/>`

Report Layout Field	TaskDefinition Source
Columns	<ReportColumnConfig> header attributes; hidden="true" attribute places column in left pane - available but not included on report detail by default `<ReportColumnConfig field="accountGroupName" header="rept_app_account_grp_memb_col_name" property="value" sortable="true" />`
Disable Report Summary Display*	disableSummary entry in Attributes map `<entry key="disableSummary" value="true"/>`
Disable Report Detail Display*	disableDetail entry in Attributes map `<entry key="disableDetail" value="true"/>`

* These fields are typically specified through the UI for customized instances of reports and saved into the My Reports instances' attributes map, rather than being specified in the report template XML. However, they can be specified in the template XML if they apply to the report's default configuration.

Although most reports do include a custom form, it is not required. When one is not specified, the Edit Report window still displays the Standard Properties and Report Layout pages; the Identity Status Summary report shows an example of this.

DataSource: Retrieving Report Data

The data shown in the detail section of the report is retrieved through a query that is built based on a combination of the <DataSource> specification and the <Columns> element. In general, a query is specified in three parts: Select, From, and Where. The Select portion (the columns list) is specified through the <Columns> element in the report definition - specifically, the <ReportColumnConfig>s listed within <Columns> element. The From and Where clauses are specified through the <DataSource> element.

There are three available datasource types: Filter, Java, and HQL. The simplest of these three is the Filter datasource, though various options available with this datasource type make it quite powerful and flexible. The other two are available for more complex report data retrieval needs, and Java is likely to be used as the datasource more often in HQL in those cases. Each of these three datasource types is discussed next.

Filter DataSource

A filter datasource executes a projection query to retrieve the data required by the ReportColumnConfigs specified for the report. It employs the SailPoint Filter object to specify the query. The object whose data is being queried is specified as the objectType for the DataSource, and the DataSource type is specified as “Filter”.

```
<DataSource objectType="sailpoint.object.Link" type="Filter">
```

Note: If the objectType is one of the top-level classes in the IdentityIQ object model (for example, the set of objects that can be exported from the iiq console or retrieved directly in from the debug pages), the fully-qualified class name is not required for this attribute. For example, Identity can be specified here as objectType=”Identity”. However, the fully-qualified name (for example, sailpoint.object.Identity) is always acceptable, even for the top-level classes, so when in doubt, specify the fully-qualified name.

This is an example of a filter <DataSource> and its <Columns> specification:

```
<LiveReport title="Uncorrelated Accounts Report">

 <DataSource objectType="sailpoint.object.Link" type="Filter">

   <QueryParameters>

<Parameter argument="correlatedApps" property="application.id"/>

       <Parameter defaultValue="false" property="identity.correlated"
valueClass="Boolean"/>

       <Parameter defaultValue="false" property="application.authoritative"
valueClass="Boolean"/>

       <Parameter defaultValue="false" property="application.logical"
valueClass="Boolean"/>

   </QueryParameters>

 </DataSource>

 <Columns>

<ReportColumnConfig field="username" header="rept_uncorrelated_ids_grid_username"
property="nativeIdentity" sortable="true" />

<ReportColumnConfig field="firstName" header="rept_uncorrelated_ids_grid_firstName"
property="identity.firstname" sortable="true" />

<ReportColumnConfig field="lastName" header="rept_uncorrelated_ids_grid_lastName"
property="identity.lastname" sortable="true" />

<ReportColumnConfig field="applicationName"
header="rept_uncorrelated_ids_grid_appName" property="application.name"
sortable="true" />

 </Columns>
```

The search criteria, making up the “where” clause for the search, are specified through one or more of several query-related elements: Query, QueryParameters, and QueryScript. Joins, sorts and groupBy columns can also be specified as needed for the query.

QueryParameters

The <QueryParameters> element is used most often. QueryParameters is a map of argument values used to create the queryOptions object that controls the search. They can be specified based on report arguments, hard-coded values, or calculated values. QueryParameters contains a list of <Parameter> elements, each of which defines one of the criteria. These <Parameter>s are "anded" together to make the where clause.

```
<QueryParameters>

   <Parameter argument="correlatedApps" property="application.id"/>

   <Parameter defaultValue="false" property="identity.correlated"
valueClass="Boolean"/>

   …

</QueryParameters>
```

There are several different options for specifying parameters in a set of QueryParameters. These options are described below, illustrated with example Parameters. Most of these examples (except where noted) were taken

from the Entitlement Owner Access Review Live Report which queries against the sailpoint**.object.CertificationItem** object, so all of these parameters relate to that object.

- **Referencing a report argument**: generally processed as “property = argument”; this parameter looks for certificationItems with a parent.certification.certificationGroups.id value in the report argument “certificationGroups”

```
<Parameter argument="certificationGroups"
property="parent.certification.certificationGroups.id"/>
```

  Note: When arguments are multi-valued, parameters based on them are automatically evaluated with “in” rather than “equals”.

- **Specifying a default value**: generally processed as “property = argument or defaultValue (if argument is null)”; this parameter looks for CertificationItems with a parent.certification.type equal to the report argument “type”; if none is provided, it defaults to the type “DataOwner”

```
<Parameter argument="type" defaultValue="DataOwner"
property="parent.certification.type"
valueClass="sailpoint.object.Certification$Type"/>
```

  Note: This example also illustrates usage of the valueClass attribute; this attribute is not necessary for string or object comparisons but is for other types, such as enumerations (such as Type in this example), Booleans, Dates, Lists, etc.

- **Specifying a hard-coded value**: an attribute can also be hard coded to be evaluated against the defaultValue by not including an argument, as shown in this parameter from the Uncorrelated Accounts Report. This is processed as “property = defaultValue”, in this case cast as valueClass (not required for strings).

```
<Parameter defaultValue=”false” property=”identity.correlated”
valueClass=”boolean”/>
```

- **Specifying different operations**: this example illustrates how to create evaluation conditions other than equals (or in) relationships; operation can be specified as GT, GE, LT, or LE (greater than, greater than or equal to, less than, less than or equal to)

```
<Parameter argument="createStartDate" operation="GT"
property="parent.certification.created"/>
```

- **Using a ValueScript**: processed as "property = return value from ValueScript"; this parameter performs processing based on the argument to return a different value that should be used in the criterion; this example uses a ValueScript to get the application name that corresponds to the applicationID in the "application" report argument; in a ValueScript, the argument is accessed through the variable name "value".

```
<Parameter argument="applications" property="parent.application">
   <ValueScript>
      <Source>
         import sailpoint.object.*;
         import sailpoint.api.ObjectUtil;
         if (value != null){
            return ObjectUtil.convertIdsToNames(context, Application.class,
value);
         }
         return null;
      </Source>
   </ValueScript>
```

```
</Parameter>
```

Note: Since object references are stored in the customized report instance XML (and passed to report input arguments) as ID values and many comparisons need to be done based on name, this convertIdsToNames() utility method is frequently used in ValueScripts in the standard reports.

- **Using a QueryScript**: used to specify any custom filter and add it into the queryOptions object that is used in the datasource filter; parameters using a QueryScript do not need to specify a property because the queryScript overrides any property on the parameter; the argument specified on the parameter can be accessed within the script through the "value" variable

  Group and populations are stored in groupDefinitions objects as a filter, so this example (from the Identity Forwarding Report) shows how a group or population selected as a report parameter is built into the datasource filter through a QueryScript.

```
<Parameter argument="groupDefinitions">
   <QueryScript>
      <Source>
         import sailpoint.object.*;
         import sailpoint.reporting.*;
        Filter f = ReportingLibrary.getGroupDefinitionFilter(context, value,
false);
         if (f!=null) {
            queryOptions.addFilter(f);
         }
         return queryOptions;
      </Source>
   </QueryScript>
</Parameter>
```

- **ValueRule and QueryRule**: These two can be specified in place of ValueScript and QueryScript, respectively, to encapsulate the beanshell of a script into a reusable rule. (These two examples were not pulled from a standard report; they represent the appropriate syntax if the reports using the ValueScript and QueryScript specified above had encapsulated those scripts into rules.)

```
<Parameter argument="applications" property="parent.application">
   <ValueRule>
      <Reference class="sailpoint.object.Rule"
id="4028460238ed9b8e0138ed9beff9090f" name="App Value Rule"/>
   </ValueRule>
</Parameter>

<Parameter argument="groupDefinitions">
   <QueryRule>
      <Reference class="sailpoint.object.Rule"
id="4028460238ed9b8e0138ed9beff90900" name="Group Query Rule"/>
   </QueryRule>
</Parameter>
```

Query

Another way to specify the filter contents is though a <Query> element. The contents of Query element are specified as a filter string and can only specify hard-coded criteria with no variable substitution (i.e. report

arguments cannot be specified within a Query element). Query allows the specification of “or” criteria, as shown in the example below:

```
<Query>IdentityEntitlement.name=="assignedRoles" ||
IdentityEntitlement.name=="detectedRoles"</Query>
```

Query and QueryParameters can be specified for the same DataSource. When both are specified, the Query filter and the Parameter filters are “anded” together to create the final where clause.

QueryScript

QueryScript creates a filter string through a beanshell script. It is designed so it can append additional criteria, including those requiring variable substitution, onto a Query element's contents. The script has access to the string value of the Query element (in a string variable called “query”) and must explicitly append the additional criteria to it; otherwise, the original query string is overwritten with the QueryScript's return value. The QueryScript shown below actually comes from an HQL datasource report (the Account Group Membership Totals Report), but the QueryScript syntax is the same for all datasource types.

```
<QueryScript>
   <Source>
      import java.util.*;


      List applications = args.get("application.id");
      if (applications != null &amp;amp;&amp;amp; !applications.isEmpty()){
         query = query + " and application.id in(:application_id) ";
      }
      return query;


    </Source>
</QueryScript>
```

Join

When the search must access more than one object to process the filter criteria, a <Join> element is required to connect the objects properly. One or more Joins can be specified for a single datasource.

For example, the Identity Roles Report displays the roles that each Identity is assigned. Most of the available filters for the report apply to the Identity object, but the role assignment is recorded on the IdentityEntitlement object, linked to the Identity object by the Identity ID. The Join element specifies that connection. The property is the value on the primary object (the DataSource objectType) and the joinProperty specifies the connection attribute on the second object.

```
<DataSource objectType="Identity" type="Filter">
   <Join joinProperty="IdentityEntitlement.identity.id" property="id"/>
   <Query>IdentityEntitlement.name=="assignedRoles" ||
IdentityEntitlement.name=="detectedRoles"</Query>
   <QueryParameters>
      <Parameter argument="identities" property="id"/>
      …
```

```
   </QueryParameters>
</DataSource>
```

OptionsRule or OptionsScript

The final elements available on a filter datasource are an OptionsRule or OptionsScript. These can be used to make modifications to the QueryOptions before the query is run; they can also replace the rest of the query specification (for example, eliminating the need for a Query, QueryParameters, QueryScript or Join element) by simply constructing the whole queryOptions in the rule or script.

Only one of these can be specified (the rule overrides the script if both are provided). The OptionsRule or OptionsScript is passed a SailPoint Context called “context”, a queryOptions called “options” and an argument map called “args”. Options contains the entire set of query criteria specified in any of the other elements (Query, QueryScript, QueryParameters, Join) and args contains the TaskDefinition argument map. The rule or script should append any additional custom queryOptions to options and return it.

```
 <OptionsScript>
    <Source>
       import java.util.*;
       import sailpoint.object.*;

       //code to add components to queryOptions goes here. e.g.: this would
       // Apply to an Identity objectType and would get only Identities whose
       // Manager is the Identity selected in the manager filter (typically,
       // an optionsScript or optionsRule would be used for a more complex

       Filter myFilter = Filter.eq("manager.id", args.get("manager.id");
       options.addFilter(myFilter);

       return options;

     </Source>
 </OptionsScript>
```

An OptionsRule is specified as a reference to a Rule object:

```
<OptionsRule>
    <Reference class="sailpoint.object.Rule" id="4028460238ed9b8e0138ed9beff90900"
name="MyReport Options Rule"/></OptionsRule>
```

Java DataSource

A Java datasource is the next most commonly used report datasource type. The XML to specify this is fairly simple and straightforward; the java class it calls can be as simple or as complex as is required to generate the desired report contents.

The java datasource class must implement the sailpoint**.reporting.datasource.JavaDataSource** interface, as described in the IdentityIQ javadocs. This interface defines all the methods that must be coded. All attributes in the taskDefinition attribute map (including all input attributes from the Signature) are passed to the Java class in an arguments map.

The <DataSource> element in the XML specifies these attributes:

DataSource Attribute	Usage
dataSourceClass	The fully qualified java class name
objectType	The primary object against which searches are performed in the java code
type	Java (tells the report executor this is a Java Datasource)
defaultSort	Optional field; sorts the returned data by the named field if no sort column is specified through the UI or taskDefinition attributes map

Note: Many of the standard reports were written with a Java Datasource and several examples of this syntax are available. Most of the standard reports use a QueryParameters element to pass data to the DataSource, which allowed the report writer to take advantage of the reportHelper class in the reporting architecture to reuse existing code. However, this is not strictly necessary and isnot commonly done in the field. Because the entire taskDefinition attributes map, including all input attributes from the <Signature>) is passed to the java class in an arguments map, they do not need to be specified as QueryParameters. The class can build the QueryOptions object needed to retrieve the data without passing the values through QueryParameters.

HQL DataSource

An HQL datasource is used in rare circumstances but is available for implementers who need to execute queries that hit Hibernate directly. This should only be used when the report developer is very knowledgeable about HQL. The HQL query must be custom written by the report developer.

Like the Filter datasource, the HQL datasource can specify its query using these types of nested elements: Query, QueryScript, and QueryParameters. The Query and QueryParameters elements function somewhat differently in an HQL datasource, though, so it is important to understand the way they are processed.

The Account Group Membership Totals Report provides an example of an HQL datasource.

```
<LiveReport title="Account Group Membership Totals Report">
   <DataSource type="Hql">
      <Query>from ManagedAttribute m where group=true</Query>
      <QueryParameters>
          <Parameter argument="application" property="application_id"/>
      </QueryParameters>
      <QueryScript>
         <Source>
            import java.util.*;

            List applications = args.get("application.id");
            if (applications != null &amp;&amp; !applications.isEmpty()){
```

```
            query = query + " and application.id in(:application_id) ";
          }
          return query;


        </Source>
      </QueryScript>
   </DataSource>
   <Columns>
        <ReportColumnConfig field="accountGroupName"
header="rept_app_account_grp_memb_col_name" property="value" sortable="true"/>
        <ReportColumnConfig field="accountGroupDisplayName"
header="rept_app_account_grp_display_name" property="displayName" sortable="true"/>
     <ReportColumnConfig field="application" header="rept_app_account_grp_memb_app"
property="application.name" sortable="true"/>
        <ReportColumnConfig field="total"
header="rept_app_account_grp_memb_col_members" property="(select count(*) from
IdentityEntitlement ie where ie.value = m.value and ie.application = m.application and
ie.name = m.attribute and ie.aggregationState = &amp;apos;Connected&amp;apos;)"/>
   </Columns>
</LiveReport>
```

In an HQL datasource, the <Query> element must specify the From clause for the query. The objectType is not required for an HQL datasource and is ignored if it is provided.

Query

The Query element can also specify some or all of the where clause. As on a Filter DataSource, the Query element can specify any hard-coded attribute evaluations (i.e. no variable substitution available) and multiple conditions can be specified with “and” or “or” relationships.

```
<Query>from ManagedAttribute m where group=true</Query>
```

QueryScript

The HQL DataSource <QueryScript> element works just like the Filter Datasource QueryScript. It contains beanshell that returns a filter string (appending to the Query's string and returning the combined string value). However, the difference in QueryParameter processing changes the way variables are processed in the script. The queryScript has access to the task argument map (in its “args” variable), so conditional processing can be done on those arguments in determining how to build the filter string. However, the contents of those variables do not need to be built into the actual query string in the queryScript; they can be referenced as variable names that are passed to the search through QueryParameters. In an HQL datasource, the search is performed based on the query string built in the query and queryScript elements; the parameters specified as QueryParameters are passed to the search method along with that query string and are substituted into the query where variable names are found.

In the example below (from the Account Group Membership Totals Report), the QueryScript examines the application.id value from the args list and if it is non-null, it appends “and application.id in (:application_id) “ to the query string. The QueryParameter application_id allows the list of applications from the task argument list to be substituted for the :application_id variable in that query string when the search is executed.

```
<QueryParameters>
    <Parameter argument="application" property="application_id"/>
</QueryParameters>
<QueryScript>
    <Source>
       import java.util.*;

       List applications = args.get("application.id");
       if (applications != null &amp;amp;&amp;amp; !applications.isEmpty()){
          // :application_id
          query = query + " and application.id in(:application_id) ";
       }
       return query;

    </Source>
</QueryScript>
```

QueryParameters

As explained in the QueryScript section above, the QueryParameters in an HQL datasource do not make up filter components in their own right but instead provide variables for substitution into the query string at the time the search is executed.

The Parameter elements within the QueryParameters for an HQL datasource is usually only specified with an argument and a property. The property is the variable name used in the query string and the argument is the argument map key in which the value to be used in the search is stored. A defaultValue or a valueScript (as described in the Filter datasource's QueryParameters section) can also be used to provide the value for the property, if desired. The Parameter's QueryScript option (which returns a QueryOptions object) cannot be used for an HQL datasource, as it does not provide a value for substitution; HQL datasources do not use a QueryOptions object in their searches.

ReportColumnConfigs

Just as with the other report types, the ReportColumnConfigs within the report's <Columns> element specify the attributes to retrieve from the query for display in the report detail grid - the “Select” portion of the query. The property attributes name the fields to retrieve. The final ReportColumnConfig - the “total” column - in the Account Group Membership Totals Report shows an example of how to include a sub query in the HQL select clause. This provides additional levels of flexibility in reflecting data on the report. A calculated field like this cannot be marked as sortable.

```
<ReportColumnConfig field="total" header="rept_app_account_grp_memb_col_members"
property="(select count(*) from IdentityEntitlement ie where ie.value = m.value and
ie.application = m.application and ie.name = m.attribute and ie.aggregationState =
&amp;apos;Connected&amp;apos;)"/>
```

Columns/ReportColumnConfig: Report Grid Presentation

The <ReportColumnConfig> elements within the <Columns> element specify which values should be returned from the query and also define how those values are presented in the report grid.

```
<Columns>

<ReportColumnConfig field="username" header="rept_uncorrelated_ids_grid_username"
property="nativeIdentity" sortable="true" />

<ReportColumnConfig field="firstName" header="rept_uncorrelated_ids_grid_firstName"
property="identity.firstname" sortable="true" />

<ReportColumnConfig field="lastName" header="rept_uncorrelated_ids_grid_lastName"
property="identity.lastname" sortable="true" />

<ReportColumnConfig field="applicationName"
header="rept_uncorrelated_ids_grid_appName" property="application.name"
sortable="true" />

 </Columns>
```

Attributes of ReportColumnConfig include:

Attribute	Usage
field	Unique name for the report column in this report
header	Column label to use in the report body; can be a string or a localizable message key
property	Object property from which the data is pulled; this value is used in the query specification
sortable	Boolean value indicating whether the report body should be sortable by this column; determines whether the column is selectable in the Sort By and Group By fields in the report specification and whether the report can be sorted by this column in preview mode
hidden	Boolean value indicating whether the column should be omitted from the report grid by default. Columns marked as hidden (hidden=”true”) appear in the left-side of the Columns list on the report template's Report Layout page, which makes them available for inclusion. However, by default they are not included on the report. Any report instances that were configured to display these fields in the report grid override this hidden attribute by including the column name in their reportColumnOrder attribute, causing the column to appear in the report output regardless of this attribute's value.
ifEmpty	Optional property to use if the value of the object property is null or empty;See Entitlement Owner Access Review Live Report's accountName field for an example: `<ReportColumnConfig field="accountName" header="rept_data_owner_col_account_name" ifEmpty="exceptionEntitlements.nativeIdentity" property="exceptionEntitlements.displayName"  sortable="true" width="110"/>`

Attribute	Usage
subQueryKey	Used for multi-valued properties to show the values as a list of comma-separated values instead of multiple rows in the report. Specifying a subQueryKey automatically renders the column as a subquery that selects the property from the dataSource objectType matching on the subsQueryKey attribute.An example exists in the Manager Access Review Live Report's tags field: `<ReportColumnConfig field="tags" header="rept_cert_col_tags" property="parent.certification.tags.name" subQueryKey="id" width="110"/>`

sortExpression	A set of fields by which the data should be sorted instead of sorting by the selected column. This attribute allows a column that is not sortable to sort the data by columns related to the selected column.
	See the following example of the permission column on the Account Group Permissions Access Review Live Report.
	
	`<ReportColumnConfig field="permissions" header="rept_cert_col_account_group_permission" property="exceptionEntitlements"sortExpression="exceptionApplication,exceptionPermissionTarget,exceptionPermissionRight" sortable="true" width="110">`
	`<RenderScript>`
	`<Source>`
	`return sailpoint.api.EntitlementDescriber.summarize(value);`
	`</Source>`
	`</RenderScript>`
	`</ReportColumnConfig>`

scriptArguments	A CSV list of additional properties to pass to a column's RenderScript; see the status field from the Policy Violation Report for an example. The renderScript then accesses these values through its scriptArgs variable (as shown in this example).
	
	`<ReportColumnConfig field="status" header="rept_viol_grid_col_status" property="status" scriptArguments="identity,policyName,constraintName,created" sortable="true" width="110">`
	`<RenderScript>`
	`<Source>`
	`import sailpoint.object.*;`
	`…`
	`String identityId = scriptArgs.get("identity").id;`
	`…`
	`</Source>`
	`</RenderScript>`

valueClass	Defines the class for the property so it can be displayed appropriately; omitted for string values

In the standard reports, the only time the “hidden” attribute is used on a ReportColumnConfig is when the column is added to the available set by an ExtendedColumnScript or ExtendedColumnRule (as described in Extended Column Script or Rule). Generally, if a column is relevant to a report, it is displayed on the report by default, though it can be removed from the detail grid by a user if they do not wish to see that data on their customized version of the report.

Note: Strings and Java constants specified in ReportColumnConfig attributes are evaluated first as message keys for automatic localization; if they do not match a defined message key, the given string value is used.

RenderScript and RenderRule

If the value returned from the query needs to be manipulated into a more user-friendly format for display on the report, this can be accomplished with a RenderScript or RenderRule. A RenderRule is used to encapsulate the beanshell into a reusable rule - useful when the same manipulation might apply to several reports. A RenderScript specifies the beanshell inline within a <Source> element. The column's property attribute is passed into the script in the variable “value”.

This example RenderScript (taken from the Revocation Live Report) displays a different localized message key depending on whether the action.remediationCompleted flag is true or false so that the report column shows an easier-to-interpret “Status” instead of a True/False flag.

```
<ReportColumnConfig field="status"
header="rept_remediation_progress_grid_col_status"
property="action.remediationCompleted" sortable="true" width="110">

   <RenderScript>

      <Source>

         import sailpoint.tools.Message;

         import sailpoint.web.messages.MessageKeys;

       return value == true ? Message.localize(MessageKeys.WORK_ITEM_STATE_FINISHED)
: Message.localize(MessageKeys.WORK_ITEM_STATE_OPEN);

      </Source>

         </RenderScript>

</ReportColumnConfig>
```

A RenderRule would be specified like this:

Rendered columns are sorted by the property attribute, not by the displayed value, so the order of rows might not appear alphabetical by the display value. At a minimum, sorting by the column groups all of the rows with the same column value together. Some properties might not be sortable, such as a property that is an object. These columns should be marked as sortable="false" even though the displayed value might seem sortable. Alternatively, a sortExpression can be specified to drive data sorting for these columns.

```
<ReportColumnConfig field="status"
header="rept_remediation_progress_grid_col_status"
property="action.remediationCompleted" sortable="true" width="110">

   <RenderRule>

     <Reference class="sailpoint.object.Rule" id="4028460238ed9b8e0138ed9bf61300de"
name="Status Message RenderRule"/>

   </RenderRule>

</ReportColumnConfig>
```

Initialization Script or Rule

The initialization script and rule allow the report developer to customize a report to address an installation's unique reporting requirements. These scripts/rules are fairly open-ended and should generally be considered tools for expert-level report creation.

Most often, initialization scripts and rules are used to customize the forms presented to the user for filter specification. For example, several standard reports use an initialization rule to build dynamic forms to present all of the installation's configured Identity attributes - both standard and extended - as filter options on some forms. Another form customization usage might be to change the set of filters available based on other filter selections; for example, a report might present a “privileged” account filter option only when the application selected for the “Application” filter has privileged accounts.

An InitializationScript is specified inline within a <Source> element:

```
<InitializationScript>
   <Source>
     import sailpoint.object.*;
     import sailpoint.reporting.ReportingLibrary;
     … (initialization code goes here; see rule example below)
   </Source></InitializationScript>
```

An InitializationRule is specified as a rule reference with the code encapsulated in the named rule:

```
<InitializationRule>
    <Reference class="sailpoint.object.Rule" id="4028460238ed9b8e0138ed9bf6130000"
name="Identity Report Form Customizer"/>
</InitializationRule>
```

The rule shown below is used in several of the standard reports (such as the User Detail Report and Identity Roles Report) to customize a form based on the standard and extended Identity attributes configured for the installation. Similar rules exist to create custom forms for other reports.

```
<Rule language="beanshell" type="ReportCustomizer" name="Identity Report Form
Customizer">
 <Description>
    This rule populates a form with fields for the standard and extended identity
attributes.
  </Description>
  <Signature returnType="Map">
    <Inputs>
      <Argument name="locale">
        <Description>
          The current user's locale
        </Description>
      </Argument>
      <Argument name="report">
        <Description>
```

```
          The base report
        </Description>
      </Argument>
    </Inputs>
    <Returns>

    </Returns>
  </Signature>
  <Source>
     <![CDATA[
     import sailpoint.object.*;
     import sailpoint.reporting.ReportingLibrary;

     ObjectConfig identityConfig = ObjectConfig.getObjectConfig(Identity.class);
     // Add standard attributes to the form

     List standardAttributes = new ArrayList();

standardAttributes.add(identityConfig.getObjectAttributeMap().get("firstname"));
     standardAttributes.add(identityConfig.getObjectAttributeMap().get("lastname"));

standardAttributes.add(identityConfig.getObjectAttributeMap().get("displayName"));
     standardAttributes.add(identityConfig.getObjectAttributeMap().get("email"));
     standardAttributes.add(identityConfig.getObjectAttributeMap().get("manager"));
     standardAttributes.add(identityConfig.getObjectAttributeMap().get("inactive"));

     ReportingLibrary.addAttributes(context, report, Identity.class,
standardAttributes, null, "Identity Attributes", locale);

     // add extended attributes to the form (multi-valued and regular)

     List extendedAttrs = new ArrayList();
     for(ObjectAttribute att : identityConfig.getSearchableAttributes()){
       if (!att.isStandard())
         extendedAttrs.add(att);
     }

     for(ObjectAttribute att : identityConfig.getMultiAttributeList()){
```

```
            extendedAttrs.add(att);
        }

        ReportingLibrary.addAttributes(context, report, Identity.class, extendedAttrs,
null, "Identity Extended Attributes", locale);

    ]]>
</Source>
</Rule>
```

The methods in the ReportingLibrary (like the one used in this example rule) are documented in the IdentityIQ Javadocs. The addAttributes method, for example, does the following:

1. Determines the form page where the attributes should be displayed (selects by section name, creates a new page based on the section name if not found, or selects the first page after Standard Properties if no section is specified)
2. Adds each attribute as an extended argument to the LiveReport object
3. Adds each attribute to the datasource QueryParameters list as a Parameter
4. Defines a Field object for each attribute and adds it to the Section

These methods can be used in custom report development, but note that it is possible that they could change in future versions of IdentityIQ, requiring those reports that rely on them to be revisited and modified. Alternatively, the code to add the attributes to the query parameters and form fields list can be explicitly written by the datasource developer.

Note: When an initialization script/rule is in place, if any of the functionality depends on the value of a specific form field, that field must be specified with the postBack attribute set to true (postBack= "true"); this causes the form to submit and reload when that value changes, and causes the initialization rule or script to execute again, picking up the new value for the field.

Signature Extended Arguments

When the initialization script or rule adds new fields to a report form, the values must be saved for any report instance for which they were specified, and they must be passed to the report at runtime to be used as report filters. This is done by adding them as extended arguments in the report definition; from there, they are automatically stored in the report instance's argument map when the report instance's TaskDefinition is saved. Even though these arguments do not exist in the report template's signature, they are generated at runtime by the initialization script and the values from the template's argument map are applied for the report's execution.

Note: This only works for these initialization-generated attributes; all static form fields must be explicitly specified in the report template's signature for them to be used in the report generation. Attributes that are included in the report instance's attribute map that do not exist in the report signature and are not generated by the initialization script or rule is not applied to the report as filters at runtime.

Extended Column Script or Rule

An extended column script or rule can be used to add additional columns to a form based on other attributes selected. For example, the script shown below adds application-attribute columns to the set of available columns based on the application selected on the form (for example, if an application has a “privileged” or “service”

account attribute, these can be optionally included in the report output when that application is selected as a filter for the report while they would not be available if a different application that did not have these attributes were selected). The extendedColumnScript or Rule should return a list of ReportColumnConfig objects; these are automatically added to the Columns list as “hidden” columns - available for inclusion on the report but not included in the report detail grid by default.

This script comes from the User Account Attributes Report and is used to add columns to the report output based on which application is selected as a filter for the report.

```
<ExtendedColumnScript>
   <Source>

   import java.util.*;
   import sailpoint.reporting.*;
   import sailpoint.object.*;

   List newCols = new ArrayList();
   Map formValues = form.getFieldValues();
  if (formValues != null &amp;amp;&amp;amp; formValues.containsKey("application") &amp;amp;&amp;amp;
formValues.get("application") != null){
       newCols = ReportingLibrary.createApplicationAttributeColumns(context,
formValues.get("application"));
   }

   return newCols;

   </Source>
</ExtendedColumnScript>
```

An ExtendedColumnRule is specified as a rule reference with the code encapsulated in the named rule:

```
<ExtendedColumnRule>
    <Reference class="sailpoint.object.Rule" id="4028460238ed9b8e0138ed9bf6130000"
name="Application Extended Column Rule"/>
</ExtendedColumnRule>
```

Note: When an extended column script/rule is in place, the field on which its functionality depends must be specified with the postBack attribute set to true (postBack= "true"); this causes the form to submit and reload when that filter field's data value changes, causing the ExtendedColumnRule to fire and detect the required condition for displaying the columns.

When columns are added to the report as a result of this rule, they first appear as “hidden” columns - available for inclusion in the report output but not selected for it. While they are still hidden, they are not saved in the report instance's XML but are regenerated as hidden columns by this rule every time the report specification is edited. Once the user adds the columns to the report detail's column list, the columns are saved in the customized report instance's attributes map in the ReportColumnOrder element, prefixed with the associated application's ID; if the report is later edited to reference a different application, these columns are automatically deleted from the report.

Validation Script or Rule

A validation rule or script is used to validate the data entered on a report form. For example, if a value is required for a specific filter for the report to run, that field can be validated as being non-null by a ValidationRule or ValidationScript.

A Validation script contains the code inline, wrapped in a <Source> element.

```
<ValidationScript>
    <Source>
        import java.util.*;
        import sailpoint.reporting.*;
        import sailpoint.object.*;
        List messages = new ArrayList();
        … (validation code goes here - see rule example below)
        return messages;
    </Source>
</ValidationScript>
```

A validation Rule is called by reference:

```
<ValidationRule>
    <Reference class="sailpoint.object.Rule" id="4028460238ed9b8e0138ed9bf61300ff"
name="Privileged Access Report Validation Rule"/>
</ValidationRule>
```

This validation rule checks the field on the Priviledged Account Attributes form for a null (or empty) value; the report requires that a value be specified for this field, so an error message is displayed and the report does not run if this field does not pass this validation. A validation script or rule returns a list of messages; if the form passes validation, this list should be empty.

```
<Rule language="beanshell" type="ReportValidator" name="Privileged Access Report
Validation Rule">
   <Description>
      This rule validates the Privileged Access Report Form
    </Description>
    <Signature returnType="java.util.List">
      <Inputs>
        <Argument name="context">
          <Description>
            A sailpoint.api.SailPointContext object that can be used to query the
database if necessary.
          </Description>
        </Argument>
        <Argument name="report">
          <Description>
```

```
        The report object
      </Description>
    </Argument>
    <Argument name="form">
      <Description>
        The submitted sailpoint Form object.
      </Description>
    </Argument>
  </Inputs>
  <Returns>
    <Argument name="messages">
      <Description>
        A list of error messages.
      </Description>
    </Argument>
  </Returns>
</Signature>
<Source>
  <![CDATA[
   import java.util.*;
   import sailpoint.object.*;
   import sailpoint.tools.Message;
   List messages = new ArrayList();

   Form.Section section = form.getSection("Priviledged Account Attributes");
   boolean found = false;
   for(FormItem item : section.getItems()){
     Field field = (Field)item;
     if(field.getValue() != null && field.getValue() != "") {
       found = true;
     }
   }

   if (!found)
        messages.add(Message.localize("rept_priv_access_err_no_attr"));

   return messages;
```

```
      ]]>
  </Source>
</Rule>
```

ReportSummary: Summary Table

The <ReportSummary> element describes the summary table in the summary section of the report.

The table header is specified in the title attribute.

```
<ReportSummary title="Uncorrelated Account Details">
```

The report summary has its own datasource; it does not use the same datasource as the report detail grid. The datasource for the report summary can be specified as a script or a rule. A script is most commonly used, but the datasource beanshell can be encapsulated in a rule for reusability if desired. Both are expressed as nested elements (<DataSourceScript> or <DataSourceRule>).

The DataSourceScript or DataSourceRule for the ReportSummary is passed these parameters:

DataSourceScript or Rule Parameters	Contents/Purpose
Context	A SailPoint Context object for executing the search
reportArgs	The TaskDefinition argument/attribute map
Report	The entire LiveReport report definition
baseHql	The from and where clause used in the report detail search if the report DataSource was an HQL datasource; null if DataSource type was not HQL
baseQueryOptions	The QueryOptions (specifying the “where” clause criteria) used in the report detail search if the report DataSource was a Filter datasource; null if Datasource type was not Filter

The summary table is usually built from data retrieved through one or more database queries that are specified and executed by the script or rule through the context (SailPointContext object), passing it a QueryOptions object populated with the necessary Filter objects. The example here illustrates how this is done.

```
<ReportSummary title="Uncorrelated Account Details">
    <DataSourceScript>
       <Source>

          import java.util.*;
          import sailpoint.tools.Util;
          import java.lang.Math;
          import sailpoint.object.*;
          import sailpoint.api.ObjectUtil;

          QueryOptions ops = new QueryOptions();
          ops.addGroupBy("correlated");
```

```
            String sources = "";
            // retrieve list of apps in reportArgs argument map; add IDs to
            // filter and names to CSV list to display as summary's "Sources" value
            if (reportArgs.containsKey("correlatedApps")){
               List apps = reportArgs.getList("correlatedApps");
               if (apps != null){
                  ops.addFilter(Filter.in("links.application.id", apps));
              List appNames = ObjectUtil.convertIdsToNames(context, Application.class,
apps);
                  sources = Util.listToCsv(appNames);
               }
            }

            List fields = new ArrayList();
            fields.add("correlated");
            fields.add("count(*)");

            int correlated = 0;
            int uncorrelated = 0;

            // get counts per Identity with links on the named applications,
            // subdivided by "correlated" flag
            Iterator results = context.search(Identity.class, ops, fields);
            while(results.hasNext()){
               Object[] row = results.next();
               int count = Util.otoi(row[1]);

               // add counts to correlated or uncorrelated totals based on correlated
               // flag
               if ((Boolean)row[0]){
                  correlated += count;
               } else {
                  uncorrelated += count;
               }
            }
            // calculate percentage of accounts that are correlated
            float percent =  correlated != 0 ? (float)uncorrelated/correlated : 0;
```

```
            String percentString = ((int)Math.floor(percent * 100)) +  "%";

            // add values to hashmap; these name/value pairs are displayed in the
            // report summary through the XML's LiveReportSummaryValue elements
            Map map = new HashMap();
            map.put("sources", sources);
            map.put("correlatedIdentities", correlated);
            map.put("uncorrelatedIdentities", uncorrelated);
            map.put("totalIdentities", correlated + uncorrelated);
            map.put("percentCorrelated", percentString);

            return map;

        </Source>
    </DataSourceScript>
```

A datasource rule would be specified as a nested element (<DataSourceRule>) that contains a rule reference.

```
<DataSourceRule>
<Reference class="sailpoint.object.Rule" id="4028460238ed9b8e0138ed9beff9090f"
name="UncorrelatedAcct Report Summary Rule"/>
</DataSourceRule>
```

The datasource script or rule returns a hashMap of values that are used to populate the corresponding LiveReportSummaryValue elements (based on their name attributes) in the ReportSummary's Values list. The LiveReportSummaryValue elements each include a unique name and a label attribute. The label can be specified as a string or a localizable message key and is displayed alongside the value in the Report's summary section.

```
<ReportSummary title="Uncorrelated Account Details">
    <DataSourceScript>
       <Source>
          …
          Map map = new HashMap();
          map.put("sources", sources);
          map.put("correlatedIdentities", correlated);
          map.put("uncorrelatedIdentities", uncorrelated);
          map.put("totalIdentities", correlated + uncorrelated);
          map.put("percentCorrelated", percentString);

          return map;
       </Source>
    </DataSourceScript>
    <Values>
      <LiveReportSummaryValue label="rept_uncorrelated_ids_grid_label_auth_sources"
name="sources"/>
       <LiveReportSummaryValue label="rept_uncorrelated_ids_summary_correlated"
name="correlatedIdentities"/>
       <LiveReportSummaryValue label="rept_uncorrelated_ids_summary_uncorrelated"
name="uncorrelatedIdentities"/>
```

```
      <LiveReportSummaryValue label="rept_uncorrelated_ids_summary_total_ids"
name="totalIdentities"/>
      <LiveReportSummaryValue label="rept_uncorrelated_ids_summary_percent"
name="percentCorrelated"/>
    </Values>
</ReportSummary>
```

Chart: Report Graph

The Chart element defines the graph that is displayed in the Summary section of the report. The chart can be represented as a pie chart or as a column or line graph. The chart data is based on the dataset for the report detail grid (the DataSource element) unless a DataSourceRule or Script is specified for the chart. Commonly, the chart represents the report body's data, grouped and counted by groupings (i.e. “value” is a count, grouped by the category and/or series).

Attributes available to define the chart are listed in this table:

Attribute	Usage
title	Name to display above the chart
type	Identifies the type of chart to display (pie, column, or line)
category	Defines the X axis in line or column graphs; defines the separate sections of a pie graph
value	Defines the Y axis in line or column graphs; defines the portion (fraction) of the pie that belongs to each section in a pie graph; often a count
series	Defines the separate columns or lines on line or column graphs; ignored for pie charts
groupBy	String value of column name (or CSV list of column names) to group data by for graphing counts
sortBy	List of sort columns for data; seldom used since groupBy can specify multiple fields in a CSV list
limit	Limits the number of records examined for the graph; seldom used, unless a similar limit was imposed on the report detail data, because the graph should generally represent all of the data in the report detail
script	Used to define a datasource for the chart other than the report detail data; script contains a <Source> element with beanshell content
dataSourceRule	Used to define a datasource for the chart other than the report details data; contains a reference to a rule where the beanshell has been encapsulated
nullSeries	Label to display if the series value (x-axis) is null (for example, a null certification action status means "Open" so that should be the label for the group of certifications whose action.status is null)
nullCategory	Label to display for data group when the category value is null

In most cases, the chart is created by selecting the value, category, and series fields from the object queried by the report detail datasource using the exact same filter criteria used by that datasource. The table below describes how the chart data is retrieved by default (when a dataSourceRule or Script are not specified for the chart).

Report Detail DataSource Type	Default Chart Query
Filter	DataSource's objectType object is queried using the queryOptions built from the DataSource Query, QueryScript, and QueryParameters elements
Java	Requires the DataSource class to have implemented the getBaseQueryOptions method; this method should return the QueryOptions used in the report detail query; also requires that the objectType is specified on the DataSource; retrieves the data for the chart from the objectType object using the QueryOptions returned by getBaseQueryOptions()
HQL	Uses the same query string employed by the report detail query, which specifies both the From and the Where clauses, to retrieve the chart's data

Standard Chart Examples

This example pie chart from the Uncorrelated Accounts Report examines the uncorrelated account data pulled from Links, groups it by application.id and counts the number of uncorrelated accounts on each application. The graph represents the number of uncorrelated accounts from each application (in this case, one uncorrelated account was found per application).

```
<Chart category="application.name" groupBy="application.id"
title="rept_uncorrelated_ids_chart_title" type="pie" value="count(*)"/>
```

The Manager Access Review Report contains an example of a Column graph that shows the count of certification items that are open, approved, or revoked (reflected in the action.status attribute), separated by roles and additional entitlements if applicable (reflected in the type attribute).

```
<Chart category="type" groupBy="action.status,type" nullSeries="cert_action_open"
series="action.status" title="rept_cert_chart_title" type="column"
value="count(*)"/>
```

Chart Script and DataSourceRule

The script and dataSourceRule elements can provide the report writer more flexibility in creating charts based on any data. However, these are generally used only in rare cases in custom reports. The standard reports do not use a Script or DataSourceRule for their charts.

When either of these is used, the beanshell within them is provided the following parameters:

DataSourceRule or Script Parameters	Contents/Purpose
context	A SailPoint Context for executing the search
args	The TaskDefinition Arguments map
report	The entire LiveReport report definition
baseHql	The from and where clause used in the report detail search if the report DataSource was an HQL datasource; null if DataSource type was not HQL
baseQueryOptions	The QueryOptions (specifying the “where” clause criteria) used in the report detail search if the report DataSource was a Filter datasource; null if Datasource type was not Filter

The code must return a list of maps (List<Map<String, Object>>) with each map representing the value, category, and series for each component of the chart. If there is no series, the “series” should be recorded as empty string ("") rather than null.

For example: {[("value","23"), ("category","ADAM"),("series","")],

[("value","5"), ("category","Financials"),("series","")],

[("value","12"), ("category","PeopleSoft"),("series","")]}

The rule or script can add onto the existing criteria from the report detail's DataSource by modifying the baseHql or baseQueryOptions or it can build the chart data with completely independent criteria. The beanshell is responsible for specifying the desired columns and search criteria, executing the database search to retrieve the data, formatting the data into the list of maps, and returning that list.

Report Forms

The layouts and contents of report-specific form pages are specified within a Form object, referenced by the report XML in a <ReportForm> element, nested within the <LiveReport> report definition. The Form object must be created and imported into IdentityIQ separately and is referenced by name.

Each <Section> defines a separate page on the Edit Report window. The page name, shown in the Sections list and at the top of the form, is specified as the Section's label attribute.

```
   <Form name="Uncorrelated Accounts Report Custom Fields">
      <Section label="Uncorrelated Accounts Parameters" name="customProperties">
<Field displayName="report_input_correlated_apps" filterString="logical==false
&amp;&amp; authoritative==false"
helpKey="rept_input_uncorrelated_ident_report_correlated_apps"
name="correlatedApps" type="Application" value="ref:correlatedApps"/>
      </Section>
    </Form>
```

Reports with large numbers of available parameters often include multiple report-specific-parameter pages, specified as multiple Sections in the Form XML, to group the parameters by category.

```
 <Form name="Identity Report Options Form Skeleton">
    <Section columns="2" label="rept_priv_access_section_priv_account_attrs"
name="Priviledged Account Attributes">
       <Attributes>
          <Map>
            <entry key="subtitle" value="rept_priv_access_section_instructions"/>
          </Map>
       </Attributes>
     </Section>
   <Section columns="2" label="rept_priv_access_section_account_props" name="Account
Properties">
        <Field columnSpan="1" displayName="rept_identity_roles_field_app"
helpKey="rept_identity_roles_helpN_app" multi="true" name="applications"
type="Application" value="ref:applications"/>
    </Section>
    <Section columns="2" label="rept_priv_access_section_identity_props"
name="Identity Properties"/>
    <Section columns="2" label="rept_priv_access_section_identity_extended_props"
name="Identity Extended Properties"/>
</Form>
```

Note: Several of the sections in this example XML do not contain any Field definitions. This is because this report uses an initialization rule to create the form fields for those sections based on system data. See Initialization Script or Rule for more information on these dynamic forms.

These are the attributes that can be specified for the Section element.

Section Attribute	Usage
label	The label for the form; can be a string or a localizable message key
name	Name for the section; must be unique per form; used programmatically but not displayed on the UI Edit Report window
Columns	Used to specify the number of columns in which fields should be displayed; fields are displayed in the order they are listed within the section, with one field added to each column in a repeating pattern; for example, in a 2-column layout, 5 fields would be displayed like this: Field 1 Field 2 Field 3 Field 4 Field 5 This attribute can be omitted for a one-column display.

Fields on each form are specified as nested <Field> elements within each <Section>. Important report-form attributes on the Field element are described below.

```
<Field displayName="report_input_correlated_apps"
filterString="logical==false &amp;&amp; authoritative==false"
helpKey="rept_input_uncorrelated_ident_report_correlated_apps"
name="correlatedApps" type="Application" value="ref:correlatedApps"/>
```

Field Attribute	Usage
displayName	The label for the field. Can be a string or a localizable message key.
helpKey	The tool tip for the field. Can be a string or a localizable message key.
name	Name for the field and must be unique per form.
type	Field type. If this entry is an object, it is automatically created as a suggest, allowing the user to select from the system's existing objects of that type.
filterString	A filter string that restricts the set of objects presented to the user for selection. Only applies to objects that are presented as suggest boxes.
value	A reference to the XML input parameter from which to retrieve the starting / default value for the field. This is how the saved values in customized report instances are populated on the form when those instances are viewed in the Edit Report window. Input parameters to the TaskDefinition XML are specified in its signature, as described in Report Signature: Passing Data from Saved Report Instances below.
postBack	A flag indicating if the form should be submitted when the field value changes.This causes the form to be reloaded and any initialization actions to be performed. This flag is important if an initializationScript or Rule or an ExtendedColumnScript or Rule needs to run to add or remove fields on the form or columns on the report based on this field value.

Field Attribute	Usage

AllowedValuesDefinition	This is specified as a nested element to provide a list of values from which the form user can select. See the User Activity Report for an example (excerpted below).
	
	`<Field columnSpan="1" displayName="label_action" helpKey="rept_input_app_activity_report_action" multi="true" name="action" type="string" value="ref:action">`
	`<AllowedValuesDefinition>`
	`<Script>`
	`<Source>`
	`import sailpoint.object.*;`
	
	`List items = new ArrayList();`
	`for(ApplicationActivity.Action action : ApplicationActivity.Action.values()) {`
	`List l2 = new ArrayList();`
	`l2.add(action.toString());`
	`l2.add(action.getMessageKey());`
	`items.add(l2);`
	`}`
	`return items;`
	`</Source>`
	`</Script>`
	`</AllowedValuesDefinition>`

Custom report forms are presented to users in the Edit Report window, along with the Standard Properties page and the Report Layout page. The Standard Properties and Report Layout pages are standard components of the report architecture and are automatically presented for any standard or custom report that implements the LiveReport architecture, whether or not the report references a custom form for report-specific parameters. The Standard Properties page is always presented first and the Report Layout page is always last; report-specific form pages are inserted between these two on the Edit Report window.

Chapter 27: Reports DataSource Example

The following sample report uses a Java data source. This sample displays Identities' name, display name, and manager status and can be filtered by manager (the manager to whom the Identities report) and application (application(s) on which the Identities have accounts). Both of these filters are multi-selectable.

```
<TaskDefinition name="Sample Report"
executor="sailpoint.reporting.LiveReportExecutor"
  subType="Identity Reports" resultAction="Rename"
  progressMode="Percentage" template="true" type="LiveReport">
   <Description>Sample report</Description>
   <RequiredRights>
       <Reference class="sailpoint.object.SPRight"
         name="FullAccessBusinessRoleMembershipReport"/>
   </RequiredRights>
   <Attributes>
      <Map>
        <entry key="report">
           <value>
             <LiveReport title="Manager Status Report">
               <DataSource type="Java"
                 dataSourceClass="sailpoint.reporting.datasource.SampleDataSource"
                 defaultSort="name">
                  <QueryParameters>
                   <Parameter argument="applications"
                     property="links.application.id"/>
                   <Parameter argument="managers" property="manager.id"/>
                  </QueryParameters>
               </DataSource>
               <Columns>
             <ReportColumnConfig field="name" header="Identity Name" property="name"
sortable="true"/>
                 <ReportColumnConfig field="displayName" header="Display Name"
sortable="true"/>
                 <ReportColumnConfig field="managerStatus" header="Is Manager"
property="managerStatus" sortable="true"/>
               </Columns>
```

```
            </LiveReport>
          </value>
        </entry>
      </Map>
   </Attributes>
   <Signature>
       <Inputs>
          <Argument multi="true" name="applications" type="Application"/>
          <Argument multi="true" name="managers" type="Identity"/>
       </Inputs>
   </Signature>
</TaskDefinition>
```

This Java datasource, SampleDataSource.java, builds and runs the query for this report based on the filters the user specifies.

```
/* (c) Copyright 2012 SailPoint Technologies, Inc., All Rights Reserved. */
package sailpoint.reporting.datasource;

import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JRField;
import sailpoint.api.sailpointContext;
import sailpoint.object.Attributes;
import sailpoint.object.Filter;
import sailpoint.object.Identity;
import sailpoint.object.LiveReport;
import sailpoint.object.QueryOptions;
import sailpoint.object.Sort;
import sailpoint.task.Monitor;
import sailpoint.tools.GeneralException;
import sailpoint.tools.Util;

import java.util.Arrays;
import java.util.Iterator;
import java.util.List;

public class SampleDataSource implements JavaDataSource {

    private Monitor monitor;
```

```
    private sailpointContext context;
    private QueryOptions baseQueryOptions;
    private Integer startRow;
    private Integer pageSize;

    private Object[] currentRow;
    private Iterator<Object[]> iterator;

    public void initialize(sailpointContext context, LiveReport report,
Attributes<String, Object> arguments, String groupBy, List<Sort> sort) throws
GeneralException {
        this.context = context;

        baseQueryOptions = new QueryOptions();

        if (arguments.containsKey("applications")){
            List<String> applicationIds = arguments.getList("applications");
           baseQueryOptions.add(Filter.in("links.application.id", applicationIds));
        }

        if (arguments.containsKey("managers")){
            List<String> managersIds = arguments.getList("managers");
            baseQueryOptions.add(Filter.in("manager.id", managersIds));
        }

        if (sort != null){
            for(Sort sortItem : sort) {
                baseQueryOptions.addOrdering(sortItem.getField(),
sortItem.isAscending());
            }
        }

        if (groupBy != null)
            baseQueryOptions.setGroupBys(Arrays.asList(groupBy));
    }

    private void prepare() throws GeneralException{
```

```
        QueryOptions ops = new QueryOptions(baseQueryOptions);

        if (startRow != null && startRow > 0){
            ops.setFirstRow(startRow);
        }

        if (pageSize != null && pageSize > 0){
            ops.setResultLimit(pageSize);
        }

        iterator = context.search(Identity.class, ops, Arrays.asList("name",
"displayName", "managerStatus"));
    }

    public boolean next() throws JRException {

        if (iterator == null){
            try {
                prepare();
            } catch (GeneralException e) {
                throw new JRException(e);
            }
        }

        if (iterator.hasNext()){
            currentRow = iterator.next();
            return true;
        }

        return false;
    }

    public Object getFieldValue(String field) throws GeneralException {
        if ("name".equals(field)){
            return currentRow[0];
        } else if ("displayName".equals(field)){
            return currentRow[1];
        } else if ("managerStatus".equals(field)){
```

```
        return currentRow[2];
    } else {
        throw new GeneralException("Unknown column '"+field+"'");
    }
}

public void setLimit(int startRow, int pageSize) {
    this.startRow = startRow;
    this.pageSize = pageSize;
}

public int getSizeEstimate() throws GeneralException {
    return context.countObjects(Identity.class, baseQueryOptions);
}

public void close() {

}

public Object getFieldValue(JRField jrField) throws JRException {
    String name = jrField.getName();
    try {
        return getFieldValue(name);
    } catch (GeneralException e) {
        throw new JRException(e);
    }
}

public void setMonitor(Monitor monitor) {
    this.monitor = monitor;
}

public QueryOptions getBaseQueryOptions() {
    return baseQueryOptions;
}

/**
```

```
     * Unused since this is not an hql report.
     */
    public String getBaseHql() {
        return null;
    }
}
```

Groups

This section contains the following information:

Chapter 28: Group and Population User Interface

Use the Group Configuration page to work with groups and populations within your enterprise. When these items are enabled, you can track and monitor activity by membership and risk information.

To access the Group Configuration page, select **Setup -> Groups** from the navigation bar.

Note: Group management is an advance process that requires the assignment of addition IdentityIQ capabilities before these pages are displayed.

The Group Configuration page has the following tabs:

- "Group Tab" on page 460 — Groups are defined automatically by values assigned to identity attributes such as Department, Location, Manager and Organization, or are based on common entitlements within an application, not common qualities as defined within IdentityIQ.
- "Populations Tab" on page 461 — Populations are query based groups created from the results of searches run from the Identity Search page. Searches that result in interesting populations of identities can be saved as populations for reuse. Because population membership is based entirely on identity search parameters, members do not have to share the same identity of account group membership.
- "Workgroups Tab" on page 463 — Workgroups enable the assignment of object ownership, certification, revocations and work items to pre-defined lists of identities. You can also assign capabilities and scope to these groups of identities so that you do not have to assign the same scopes and capabilities to each individual member of the group.

Group Examples

Groups Associated with Identity Attributes

Groups associated with identity attribute values are defined by the values assigned to those attributes. For example, the Location identity attribute might have a value for each city in which your enterprise has an office, such as Austin, New_York, and London. In that case, there are three groups created, Austin, New_York, and London, one for each value of the attribute, and each containing the identities that have the corresponding value assigned to Department.

Groups Based on Common Entitlements

Groups based on common entitlements within an application are defined by shared access and are listed under role. An entitlement is either a specific value for an account attribute or a permission. A role is a collection of entitlements that enable an identity to perform certain operations within your enterprise. When the role group attribute is created and enabled, each role becomes a group consisting of all identities that share the entitlements that make up that role. Identities assigned entitlements that do not combine to match the criteria of a role are

assigned to the group No role. Th Global group contains all identities.

Group Tab

The Groups table contains a list of the high-level containers, or group factories, that contain the actual groups used within IdentityIQ. Each group factory is associated with either an identity attribute or an entitlement within an application. These group factories are not groups themselves, but are used to define, maintain, and enable groups.

The Group tab contains the following information:

Table 100—Groups Tab Column Descriptions

Column Name	Description
Name	The name assigned to the group factory when it was created.
Attribute	The attribute used to define the groups within the group factory.
Description	Description of the group factory or the groups contained within.
Status	The status of the groups within the group factory, enable (check mark) or disabled (exclamation mark). This status controls all of the groups contained within this group factory.

Click on a group factory or right-click and select edit to display the Edit Group page. The Edit Group page contains the group factory information from the table and a list of the groups associated with the group factory. For example, for a Manager group factory the table contains a row for every value assigned to the manager attribute in IdentityIQ. See “Edit Group Page” on page 460.

To create a new group, click **Create New Group** to open the Edit Group page.

To delete a group factory, right-click and select **Delete**.

Edit Group Page

This page is used to enable or disable all of the groups contained within a group factory, recreate a group factory that has been deleted, and view the groups that make up a group factory. Creating multiple group factories of the same type produces identical results when a task is run that updates group information. For example, if you create three (3) group factories, X, Y, and Z and specify the Department attribute for each, you receive identical results for all three group factories when you run a task that updates group information.

The Edit Group page contains the following information:

Table 101—Edit Groups Column Descriptions

Column Name	Description
Group Information:	
Name	The name assigned to the group factory when it was created.
Group Attribute	The attribute used to define the groups within the group factory.
Description	Description of the group factory or the groups contained within.

Table 101—Edit Groups Column Descriptions

Column Name	Description
Enabled/Disabled	The status of the groups within the group factory, **Enabled** or **Disabled**. This status controls all of the groups contained within this group factory. **Enable** — the groups are active and available for use and activity searching. **Disabled** — the groups exist, but are not included in statistical tracking or available on the search pages.
Scope	The scope for this group factory. If scope is assigned, only the users that control the designated scope can see this group factory in select lists on pages such as the Certification Schedule or Search pages. The sub-groups associated with this application are visible to a user with any or no controlled scope. Depending on configuration settings, objects with no scope assigned might be visible to all users with the correct capabilities.
Sub-Group Information: **Note: This information is not displayed until group aggregation is performed by a task. See "Tasks" on page 283.**	
Name	The name of the group, or the value assigned to the specified attribute.
Member Count	The number of identities matching the group criteria.
Policy Violations	The total number of policy violations for members of the group.
Composite Score	The average composite risk scores of each member of the group.
Owner	The owner of the sub-group, if one is assigned.
Last Updated	The last time a task was run that updated the group information.

Populations Tab

The Populations tab contains a list of populations that either you created from identity searches or that were created by other users and defined as public. Populations are query based groups created from the results of searches run from the Identity Search page. Searches that result in interesting populations of identities can be saved as populations for reuse. Members of a population might not share any of the same identity attributes or account group membership. Population membership is based entirely on identity search parameters.

The Populations tab contains the following information:

Table 102—Populations Tab Column Descriptions

Column Name	Description
Name	The name assigned to the population when it was created.
Description	Description of the population.

Table 102—Populations Tab Column Descriptions

Column Name	Description
Visibility	If the population is Private or Public. **Private** — only visible to the user that created them. **Public** — available to any user with access to pages on which they are used and control of the correct scope, if scoping is active.
Owner	The name of the population owner, if one is assigned.
Status	The status of the population, enable (check mark) or disabled (exclamation mark). **Enable** — the populations are active and available for use in activity searching. **Disabled** — the populations exist, but are not included in statistical tracking or available on the search pages.

Click on a population or right-click and select edit to display the Edit Population page. The Edit Population page contains the population information and a list of associated identities. See “Edit Population Page” on page 462.

To delete a population, right-click and select **Delete**.

Edit Population Page

This page is used to edit population information, enable or disable populations, mark populations as private or public, set the scope for the population, and view the identities that make up a population.

Note: Any user that has access to a public population can make changes on that population.

Note: If you mark a public population as private, and you are not the creator of that population, you can no longer see that population.

Click on an identity to display the View Identity page for that user.

That Edit Population page contains the following information:

Table 103—Edit Populations Column Descriptions

Column Name	Description
Group Information:	
Name	The name assigned to the population when it was created.
Description	Description of the population.
Private	Select or clear the check-box to specify if the population is private or not private. **Private** — only visible to the user that created it from the search results page. **Not Private** — available to any user with access to pages on which they are used and control of the correct scope, if scoping is active.
Enabled/Disabled	Select or clear the check-box to specify if the population enabled or not enabled. **Enable** — the populations are active and available for use inactivity searching. **Not Enabled** — the populations exist, but are not included in statistical tracking or available on the search pages.

Table 103—Edit Populations Column Descriptions

Column Name	Description
Scope	The scope for this population. If scope is assigned, only the users that control the designated scope see this population in select lists on pages such as the Certification Schedule or Search pages. This scope only applies to the population, not the identities contained within.
Owner	Assign an owner for the population.
Population Information:	
Population Count	The number of identities in IdentityIQ matching the populations search criteria.
Name	The value of the accountId attribute for the identity.
First Name	The value of the firstname attribute for the identity.
Last Name	The value of the lastname attribute for the identity.
Manager	The value of the manager attribute for the identity.
Last Refresh	The date on which the identity was last refreshed.

Workgroups Tab

The Workroups tab contains a list of workgroups enable the assignment of object ownership, certification, revocations and work items to pre-defined lists of identities. In addition to grouping Identities you are also able to assign capabilities and scope to these groups of identities so that you do not have to assign the same scopes and capabilities to each individual member of the group.

The Workgroups tab contains the following information:

Table 104—Workgroups Tab Column Descriptions

Column Name	Description
Name	The name assigned to the workgroups.
Description	A short description of the workgroup.
Modified	The date and time the workgroup was last modified.

Click on a workgroup or right-click and select edit to display the Edit Workgroup page. The Edit Workgroup page contains the group information and a list of capabilities and members. See “Edit Workgroups Page” on page 463.

To create a new workgroup, click **Create Workgroup** to open the Edit Workgroup page.

To delete a workgroup from the list, right-click and select **Delete**.

Edit Workgroups Page

This page is used to edit workgroup information and view the capabilities, scope and members that make up a group.

That Edit Workgroup page contains the following information:

Table 105—Edit Account Groups Column Descriptions

Column Name	Description
Group Information:	
Name	The name assigned to the workgroup.
Owner	The owner assigned to this group.
Description	Description of the group.
Scope	The scope for this workgroup. If scope is assigned, only the users that control the designated scope can see this workgroup in select lists on pages such as the Certification Schedule or Search pages. This scope only applies to the workgroup, not the capabilities or identities contained within.
Group Email	Specify the email address assigned to this workgroup. A workgroup email address should be a distribution list. If no address is specified here, notifications are sent to each member of the group. **Note: A workgroup email account needs to be created in your email system.**
Notification Setting	Specify to whom notifications should be delivered. **Note: If you select Notify members and group email and the group email is a distribution list, the members receive the notification twice.** **Notify members and group email** - send notifications to each group member and the group email address. **Notify group email only** - send notifications to the group email address but not the individual group members. **Notify members only** - send notifications to each group member, but not the group email address. **Disable notifications** - send no notifications to this group. This restriction only applies to items assigned to the workgroup.
Rights:	
Capabilities	The SailPoint capabilities available. The capabilities currently assigned to the workgroup are highlighted on the list. **Note: Each member of the group assumes the capabilities of the group, even if different capabilities were assigned to them individually.** Use the Ctrl and Shift keys to select multiple capabilities.

Table 105—Edit Account Groups Column Descriptions

Column Name	Description
Authorized Scope	The scopes controlled by this workgroup. Scope is used to determine the objects to which the members of this group have access. Control determines access. If scoping is active, the workgroup members can only see objects that are within the scopes controlled by the group. Assign scopes to the workgroup using the suggestion field at the top of the Authorized Scopes list box. - Click the arrow to the right of the suggestion field to display a list of all scopes defined. - Enter a few letters in the suggestion field to display a list of all scopes that start with that letter string. Depending on configuration, objects with no scope assigned might be visible to all users with the correct capabilities. See “Scopes” on page 47
Can Access Assigned Scope	Select this option to enable the workgroup members to control the scope to which they are assigned. If this option is cleared, the users do not have access to objects within the scope to which the workgroup is assigned. Control determines access. If scoping is active, identities can only see objects that are within the scopes they control.
Members: The list of members of the workgroup. Use the drop-down list at the bottom of the table to select identities and the click **Add Member** to add members to the workgroup. Use the select boxes to select members and click **Remove Members** to remove members from the workgroup.	

Chapter 29: Introduction to Group Constructs

IdentityIQ offers several mechanisms for grouping Identities into sets based on shared characteristics. Each type of grouping has a distinct purpose and offers unique capabilities. This document explains the differences between them and the uses for each.

The grouping constructs discussed in this document include:

- Roles
- Workgroups
- Populations
- Groups

At a high level, these constructs are distinguished from each other as described below:

Roles: model the organizational structure, job functions, and system Entitlements; present Entitlement data in a way that is readily understood by non-technical reviewers

Workgroups: associate sets of Identities to facilitate sharing of IdentityIQ responsibilities; responsibilities can include Certification reviews and Application or Entitlement ownership, among others

Populations: query-generated lists of Identities that share a common set of attributes; used as a filter on the set of Identities included in a task, Certification, or report

Groups: sets of Identities created based on the value of a single Identity Attribute; used as a filter on the set of Identities included in a task, Certification, or report

Chapter 30: Roles

IdentityIQ's Role functionality is used to model a company's structure and business operations. Roles are designed to be highly flexible and customizable, allowing them to be used to model a wide array of business structures and functions.

By default, there are four types of Roles configured in IdentityIQ:

- **Organizational**: organize and manage the role hierarchy
- **Business**: identify job functions or titles
- **IT**: encapsulate sets of system Entitlements
- **Entitlement**: represent individual system Entitlements

Custom Role types can be created to model a structure that doesn't easily fit into the IdentityIQ default model. In addition, the existing Role types can be configured to function differently from their default behavior to meet each organization's business needs.

A separate document has been created to explore all of these Role types. It offers some examples of how Roles can be used to model a business and to facilitate Identity management. Refer to the Role Management in IdentityIQ document for more information on Roles.

Chapter 31: Workgroups

A Workgroups is a grouping of Identities that can be assigned activities within IdentityIQ as if the group were a single Identity. While a Role describes and manages activities and access outside of IdentityIQ, Workgroups specifically relate to activities and access within IdentityIQ.

Using Workgroups

Workgroups are primarily used in two ways: for allowing Identities to share responsibilities and for managing IdentityIQ Access for groups of Identities as a unit.

Responsibility Sharing

IdentityIQ allows activities or responsibilities to be assigned to Workgroups just as they can be assigned to an Identity. Grouping Identities into Workgroups makes it possible for multiple people to share responsibility for certain functions, which can help with managing activities that must be performed by someone but do not necessarily need to be owned or performed by a specific person.

The following activities are assignable to a workgroup:

- Application Owner
- Application Revoker
- Certification Owner
- Role Owner
- Entitlement Owner
- Account Group Owner
- Policy Owner
- Policy Violation Owner
- Policy Violation Observers

Consider, for example, a large-application System Administration team made up of 5 people who share responsibility for managing access and permissions for many users. These shared responsibilities could be divided among the team members by setting different team members as the Application Owner, Revoker, Certification Owner, etc. If, however, all team members are qualified and empowered to address any of these requests, it could be substantially more efficient to create a Workgroup for this team and assign these activities to the Workgroup, rather than assigning ownership to any one of the team members. Access/Revocation/Certification requests can then be funneled to the group to be processed by the first available team member.

IdentityIQ Access Management

System capabilities within IdentityIQ can also be managed for an entire population of Identities by assigning them to the same Workgroup. For example, if a help desk team all needs the same IdentityIQ capabilities they can be assigned to a Workgroup and their access can be managed through the Workgroup instead of on each individual Identity. Capabilities set on individual Identities remain in effect in addition to the capabilities assigned to the Workgroup. If one person in the group, such as the team lead, requires additional IdentityIQ capabilities, the unique permissions for that person can be managed on their Identity without affecting the other group members' access.

Workgroup Creation

Workgroups are created on the **Define** -> **Groups** -> **Workgroups** tab by clicking **Create Workgroup**.

A **Group Email** address can be specified, and emails can be configured to send to the group and/or the individual members. The group's common **Capabilities** and **Scopes** are specified in the **Rights** section, and Identities are added to the workgroup in the **Members** section at the bottom of the **Edit Workgroups** window.

Chapter 32: Populations and Groups

Populations and Groups are two more grouping constructs in IdentityIQ. Both of these are used subdivide identity sets within IdentityIQ for reporting and internal system tasks. Populations and Groups are created through different mechanisms, but they are used in similar ways throughout the IdentityIQ application.

Populations are sets of Identities generated from queries on the Advanced Analytics page and can be based on multiple criteria, such as North America, non-manager, accounting department employees. Any Identity Attribute marked as **Searchable** can be used as a Population criterion. The result set for the query (the Population) is a single set of Identities who share a common set of properties.

Groups are sets of Identities that share a common value for a specific Identity Attribute. Only Identity Attributes marked as **Group Factory** attributes can be used as a group filter attribute in the creation of Groups. Groups are usually created in sets. For example, generating groups based on the **Attribute Region** can produce five groups: North America, Western Europe, Asia, South America, Eastern Europe.

When a Population or Group is saved, the query criteria to generate it is recorded and not the set of Identities that matched the criteria at that moment. Each time the Population or Group is used, the query is run and the current set of Identities matching the query criteria is retrieved and applied to the operation.

Creating Populations

To create Populations, you specify query criteria on the Advanced Analytics page of the IdentityIQ user interface (menu bar option and save it as a population. To access the Advanced Analytics page, from the Navigation menu bar go to **Intelligence -> Advanced Analytics**.

The UI provides two methods of specifying the criteria: the basic Identity Search and the Advanced Search windows. Identity Search allows you to specify simple filter values for Identity Attributes to define the population. With Advanced Search, you can be specify more complex search criteria, including grouping of filter criteria, choosing “and” vs. “or” relationships between criteria, and specifying search types other than “equals”.

Basic Identity Search

The default view of IdentityIQ's Advanced Analytics option is the Identity Search tab, which offers a variety of Identity Attribute for which search values can be entered. These criteria are evaluated together in an “and” relationship to select the population's members, meaning all Identities in the population will meet all search criteria specified.

In addition to basic Identity Attributes, application accounts held, detected or assigned Roles, associated Workgroups, and Risk Attributes can be used to filter Identities in a basic search.

Multi-Valued Attributes are specified separately, with the option of selecting multiple values in either “and” Orr” relationships (requiring the Identity to have all of the values assigned or any one of them, respectively).

The fields selected in the **Fields to Display** list are shown on the search results window. Once the search is saved as a population, however, the display fields do not really matter; when used in other parts of IdentityIQ as a processing filter, populations return the Identities that match the criteria, not just the specified display fields.

Once the parameters have been specified, click **Run Search** at the bottom of the window to execute the search based on the specified criteria.

To create the Population, click **Save Identities as Population** from the **Result Options** list.

Advanced Search

The Advanced Search options, accessible by clicking **Advanced Search** on the Identity Search tab, provide more flexibility in specifying search criteria.

Individual filters are specified by selecting a field, choosing a search type (=, <>, like, null, not null) and entering a valuer field is suppressed for null/not null options). The “like” options can be further narrowed by whether the field value should start with, end with, contain anywhere, or be an exact match for the Value specified. Then, the filters can be connected through “and” or “or” relationships in any fashion, including grouping and nesting of criteria.

For example, to include in the population all Identities in the Austin location that are either Managers or in the Accounting department, you can specify the search criteria as shown here:

To edit the filter source directly, click [**view /edit filter source**]. This provides even more flexibility in specifying filter criteria in ways that might not be available through the user interface.

Once filters are modified through the filter source and saved, the standard representation of the search criteria is updated in the user interface to reflect the changes. When the variables selected are not ones the system is able to display in its reader-friendly format, the message The filter you have entered cannot be displayed but will be applied to your search is shown instead.

Filter Source Specification

Only persistent variables in the object model can be specified in the query filter. In general, this set matches the list of variables available through the public “get” and “set” methods shown in the IdentityIQ Javadocs that ship with the product. The variable names to specify match the method names without the “get”/”set” prefix. For example, the “**first name**” variable is accessible through the getFirstname() method, so the variable for the filter string would be firstname (the first letter of the variable name is always lowercase; the rest matches the camel case of the method name).

Fields within objects contained within the Identity object can be queried with the object.attribute syntax (for example, bundles.name or links.application.name). Multi-valued Identity Attributes can be accessed through the IdentityExternalAttribute object, and multi-valued Account Attributes can be queried through the LinkExternalAttribute object using syntax that mirrors the following:

```
IdentityExternalAttribute.collectionCondition("((id.join(IdentityExternalAttribute.
objectId) && IdentityExternalAttribute.attributeName i== \"IdentityAttributeName\"
&& IdentityExternalAttribute.value.startsWith(\"attributevalue\")))")
```

or

```
LinkExternalAttribute.collectionCondition("((links.id.join(LinkExternalAttribute.ob
jectId) && LinkExternalAttribute.attributeName i== \"AccountAttributeName\" &&
LinkExternalAttribute.value.startsWith(\"attributevalue\")))")
```

The table below indicates the syntax required to add filters of various data types to the filter source.

Table 106—Filter Syntax

Field Data Type	Structure	Example
String	“value”	department == “Accounting”
Numeric	value	location <= 10
Boolean	value	managerStatus == true

Table 106—Filter Syntax

Field Data Type	Structure	Example
Date	DATE$[long value of time - milliseconds since Jan 1, 1970]	lastLogin > DATE$1318884600000
Char (single character)	'value'	middleInitial == 'D'
Float	Value (floating point literal)	average < 250.144
Enumeration	EnumName.EnumValue	Type == CertificationItem.Type.Exception

Note: The IdentityIQ object model currently has no persistent Char or Float fields, and it is rare for Enumerations to be queried through these pages. Those three data types are included here primarily as interesting information.

The filter compiler can interpret the following operators and expressions:

Table 107—Operators and Expressions

Conditional Operators	&&, ||
Parentheses groupings and function references	(,), startsWith, startsWithIgnoreCase, endsWith, endsWithIgnoreCase, contains, containsIgnoreCase, in, inIgnoreCase, join, isNull, notNull, isEmpty, collectionCondition, subquery
Property Operators	==, !=, <=, >=, >, <, i==, i!=, i>=, i<=, i>, i< (i means ignore case)

Creating Groups

Three types of objects are involved in the creation of Groups:

- **Group Factory**: store the definition of which Attribute should be used for grouping and what to call the associated set of Groups
- **GroupDefinition**: contain the actual filter used to match identities to the group. Populations are also stored as GroupDefinition objects. Running the 'Refresh Groups' task scans the GroupFactories which in turn creates GroupsDefinitions for the values of the factory attribute.
- **GroupIndex**: also referred to as group scorecard; maintain statistics about a particular GroupDefiniiton (number of members, policy violations, composite risk score).

Groups are created on the Group Configuration window (menu option **Define** -> **Groups**) by clicking **Create New Group** on the Groups tab.

The **Name** field specifies what the GroupFactory will be called. A single **Group Attribute** is selected to define the selection criterion for membership in each of the created Groups; only Attributes that have been defined as "Group Factory" attributes can be used in creating Groups, so the selection list only includes those Attributes. When the Group is saved, a GroupDefinition is created for each value of that Attribute in the current set of Identities.

Identities' Group membership is determined at the time the Group is applied to an activity in IdentityIQ (such as when a Certification or a Task runs) based on the GroupDefinition filter. If an Identity's Group Attribute value changes, its new value is used for Group-based actions from the moment of the change. However, the statistics tracked in the GroupIndex, as well as the list of GroupDefinitions themselves, are only updated when an Identity Refresh task runs for which the **Refresh the group scorecards** option is selected. This means that if a new value

is added for the Group Attribute (for example., in the Manager Group example above, a new manager is hired and assigned for a set of Identities), the new Group corresponding to that value will not be created or applied to any system activity based on the Group Factory until the refresh task runs.

Group and Population Definitions in XML

The XML representation of the Group Definition (filters defining a Population or Group) can be viewed and edited from the IdentityIQ debug pages by selecting **GroupDefinition**, clicking **List**, and then selecting the desired population or group name from the list.

The XML can be saved to create deployment artifacts that can be used for reimporting the definitions into a new environment. It can also allow one definition to be used as a template for creating others that can be imported into IdentityIQ instead of having to be generated through the user interface (for example, modifying the definition shown above, created for Location = Austin, to create an identical one for Location = Quebec).

Using Populations and Groups

Groups and Populations are used to apply actions in IdentityIQ to specific sets of Identities, rather than to every Identity in the system. They can be used in these areas:

- Filters on Identity Refresh and Policy Scan Tasks
- Advanced Certification selection criteria
- Advanced Analytics: Access Review and Activity Search query criteria
- Report Filters
- IT Role Mining Filters

Using Populations or Groups as filters for these activities makes it possible to run these queries and processes for select sets of Identities. By selecting only the desired Identities and omitting the rest from the process, these filters allow for targeted data analysis and more efficient system processing.

As Task Filters

Populations and Groups can be specified as filters on Identity Refresh and Policy Scan tasks. These include all custom tasks created based on the Identity Refresh and Policy Scan task templates.

In Certifications

Advanced Certifications generate Access Reviews for specific Populations or Groups. The desired Population(s) or Group(s) must be specified in the **What to Certify** section of the Certification's **Basic** page.

As Advanced Analytic Criteria

Two of the Advanced Analytics query types allow Populations or Groups to be specified as search criteria: Access Review Search and Activity Search. The Access Review Search filter results in inclusion of only Access Reviews that were generated for the specified Population or Group. The Activity Search only allows Populations (not Groups) to be used as a filter limits the returned set of Identities to ones that are part of the selected Population.

As Report Filters

Populations and Groups can be used as filters on several of the pre-configured IdentityIQ reports, including the Advanced Access Review Report, the Identity Effective Access Report, the Identity Risk Report, and the Identity Role Report. The Advanced Access Review Report filter means the report is run only for Access Reviews created for the selected Population or Group. For the other reports, the filter allows only Identities that are part of the selected Population or Group to be included on the report.

In IT Role Mining

Populations can be specified as Identity selection criteria for IT Role Mining activities. Groups cannot be specified here.

Password Management

This section contains the following information:

Chapter 33: Introduction

IdentityIQ supports multiple login configurations, including single sign-on, pass-through authentication, and validation against IdentityIQ's internally stored passwords. Pass-through authentication and internal passwords can be managed through the IdentityIQ user interface.

IdentityIQ's internal set of passwords are governed by the IdentityIQ password policy. These internal passwords are always available as a fallback login validation for IdentityIQ, even when other authentication methods are used; either the user or an administrator can reset an internal password through IdentityIQ's change password options.

When pass-through authentication is used, IdentityIQ enables the specification of challenge questions that can enable users to reset their own forgotten passwords, once they authenticate to IdentityIQ by correctly answering those questions. New passwords entered through this forgot password feature are validated against the pass-through authentication application's password policy and are reset on that application directly.

This section is divided into three chapters, each covering key subjects related to Password Management:

Chapter 34: Application Password Management

IdentityIQ can use the Lifecycle Manager product to manage passwords across many of the applications with which it is associated. It can enforce password policies specified for the applications, which can include requirements for length, complexity, unique history, and mandatory reset.

To manage passwords across application, you must configure both IdentityIQ and the applications on which you are going to manage passwords. Password management is further governed by the capabilities of the connector in use for each application and some applications have specific configurations requirements that go beyond the basic password management requirements.

Enabling Password Management in IdentityIQ

The ability to manage passwords in other applications through IdentityIQ is controlled by the Lifecycle Manager Configuration settings for the installation. To configure Lifecycle Manager, click the gear icon and select **Lifecycle Manager Configuration**.

In Lifecycle manager, request permissions are configurable for Identities in four categories:

- **Self Service**: specifies the types of requests that can be made by an Identity on their own behalf
- **Managers**: specifies the types of requests that can be made for other Identities (and the set of Identities for which those requests can be made) when the requester is a Manager (isManagerflag = True)
- **Help Desk Personnel**: specifies the types of requests that can be made for other Identities (and the set of Identities for which those requests can be made) when the requester has the Help Desk Personnel Capability in IdentityIQ
- **All Users**: Specifies the types of requests that can be made for other Identities (and the set of Identities for which those requests can be made) by any user

The password management function can be turned on for any or all of these categories by selecting the **Manage Passwords** Lifecycle Action under each request-permission category.

Defining Special Characters Available Password Use

IdentityIQ enables you to define the special characters that can be used in passwords throughout your deployment of the product. A default set of special characters are included in the System Configuration object.

The special characters enabled for use in passwords are listed in the passwordSpecialCharacters key. To edit these items:

- Click the **Gear** icon in the navigation menu, go to the **Global Settings ->** IdentityIQ **Configuration -> Password tab -> Password Policy** area, and click **Define Character Type**.
  Or
- Go to the debug page of the IdentityIQ user interface and select object type Configuration from the drop-down menu. Select the SystemConfiguration object and edit the value for the passwordSpecialCharacters entry key. For example:

```
<entry key="passwordSpecialCharacters" value="~!@#$%^*_+-={}\\][:;?,."/>
```

Configuring Applications for Password Management

Password management is further governed by the capabilities of the connector in use for each application. Passwords can be managed through IdentityIQ for any application using a read-write connector that has the PASSWORD feature enabled; this feature is enabled when the features String attribute on the application contain the word “PASSWORD”. The application definition, including its featuresString attribute, for each application is viewable in the XML representation of the Application object (accessible from the debug pages or from the iiq console).

```
<Application connector=”sailpoint.connector.LDAPConnector” created=”1334252935835”
featuresString=”AUTHENTICATE, PROVISIONING, ENABLE, PASSWORD, MANAGER_LOOKUP,
SEARCH, ACCOUNT_ONLY_REQUEST" id="4028833636890f860136a7ac1a6c054f"
modified="1335456303423" name="ADAM Direct" profileClass=”” type=”ADAM - Direct”>
```

Note: **Not all read-write connectors have the PASSWORD feature enabled. The Connector Registry entry for each connector includes all the valid features for that connector in its featuresString attribute. Specifying PASSWORD in the featuresString of an application to which the feature does not apply does not successfully enable password management for the application. To view the Connector Registry entries from the debug pages, select Configuration from the Objects list and click List. Then click ConnectorRegistry to view the connector registry XML.**

Configuring Password Policies for an Application

Password policies specify the password requirements for an application. These can include minimum and maximum lengths for the password and requirements for its makeup (number of letters, digits, uppercase letters, lowercase letters, special characters). The policies can also restrict password choice based on matches in password history, the password dictionary, the Identity's list of attributes, and the Identity's account attributes.

A separate password policy can be defined for each application in IdentityIQ. In fact, multiple policies can be defined for each application.

Defining a Password Policy

Complete these steps to define an application's password policy:

1. Open the application definition. From the navigation menu, go to **Applications -> Application Definition ->** [select application from list or click **Add New Application** to create a new application]. Then click the **Password Policy** tab.

2. Click **Create New Policy** to create a new password policy, click a policy name in the list to edit an existing policy, or click **Add Existing Policy** to select a predefined password policy from the drop-down list, see "Policy Re-Use" on page 486.
3. Name the policy (required) and provide a brief description. Specify any required password characteristics. Most of these characteristics are self-explanatory; these few are further explained here since they might be unclear:
   - **Password history length:** specifies number of previous passwords in password history to check against for uniqueness (prevents re-use of a password over the specified number of password changes); the current password is included in the count
   - **Validate passwords against the password dictionary**: compares password to an internally-stored, implementation-specific password dictionary, ensuring that the password is not, and does not contain, any word in that dictionary (see Password Dictionary below)
   - **Validate the password against the identities list of attributes**: ensures that values stored as Identity attributes (for example, last name, department, office number, region) are not used as the password
   - **Validate the password against the identity's account attributes**: prevents values stored on the application account from being used as the password

   Note: The password history, if a Password history length value is specified, it is stored as a <PasswordHistory> element on the <Link> (account representation) within the Identity object. It is stored as a comma separated values list of encrypted passwords. The number of passwords stored is determined by the Password history length value specified. New passwords set for the account cannot match any password in the list.
4. Select an **Identity Filter** if this policy should only apply to certain sets of Identities. The default Identity filter is All, which means the policy applies to all Identities. Other options are:
   - **Match List**: specify Identity Attributes or Application Attributes/Permissions by which Identities can be matched for this policy to apply (for example, Identity Attribute: Department = Accounting)
   - **Filter**: specify a filter (as CompoundFilter XML) that can be used to identify Identities to which this policy applies
   - **Script**: specify a segment of beanshell that selects Identities that should use this policy
   - **Rule**: specify a rule (type: IdentitySelector) that returns a list of Identities to which this policy should apply
   - **Population**: apply this filter to the Identities in an existing IdentityIQ Population

   Note: The first policy defined should be the default policy that applies to all users. This policy serves as the "fallback" policy if none of the more restrictive policy Identity Filters apply to the Identity whose password is being validated. If more than one policy is specified with Identity Filter = All, only the last one created is applied in any Identity password validation. This is further explained in the Password Validation Process section.

Password Dictionary

The password dictionary is a set of words (or character strings) that have been deemed impermissible as passwords or password contents for the specific IdentityIQ installation. It is populated by importing a Dictionary XML object through the iiq console or the Import from File option under System Setup. The XML looks like this, and the prohibited words in the password dictionary are included as <DictionaryTerm> elements:

```
<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE sailpoint PUBLIC "sailpoint.dtd" "sailpoint.dtd">
```

```
<sailpoint>
   <ImportAction name='merge'>
     <Dictionary name="PasswordDictionary">
      <Terms>
        <DictionaryTerm value="password"/>
        <DictionaryTerm value="identity"/>
      </Terms>
     </Dictionary>
   </ImportAction>
</sailpoint>
```

Include the <ImportAction name='merge'> element to add new terms to the dictionary without overwriting the existing dictionary entries. Omit this element to overwrite the dictionary with a new set of terms.

If removing terms or replacing the entire dictionary, then delete the dictionary object first using the console or debug pages. The terminator will handle removing both the Dictionary and the dictionaryTerms.

Note: Terms included in this dictionary are prohibited even as any part of a password when password dictionary validation is enforced. For example, if the term "rock" were included in the password dictionary, these passwords would all be prohibited: rocketlauncher, sprocket, Sh@mrock125. Additionally, validation against the password dictionary is case insensitive, so RocKeTTe would also be prohibited in this case.

Policy Re-Use

Previously created policies (for example, ones created for one application but applicable to more than one application) can be added to an application instead of recreating the same policy over and over. For example, if super-user accounts on all applications have the same password requirements, the super-user policy could be created once and copied to all applications.

To copy an existing policy from one application to another:

1. On the **Password Policy** tab for the target application, click **Add Existing Policy.**
2. Select the desired password policy by name. The password policy characteristics are displayed for review.
3. Configure the **Identity Filter** to apply the policy to the appropriate set of Identities for this application. Filters might differ from one application to another (for example, different application attributes or permissions or different Identities attributes can designate a super-user on one application but not on another), so they do not carry over between applications in this policy sharing feature.
4. Click **Save** to save the filter on this application.

The policy now appears in the application's **Password Policies** list with a warning icon. Hovering over this icon displays the message "Be careful editing this policy, it is also used by another application." Changes made to the requirements in a shared policy affect all applications using that policy. Changes made to the Identity Filter on these shared policies only affect the individual application's use of the policy.

Password Validation Process

In many cases, the password policy for an application applies to all users, so there is only one password policy per application. Sometimes, more than one policy is created for a single application to specify different password requirements for different levels or types of user access. In the password management process, when a user's password is being changed, the policy checker scans all of the policies that apply to the identity and creates one super-policy that covers all of the restrictions for that user.

If no password policy is defined for the application, no password policy is enforced and any password entered for a password change is accepted by IdentityIQ and passed to the application to be set as the account's new password.

Application Change Password Provisioning Policy

Some applications support more than one password type. For example, Lotus Notes has three different types of passwords that need to be managed, a vault password, a file password, and an internet password. IdentityIQ can be used to configure those applications so that all password types can be managed through a change password provisioning policy.

The change password provisioning policy template is loaded when a change password request is created through Lifecycle Manager. This template is only loaded for change password. The other password management requests are not affected.

The Change Password provision policy is configured on the Provisioning Policy tab of the Application Configuration page.

Requesting a Password Change

Password changes, self-service or for others, are requested through the Manage Access QuickLink for Lifecycle Manager. When the request is submitted, it is immediately processed through a workflow, by default, the LCM Manage Passwords workflow.

By default, application password requests (forgot, expired, or change), either self-service or for others, invoke the LCM Manage Passwords workflow. This workflow's default configuration requires no application-owner or manager approvals on a password change. It creates and processes a provisioning plan that contains the requested password changes and then notifies the user by email when the change is complete.

If the change request is for an account whose application is configured with a Change Password provisioning policy, additional information is required before the change occurs. See “Application Change Password Provisioning Policy” on page 487.

Self-Service Requests

When a user wants to, and is authorized to, change their own password on an application, they must complete these steps in IdentityIQ:

1. From the **Manage Access Quicklink**, click **Change Passwords** and select **For Me**.
2. Select the application account or accounts for which the password is being changed.

Note: Hover over the help text icon () by the application name to review its password policy requirements.

3. Enter the **Current Password** for each account being updated. Enter the new password twice: once in **New Password** and once in **Confirm Password**.
   If more than one application's password is being changed at a time and the new passwords should all be identical, select **Synchronize passwords for selected accounts**. Each of the selected accounts is then prompt for the **Current Password** for that account but the New Password and **Confirm Password** boxes are displayed only once at the top of the window and apply to all applications whose passwords are being changed.
4. Click **Submit** at the bottom of the window to submit all password changes.

   Note: If the entered passwords do not match or if the password does not meet the requirements of all of the application's password policies, an error message is displayed on this window and the password values must be re-entered before the requested changes are successfully submitted.
5. A summary of the requested changes is displayed on the next window. Review this summary and click **Submit** (or click **Cancel** or **Make Additional Changes** if the changes noted in the summary do not match the desired changes). Individual request line items can be deleted from this window by clicking the **x** icon on any row. Comments can be added to any of the change records by clicking the icon in the **Add Comments** column. These comments are stored on the IdentityRequest object, which can be accessed later through the **My Work** > **Access Requests** menu option.

   Note: The password reset only occurs if all requested changes can be made successfully. If the password reset fails, an error message is displayed at the top of the page indicating the failure.

Requests for Others

As described in Enabling Password Management in IdentityIQ, the sets of Identities for which a user can make requests, as well as the types of requests available to each user, depend on the Lifecycle Manager Configuration settings that apply to that Identity. The rest of this section assumes that the logged-in user is authorized to make password requests for the Identity needing a password change.

Complete these steps to reset another user's password on an external application through IdentityIQ:

1. From the **Manage Access Quicklink**, click **Change Passwords** and select **For Others**.
2. Select the Identity for whom the password change is required.
3. Specify the password change method:
   - **Set passwords for the selected accounts**: enter new passwords manually on this window
   - **Synchronize passwords for selected accounts**: apply a single manually entered password to all of the selected accounts (rather than entering a separate new password for each selected account)
   - **Generate passwords for the selected accounts**: allow system to generate new passwords

   Note: When passwords are reset for another user, the system automatically sets a flag that tells the external application to require a password reset upon initial login by the user, so whether the password is manually set or generated, the user is prompted to change it when they first sign in to the target application.

   Note: The Generate passwords for the selected accounts option can be turned on or off from the Lifecycle Manager Configuration window, Additional Options tab. Select or clear the Enable password auto-generation when requesting for others box in the Manage Password Options section.
4. Select the application account or accounts for which the password is being changed.

5. Enter the new password twice - once in **New Password** and once in **Confirm Password** - if prompted.
   - If **Generate passwords for the selected accounts** is selected, the system does not prompt for a new password.
   - If **Synchronize passwords for the selected accounts** is selected, the password prompting occurs one time at the top of the window above the accounts list.
   - Otherwise, each selected application account has a set of password prompt boxes.
6. Click **Submit** at the bottom of the window to submit all password changes.

   Note: If the entered passwords do not match or if the password does not meet the requirements of the application's password policy, an error message is displayed on this window and the password values must be reentered before the requested changes can be successfully be submitted.
7. A summary of the requested changes is displayed on the next window. If the password is a generated password, the password is displayed in the **Password** column. If it was manually entered, it is represented with ***** in that column. Review this summary and click **Submit** (or click **Cancel** or **Make Additional Changes** if the changes noted in the summary do not match the desired changes). Individual line items can be deleted from this window by clicking the icon on any row. Comments can be added to any of the change records by clicking the icon in the **Add Comments** column. These comments are stored on the IdentityRequest object, which can be accessed later through the access request pages.

   Note: The password reset only occurs if all requested changes can be made successfully. If the password reset fails, an error message is displayed at the top of the page indicating the failure.

LCM Manage Passwords Workflow

By default, passwords requests (forgot, expired, or change), either self-service or for others, invoke the LCM Manage Passwords workflow. This workflow's default configuration requires no application-owner or manager approvals on a password change. It creates and processes a provisioning plan that contains the requested password changes and then notifies the user by email when the change is complete.

If the change request is for an account whose application is configured with a Change Password provisioning policy, additional information is required before the change occurs.

The default email template for password change notification sends a summary of the change request. This includes the requester, some representation of the new password, and any comments entered on the request (from the **Summary of Requests** window). If the password was system generated, that password is included in the email body. If it was a manually entered password, it is displayed in the email body as ******; in the case of request-for-others password resets, the new password value must be verbally, or otherwise, communicated to the user by the person who made the change.

To direct IdentityIQ to use a different, custom workflow for password management, create a workflow of type LCMProvisioning and select it as the Manage Passwords business process on the **Lifecycle Manager Configuration** window's **Business Processes** tab.

Passwords on New Account Requests

New account requests often contain password values. If you want to use default account-creation passwords that are different from the standard password policy for that application, IdentityIQ uses a configuration setting to govern the enforcement of password policies on account creation.

To enforce password policies on account creation, complete these steps:

- On the **Lifecycle Manager Configuration** page located under the gear icon menu, **Additional Options** tab, select the Check Password Policy rule as the **Password Validation Rule**. Check Password Policy is a rule that is supplied with Lifecycle Manager that validates the password field on an application's provisioning policy against the application's password policy. To write a custom rule, click the button to the right of that box.
- Define a Create provisioning policy for the application that includes a **password** field. This field name must end with `password`, must be of type **Secret**, and must not have its own validation rule specified for the Password Validation Rule to be applied. The connector maps this password provisioning policy field to the application's password field as the account is created.

When the provisioning policy form is presented for completion, by default to the application owner, the value entered in the **Password** field on the form is validated against the application's provisioning policy.

Troubleshooting Password Management with Provisioning Plan Debugging

Password changes are managed as provisioning activities, creating a provisioning plan that reflects the password change as an account modification request. Problems encountered with password management during the early set-up phases can be more easily diagnosed by turning on logging of the provisioning plan for individual applications as a debugging tool. The provisioning plan shows the actions IdentityIQ intends to perform on the account.

To turn on logging, add this XML block to the desired application's XML through the IdentityIQ Debug pages.

```
<!-- Inserting a provisioning configuration to support dumping the
     provisioning plan out to the log file during every execution. -->
<!-- The deleteToDisable flag prevents account deletion activities, changing them
     to disable account requests instead of delete  -->
<ProvisioningConfig deleteToDisable="true">
  <PlanInitializerScript>
    <Source>
       System.out.println("DEBUG: ProvisioningPlan: \n" + plan.toXml());
    </Source>
  </PlanInitializerScript>
</ProvisioningConfig>
```

This writes the provisioning plan to standard out (appearing as shown below). Note that the password is written in the provisioning plan in plain text, so this ProvisioningConfig should not be left in the Application XML in a production environment.

```
DEBUG: ProvisioningPlan:

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE ProvisioningPlan PUBLIC "sailpoint.dtd" "sailpoint.dtd">

<ProvisioningPlan targetIntegration="ADAM">

  <AccountRequest application="ADAM"
nativeIdentity="CN=Adam.Kennedy,DC=sailpoint,DC=com" op="Modify">

    <AttributeRequest name="password" op="Set" value="test123">

      <Attributes>

        <Map>
```

```
          <entry key="preExpire">
            <value>
              <Boolean>true</Boolean>
            </value>
          </entry>
        </Map>
      </Attributes>
    </AttributeRequest>
  </AccountRequest>
  <Attributes>
    <Map>
      <entry key="identityRequestId" value="0000000028"/>
      <entry key="requester" value="admin"/>
      <entry key="source" value="LCM"/>
    </Map>
  </Attributes>
  <Requesters>
    <Reference class="sailpoint.object.Identity"
id="2c901c1e34aa96a70134aa96e40200ba" name="admin"/>
  </Requesters>
</ProvisioningPlan>
```

Chapter 35: IdentityIQ Password Management

IdentityIQ supports multiple login configurations, including single sign-on, pass-through authentication, and validation against IdentityIQ's internally stored passwords. Pass-through authentication and internal passwords can be managed through the IdentityIQ user interface.

IdentityIQ's internal set of passwords are governed by the IdentityIQ password policy. These internal passwords are always available as a fallback login validation for IdentityIQ, even when other authentication methods are used. A user or an administrator can reset an internal password through IdentityIQ's change password options.

When pass-through authentication is used, IdentityIQ enables the specification of challenge questions that can enable users to reset their own forgotten passwords, once they authenticate to IdentityIQ by correctly answering those questions. New passwords entered through this forgot password feature are validated against the pass-through authentication application's password policy and are reset on that application directly.

For information, see “Password” on page 11 and "Login Configuration" on page 19.

IdentityIQ Password Configuration

IdentityIQ supports one-way hashing for following identity secrets:

- IdentityIQ password
- IdentityIQ password history
- IdentityIQ authentication question answers
- Application password history for external applications, such as Active Directory.

Note: Hashing support for application password history is enabled even if an application does not have a password policy.

To enable one-way hashing of secret values, click the **Gear** icon and select **Global Settings ->** IdentityIQ **Configuration -> Passwords** tab **-> Configuration**.

IdentityIQ Password Policy

The password policy for the IdentityIQ internally stored passwords is set in the System Setup configuration pages. Click the **Gear** icon and select **Global Settings ->** IdentityIQ **Configuration -> Passwords** tab **-> Password Policy**.

Most of the setting options are the same as the password policy options for application passwords. Unique settings available only for the IdentityIQ password policy are:

- **Define Character Types**: used to define allowable character types: Digits, Uppercase Characters, Lowercase or Non-English Characters, Special Characters. All characters are allowed if these fields are empty.
- **Days until expiration for manually set passwords**: used when a user resets their own password through the Edit Preferences window. This option sets the password expiration date by adding the specified number of days to the current date. The user is required to reset their password the first time they log into IdentityIQ on or after that expiration date.
- **Days until expiration for generated passwords**: used when an administrator resets a user's password through the Identity Cube's Attributes page. This option sets the password expiration date by adding the specified number of days to the current date. The user is required to reset their password the first time they log into IdentityIQ on or after that expiration date.
- **Minimum Hours between password changes**: specifies the amount of time (in hours) that must elapse before a user can reset their own IdentityIQ login password after they have reset it once. This does not prevent an administrator from resetting the user's password and does not prevent the user from resetting the password again immediately after it was reset by an administrator.
- **Require users to enter their current password when setting a new password**: enables a user to change their IdentityIQ password only if they enter the correct current password for the account.

Additionally, the following password policy options might not be immediately clear to a user, so they are described more fully here:

- **Password history length**: specifies number of previous passwords in password history to check against for uniqueness (prevents re-use of a password over the specified number of password changes)
- **Validate passwords against the password dictionary**: validates new IdentityIQ passwords against the password dictionary (see “Password Dictionary” on page 485)
- **Validate password against the identity’s list of attributes**: ensures that values stored as Identity attributes (last name, department, office number, region, etc.) are not used as the password

The **Validate passwords against the Identity's account attributes** option found on the application password policies does not apply to the IdentityIQ password policy. Those attributes are specific to each application and present a security risk when used in the login credentials for that specific application, but they do not pose the same risk for the IdentityIQ login.

Note: **The password history, if a Password history length value is specified, is stored as a <PasswordHistory> element on the Identity object. It is stored as a comma separated values list of encrypted passwords. The number of passwords stored is determined by the value set for the Password history length. IdentityIQ prevents the setting of a new IdentityIQ password for the user that matches any password in the list.**

Defining Special Characters for Password Use

IdentityIQ enables you to define the special characters that can be used in passwords throughout your deployment of the product. A default set of special characters are included in the System Configuration object To edit these special characters, go to the **Gear** icon and select **Global Settings ->** IdentityIQ **Configuration -> Passwords** tab and click the **Define Character Types** button. Alternatively, you can go to the debug page of the IdentityIQ user interface. The special characters enabled for use in passwords are listed in the passwordSpecialCharaters key.

Resetting IdentityIQ Internal Passwords

Each user's internally-stored password in IdentityIQ can be updated by that user on the Edit Preferences window. A user with rights to edit Identities' passwords (Password Administrator, Identity Administrator, etc.) can change passwords for other users as well through the Identity Cube.

Note: Passwords set through these options are the internally stored passwords for IdentityIQ. They are used as the primary authentication resource when the default login configuration is used. If pass-through authentication is enabled, the internal password (if one exists) is used to authenticate a user to IdentityIQ if authentication against the pass-through authentication resource fails. This password reset is not pushed out to any external resource.

Self-Service Password Reset

To change your own IdentityIQ password:

1. From the navigation menu bar, click the user name and select **Preferences**.
2. Click the **Change Password** link to display the section in which the new password can be entered.
3. Enter the new password twice, once in **New Password** and once in **Confirm New Password**. The password must meet the requirements of the IdentityIQ password policy.
4. If the IdentityIQ password policy requires that the current password be entered, the **Current Password** box also appears, and that value must be entered for the password change to be allowed.
5. Click **Save** at the bottom of the window to save the password changes.

Password Resets for Others

Note: To use this feature, you must have authority to reset passwords for other users.

To change an IdentityIQ password for another user:

1. From the navigation menu bar, click **Identities -> Identity Warehouse ->** [select Identity name]. Then click **Change Password** to display the password reset fields.
2. Enter the new password twice (once in **Password** and once in **Confirm Password**). The password must meet the requirements of the IdentityIQ password policy.
3. If this is a temporary password that the user should be prompted to reset, select **Require the user to change their password the next time that they log in**.
4. Click **Save** to save the password change. Password changes for others do not require the user to enter the current password even if that requirement exists for self-service password changes.

Password Expiration Resets

When a password expiration date is set for the IdentityIQ password, the system forces the user to change their password the first time they try to sign in, on or after the specified date.

First the user is informed that the password has expired. Click **Close** to acknowledge and dismiss this message.

Then the user is prompted to enter a new IdentityIQ password. Enter the new password twice (in **New Password** and **Confirm Password**) and click **Change**.

Password Management with Pass-Through Authentication

Note: This feature is available when pass-through authentication is in use and can only be used to reset the password for a pass-through-authentication application.

When IdentityIQ is configured for pass-through authentication, the Forgot Password option can be turned on to enable a user to reset their password in the authenticating application. A user can then authenticate to IdentityIQ through authentication questions when they are unable to remember their password.

To enable this feature, from the Navigation bar, go to the **Gear** icon **-> Global Settings -> Login Configuration -> User Reset** tab and select **Enable Forgot Password**.

This feature causes the **Forgot Password?** link to appear on the IdentityIQ login window. When a user clicks this link, they are prompted to answer one or more authentication questions that enable IdentityIQ to verify their identity. After a user successfully answers th authentication questions, the user is prompted for a new password. The pass-through application is then updated with that new password.

Pass-Through Authentication Requirements

Though the setup of pass-through authentication is not the focus of this document, there are a few configurations that are required for Pass-Through Authentication to work. If these configurations are not properly completed, authentication features related to Pass-Through Authentication can be prevented from working.

The **Authentication Search Attributes** field for the application must contain the names of the application account schema attribute(s) that contain the Username entered during sign-on. This field tells IdentityIQ which application fields to search to locate the matching application account. One or more attribute names can be specified in this field.

Defining the Authentication Questions

To specify the authentication questions, from the Navigation bar, go to the **Gear** icon **-> Global Settings -> Login Configuration -> User Reset** tab -> **Authentication Question Configuration -> Questions** area. A default set of authentication questions is provided. Any of these can be removed from the list by clicking the icon next to the question to be deleted. Custom questions can be defined as needed by clicking the icon next to the last question in the list and entering a new question in the box that appears.

Configuring the Authentication Question Settings

To configure authentication questions, from the Navigation bar, go to the **Gear** icon **-> Global Settings -> Login Configuration -> User Reset** tab -> **Authentication Question Configuration -> Settings** area.

The purpose of each of these settings is described below:

- **Number of questions asked to authenticate an identity** — Specifies the number of correct answers to authentication questions the user has to provide to be authenticated by these questions
- **Number of authentication answers a user must have defined in** IdentityIQ — Specifies the number questions for which the user must provide answers in advance so they can be authenticated using these questions; questions without known answers cannot be used for authentication because there is no "correct" answer to be matched
- **Prompt users for answers to unanswered authentication questions upon successful login** — Causes IdentityIQ to check (during login) whether the user has the required number of authentication answers provided already and, if not, prompt the user for those answers
- **Maximum number of unsuccessful authentication attempts before** IdentityIQ **lockout** — Locks the IdentityIQ account when a user enters invalid authentication answers this number of times
- **Number of minutes a user will remain locked out due to unsuccessful authentication**: Determines the duration of the lockout before the user can try again to sign in to IdentityIQ. During the lockout period, an administrator with the appropriate system capabilities can unlock the account by clicking **Unlock Identity** on the Identity Cube's **Attributes** tab.

Recording Authentication Answers

A user can only be authenticated through these questions if the answers are pre-recorded in IdentityIQ. Users can be required to provide these answers or they can choose to provide (or modify) their own answers.

Requiring Authentication Answers

Users can be forced to provide answers to these questions by selecting **Prompt users for answers to unanswered authentication questions upon successful login** in the Authentication Questions Settings. This causes the system to check whether each user has the required number of authentication answers recorded during the login process. If too few answers are recorded for a user, the **Answer Authentication Questions** window is display and the user is required to answer these questions before they can gain access to IdentityIQ. The number of questions shown depends on the required number of answers in the Authentication Question Settings (**Number of authentication answers a user must have defined in** IdentityIQ). The user can select any of the configured questions from the question drop-down lists.

Users who have already provided the required number of answers are not prompted again; this window is bypassed in subsequent logins and they are taken directly to the normal IdentityIQ interface.

Independently Providing or Editing Authentication Answers

If users are not forced to provide authentication answers, users can choose to provide the answers through the Edit Preferences page. Users can also update their authentication answers on this window, including changing their answers or choosing different questions.

1. From the Navigation menu bar, click the user name and select **Preferences**.
2. On the Edit Preferences page, click the **Edit Authentication Questions** link to display the questions and currently provided answers.
3. Select the desired questions from the question lists and provide the appropriate answer for each question. Click **Save** to save the changes.

Chapter 36: Application-Specific Password Management Requirements

Some applications have specific configurations requirements that go beyond the basic password management requirements previously discussed in this document. This section explores some of those application-specific requirements.

Active Directory and ADAM: SSL

Both AD and ADAM require a secure connection (SSL) for any password management activities. IdentityIQ offers two separate read-write connectors for each of these applications.

SSL Configuration for the Direct Connector

Installations using the AD or ADAM Direct connector must generate and install an SSL certificate under AD/ADAM and then build a java key store for IdentityIQ that trusts the AD/ADAM SSL certificate.

These are the basic steps for building that java key store and configuring IdentityIQ to use it.

1. On a Domain Controller, log in as an administrator and open Internet Explorer. Navigate to **Tools** -> **Internet Options** -> **Content** and click **Certificates**.
2. Switch to the **Trusted Root Certificate Authorities Tab** and select the certificate issued by your Active Directory integrated Certificate Server. Click **Export**.
3. Choose **Base-64 encoded X.509(.CER)** as the **Export File Format**.
4. Specify file name for the exported certificate.
5. Finish the export and copy the exported.cer file to the Java client machine.
6. At the client machine run the following command from the jdk bin directory.

   ```
   keytool -import -alias [aliasname] -keystore [keystore filename] -file [fully qualified certificate filename]
   ```

   The key store (jks) file is created in the bin directory where the keytool command is found. The name of the file is the name you specified following the -keystore parameter, such as myCaCerts.jks.
7. Create the Application in IdentityIQ using the appropriate direct connector (Active Directory or LDAP - ADAM). Select **Use SSL** and provide all the required values. Save the application (do not click **Test Connection** yet).
8. Assuming that the keystore is created in `/tomcat/apache-tomcat-7.0.47/`, enter the following in `catalina.sh`:
   ```
   -Djavax.net.ssl.trustStore=/tomcat/apache-tomcat-7.0.47/myCaCerts.jks
   -Djavax.net.ssl.trustStorePassword=password
   ```
9. Restart the Tomcat server.

10. Return to the Application Definition in the UI and click **Test Connection** to verify that the SSL connection is properly configured.

Windows Local and Active Directory: IQService Agent

Note: AD and ADAM require a secure connection (SSL) for any password management activities.

The IQService is a native Windows service that enables IdentityIQ to participate in a Windows environment and access information only available through Win32 APIs. You must install and register an IQService before you can provision to Active Directory, aggregate Terminal Services attributes, collect information from the Windows Event Logs, or load local Windows users or groups through the Direct connectors. This includes provisioning of password changes.

IQService can be installed on an independent Windows computer or on a Windows machine that is a member of a domain. It listens for connections from an IdentityIQ instance and can be used to do one of several things, including:

- Aggregate access to the file shares on the server
- Aggregate local user and group definitions from the independent Windows machine
- Aggregate users and groups from the Active Directory or ADAM domain of which the machine is a member
- Change the passwords for a user who has rights to the independent Windows machine or the domain

The application definition for the Active Directory or Windows Local application must then be configured with the host and port where IQService is installed and listening.

Windows Desktop Password Reset Utility

Since a user would normally have to successfully log into their computer before accessing IdentityIQ (or any other application) through a web browser, enabling reset of a Windows Desktop password requires the installation of a utility application called IdentityIQ Lifecycle Manager Desktop Password Reset. This application adds a link or button to the Windows login screen that can be configured to connect users to IdentityIQ's Forgot Password feature (or any other web-based password management solution) in a restricted browser to change their password; this functionality bypasses the Windows login credential requirement for this specific and limited purpose.

Note: Users can only be authenticated and permitted to change the Windows Desktop password through the IdentityIQ Forgot Password functionality if they have previously configured challenge question answers that can be used for authentication.

This utility is available to any customer who has licensed the Lifecycle Manager product.

When this application is installed, the **Forgot Password?** button, tile, or link appears on the login windows.

If configured to point to the IdentityIQ Forgot Password functionality, the restricted browser window displays the IdentityIQ's challenge question authentication windows.