

[Search All Content](#)

Compass › Products › IdentityIQ › [Technical White Papers](#)
› [Audit Configuration in IdentityIQ](#)

[Options](#)

Audit Configuration in IdentityIQ



This document describes how to configure IdentityIQ to generate audit records and describes its preconfigured audit events.

- Introduction
- Audit Configuration Page
 - General Actions
 - Link Attribute Changes
 - Identity Attribute Changes
 - Class Actions
 - SCIM Resource Actions
- Custom Audit Records
 - Auditing from Rules or Scripts
 - Logging a Complete AuditEvent Object
 - Specifying Event Details to Auditor LogAs() Method
 - Calling Other Auditor.log() Methods
 - Auditing in Workflows
 - Auditing Examples
- Viewing Audit Events
 - Other Options for Audit Log Viewing

[↑ Top](#)

- View XML Objects or Database Table Records
- Create Custom Reports
- Audit Best Practices

Introduction

IdentityIQ can capture audit records on a variety of actions which occur in the system. Each installation may have different requirements for the types of activities that need to be audited, so IdentityIQ provides auditing flexibility through system configurations.

The built-in auditing options can be turned on and off through the IdentityIQ user interface on the Audit Configuration page. Additional audit records can be added programmatically by rules and workflows. This document first describes how to create audit events using these mechanisms and then explains how to view those audit events to see what has been recorded.

Audit Configuration Page

The majority of IdentityIQ's auditing is managed through the Audit Configuration page in the UI. To access the Audit Configuration page, click the **gear** icon > **Global Settings** (or, in versions 6.4 and earlier, **System Setup**) > **Audit Configuration**. This page is subdivided into five tabs: **General Actions**, **Link Attribute Changes**, **Identity Attribute Changes**, **Class Actions**, and **SCIM Resource Actions**, each governing a different category of audit events.

NOTE: The **Link Attribute Changes** tab is hidden if no editable link attributes are defined for the installation.

General Actions

The **General Actions** tab contains a long list of actions typically performed while using IdentityIQ that can be logged for auditing purposes. These include **Top** actions like logging in, running a task, and signing off on a certification. By

default, approximately a third of these actions are preconfigured to generate audit records; when auditing is turned on for an action, its corresponding box on this page is selected. An administrator can turn logging on or off for any of these actions to capture only the ones that matter to the organization's auditors.

Audit Configuration

General Actions | Link Attribute Changes | Identity

General Actions

- Login ☒
- Logout ☐
- Login Failure ☒
- Session Timeout ☐
- Import File ☒


Figure 1: Audit Configuration: General Actions Tab

The actions available on this page are listed below along with a description of the events they capture when selected for auditing.


Auditable Action	Description	On by Default
General System Actions		
Login	Creates an audit table entry for every successful IdentityIQ login, recording the username	no
Logout	Creates an audit table entry for every successful IdentityIQ logout, recording the username	no
Login Failure	Creates an audit table record for every failed login attempt, recording the username	no
Session Timeout	Creates an audit table entry for every IdentityIQ logout caused by the session timing out, recording the username	no
Import File	Records name of every file imported through the UI import option: gear icon > Global Settings (or in 6.4 and prior System Setup) > Import from File , or the console	no

↑ Top

Auditable Action	Description	On by Default
Run Task	Generates an audit record for each task run, recording the task name and the name of the user who launched it (Scheduler if it was a scheduled task)	no
Email Sent	Creates an audit record for each email sent by the system, recording the recipient email address and message subject. This audit entry does not indicate successful delivery or valid email address; it just means the email was successfully launched by IdentityIQ.	no
Email Failure	Creates an audit record any time email sending fails due to an IdentityIQ configuration problem or server connection error; failure reasons could include: incorrect email setup, failure to connect to the SMTP server, no email address configured, etc.; records the intended recipient email address, subject, and the error message	no
Certification Actions		
Delegate Certification Item	Creates an audit record when a certification item is delegated to another user, recording the delegator, new owner, the name of the delegated item, and the delegation comments	no
Delegation Completion	Records completion of delegation work items; shows the Identity completing the item and the work item number	no
Delegation Revocation	Captures when a delegation work item is revoked by the delegator, returning the certification items to the original certification; records the revoker's name, the former delegate, the work item number, and information about what was in the delegated (identity name, role name, etc.) as well as the certificationEntity and workItem IDs	no
Reassign Certification	Records an audit log entry when one or more certification entities are reassigned to another certifier; shows the original certifier, the new certifier, and the reassignment comments, as well as the original certification and reassignment certification IDs	no

 [Top](#)

Auditable Action	Description	On by Default
Rescind Certification	Logs an entry when a certification reassignment is recalled by the original certifier; includes name of parent certification, owner name, and the name of the certifier from whom the reassignment was recalled, and IDs for the recalled and parent certification	no
Remediate Certification Item	Creates an audit entry when a policy violation is corrected or a certification item is remediated, resulting in a remediation action; the remediation requestor and target identity are both shown, as are the owner of the remediation and the type of remediation action (Provisioning Request or WorkItem); for policy violation remediations, the policy and rule violated are also shown	no
Certification Signoff	Generates an audit record when a certification is signed off, recording the certification name and the certifier; if a certification includes a signoff approver rule, only the last person to sign off generates this audit event; any others in the chain generate certification signoff approval events (see below)	no
Certification Signoff Approval	Creates an audit record when a certification is signed off and a configured signoff approver rule routes the certification to another approver; records the certification name, the name of the person signing off, and the name of the user to whom the certification was sent for approval	no
Workflow and Work Item Auditing		
Escalate Work Item	Logs the old owner name and new owner name when a work item escalation occurs	no
Expire Work Item	Logs an audit record when a workItem reaches notification timeout and can no longer be escalated	no
Approve Work Item	Deprecated in favor of Update Role / Disable Role set of audit events; will be removed from AuditConfig in a future release	no

 Top

Auditable Action	Description	On by Default
Reject Work Item	Deprecated in favor of Update Role/ Disable Role set of audit events; will be removed from AuditConfig in a future release	no
Start Workflow Process	Creates an audit entry every time a workflow is launched; the name of the launched workflow is recorded	no
Update Role	Triggered by the Role Modeler – Owner Approval workflow; creates an audit event when a role modification undergoes approval (whether approved or rejected – shows “success” if approved, “failure” if rejected)	no
Disable Role	Triggered by the Role Modeler – Owner Approval workflow; creates an audit event when a disabled role undergoes approval (whether approved or rejected)	no
WorkItem Forwarded	Records an audit event when a user forwards a workitem to another user; includes the original owner, the new owner, the workitem number and ID, and any forwarding comments	yes
Assign Work Item	Deprecated audit event; will be removed from the AuditConfig in a future release	no
Assign Remediation Item	Deprecated audit event; will be removed from the AuditConfig in a future release	no
Perform Maintenance Task Statistics Logging The logging options in this section all report the counts of items processed by the respective functions of the Perform Maintenance task. They report the function name/description and a count.		
Prune Histories	Logs deletion of history snapshots Timing of snapshot deletion is controlled by the Days before snapshot deletion field on the gear icon > Global Settings (or in 6.4 and prior, System Setup) > IdentityIQ Settings > Miscellaneous page	no

Auditable Action	Description	On by Default
Prune Task Results	Logs deletion of task results Timing of task results deletion is controlled by the Days before task results deletion field on the gear icon > Global Settings (or in 6.4 and prior, System Setup) > IdentityIQ Settings > Miscellaneous page	no
Prune Requests	Logs deletion of request queue items Timing of request deletion is controlled by the requestMaxAge setting in the System Configuration XML	no
Prune Syslog Events	Logs deletion of events recorded in the syslog table Timing of task results deletion is controlled by the Days before syslog event deletion on the gear icon > Global Settings (or in 6.4 and prior, System Setup) > IdentityIQ Settings > Miscellaneous page	no
Prune Certifications	Logs deletion of certifications This only applies if certifications are not being archived for the installation. To be eligible for deletion, certifications must be top-level certifications that have been signed and finished and are older than the configured certificationMaxAge; that value is set through the Days before certifications are archived field on the gear icon > Global Settings (or in 6.4 and prior, System Setup) > IdentityIQ Settings > Miscellaneous page.	no
Archive Certifications	Logs archiving of certifications (when archiving is enabled) Timing of certification archiving is controlled by the Days before certifications are archived field on the gear icon > Global Settings (or in 6.4 and prior, System Setup) > IdentityIQ Settings > Miscellaneous page.	no

Auditable Action	Description	On by Default
Prune Certification Archives	Logs deletion of certification archives Timing of certification archiving is controlled by the Days before certification archive deletion field on the gear icon > Global Settings (or in 6.4 and prior, System Setup) > IdentityIQ Settings > Miscellaneous page.	no
Finish Certifications	Logs count of certifications progressed to the “finished” state	no
Scan Remediations	Logs the number of certifications scanned for remediations to determine whether the requested remediation activities have been completed	no
Forward Inactive [User] Work Items	Logs the number of workitems found assigned to inactive users and forwarded to another user (using the inactive user work item escalation rule)	no
Denormalize Scopes	Logs the number of scopes that were denormalized to correct scope paths recorded on other system objects Objects record the full path to their assigned scopes but if the scope hierarchy changes, these paths become “dirty” and must be denormalized to regain optimal performance.	no
Certifications Phased	Logs the number of certifications moved to the next appropriate phase	no
Certification Items Phased	Logs the number of certification items (in continuous certifications) that were moved to their next appropriate phase; Certification items in continuous certifications phase individually rather than as a whole certification	no
Continuous Certifications Processed	Logs the number of continuous certifications found and processed	no
Continuous Certification Items Required	Logs the number of continuous certification items moved to a Certification Required state	no
Continuous Certification Items Overdue	Logs of the number of continuous certification items moved to a Certification Overdue state	no

Auditable Action	Description	On by Default
Prune Provisioning Transactions	<p>Logs deletion of provisioning transactions (these are visible on the Administrator Console page under the gear menu)</p> <p>Timing of provisioning transactions deletion is controlled by the Days before provisioning transaction event deletion field on the gear icon > Global Settings (or in 6.4 and prior, System Setup) > IdentityIQ Settings > Miscellaneous page</p>	no
Provisioning-Related Audits		
Provision	<p>Creates an audit event for any provisioning result (entitlement adds, removals, account updates, etc.); indicates the requester, the user to whom the provisioning action applies, and the status (or the error message if it fails)</p>	no
Provisioning Expansion	<p>Creates an audit entry when a provisioning plan gets expanded into component parts that each must be provisioned separately; for example, when a role gets connected to an Identity, it may require creation of an account and then addition of entitlements on that account, each of which would be a separate provisioning actions and would therefore get its own separate audit entry; includes details about the specific provisioning action.</p>	no
Identity Event Audits		
Identity Manually Correlated	<p>Not currently used in logging audit events; manual correlation of accounts to identities creates an event for Link manually moved from one identity to another (see next item)</p>	no
Link manually moved from one identity to another	<p>Creates an audit record when an uncorrelated account gets manually correlated or when an account is moved from one Identity to another with the Move Account option on an Identity Cube's Application Accounts page</p>	no

Auditable Action	Description	On by Default
Identity Event	When an Identity Lifecycle Event occurs this will trigger and Audit entry Example Create a new Identity via LCM	yes
Role sunrise	Creates an audit record when a role is auto-activated on a sunrise (activation) date specified for the role; only applicable when role sunrise/sunset activation is configured	yes
Role sunset	Creates an audit record when a role is auto-deactivated on a sunset (deactivation) date specified for the role; only applicable when role sunrise/sunset activation is configured	yes
Policy Violation Audits		
Allow Exception on Violation	Records an audit event when a policy violation exception is allowed in a certification or through the My Work (or in version 6.4 and prior, Manage) > Policy Violations page; shows the user who allowed it, the user for whom it was allowed, the violated rule, the exception expiration date, and the comments entered for the exception	no
Correct Violation	Audits when a policy violation is corrected (e.g. by removing conflicting access); records the names of the user making the correction and the user who was in violation, the violated policy and rule names, and the remediation actions taken	no
Delegate Violation	Creates an audit record when a user delegates responsibility for addressing a policy violation to another user	no
Plugin Audits		
Plugin Installed	Creates an audit event when a user installs a Plugin; records the Plugin name (Target) and the user who installed the Plugin (Source)	no
Plugin Upgraded	Creates an audit event when a user upgrades a Plugin	no
Plugin Uninstalled	Creates an audit event when a user uninstalls a Plugin; records the Plugin name (Target) and the user who uninstalled the Plugin (Source)	no

 [Top](#)

Auditable Action	Description	On by Default
Plugin Enabled	Creates an audit event when a user enables a Plugin; records the Plugin name (Target) and the user who enabled the Plugin (Source)	no
Plugin Disabled	Creates an audit event when a user disables a Plugin; records the Plugin name (Target) and the user who disabled the Plugin (Source)	no
Plugin Configuration Changed	Creates an audit event when a user changes the configuration for a Plugin; records the Plugin name (Target), the user who changed the Plugin (Source), and the attributes' new values	no
Alerts		
Alert Processed	Creates an audit event when an alert record is evaluated against the IdentityIQ alert definitions. Only applicable when IdentityIQ is aggregating alerts from another system (e.g. SecurityIQ)	no
Alert Action Created	Creates an audit event when IdentityIQ has triggered a certification, email, or workflow in response to alert processing	no
LCM Event Actions		
Approve Line Item	Generates an audit entry when a user approves a pending access request item; records the requester, the approver, and several pieces of information about the access requested (account name, username, requested access, etc.)	yes
Accounts Request Started	Creates an audit entry when an account-related request is entered through LCM; the entry shows the requester, the user for whom the request was made, the application name, and (except for create requests) the account name	yes

Auditable Action	Description	On by Default
Comment Added	Creates an audit entry showing all work item comments recorded on a work item as it is processed; does not apply to certification work items -- only LCM request work items; generates one comment audit entry containing all comments from the work item when the work item is completed	yes
Create	Creates an audit event when a Create Identity action occurs through LCM; records all attributes provided on creation	yes
Delete	Adds an audit entry when an account is deleted through an LCM request; records the requesting user, the application, and the account name (native identity)	yes
Disable	Logs an audit event when an account is disabled through LCM, recording the requesting user, the application and the account (native identity)	yes
Enable	Creates an audit record when an account is enabled through LCM	yes
Entitlement Added	Creates an audit record when an Entitlement is added through an LCM request; records the requester, the affected Identity, the application and the attribute name/value that make up the requested entitlement	yes
Entitlement Removed	Logs an audit record when an Entitlement is removed through an LCM request; includes the requester, application, account name, attribute name/value of access to remove	yes
Entitlements Request Started	Does not currently create AuditEvents automatically; LCM workflow could be modified to add auditEvents for this action if desired	yes
Expired Password Start	Creates an audit event when a pass-through authentication login password has expired (on the native system) and is changed through IdentityIQ. NOTE: Expired password resets of internally stored IdentityIQ passwords do not generate this audit event.	yes

 Top

Auditable Action	Description	On by Default
Forgot Password Start	<p>Creates an audit entry to record a password reset through the Forgot Password feature (only available with pass-through authentication); records the identity name and the account name on the pass-through-authentication system</p> <p>NOTE FOR versions prior to 6.2: An error exists in the default AuditConfig that crosses the mapping between Forgot Password Start and Expired Password Start. Installations using these audit features should edit the AuditConfig to change the two AuditActions. They should read:</p> <pre><AuditAction displayName=&quot;Forgot Password Start&quot; enabled=&quot;true&quot; name=&quot;ForgotPasswordStart&quot; /></pre> <pre><AuditAction displayName=&quot;Expired Password Start&quot; enabled=&quot;true&quot; name=&quot;ExpirePasswordStart&quot; /></pre> <p>This is targeted to be fixed in version 6.2.</p>	yes
Identity Create Request Started	Creates an audit entry when a user submits a Create Identity request through LCM; logs identity name, workflow used to process the request, where the request came from, and the operation (Create)	yes
Identity Edit Request Started	Creates an audit entry when a user submits a Create Identity request through LCM; logs identity name, workflow used to process the request, where the request came from, and the operation (Create)	yes

Auditable Action	Description	On by Default
Manual Provisioning	Logs an audit entry when a request through LCM requires a manual workitem to complete the provisioning activity; records the performer, operation (e.g. add, remove), application, account name on which action is performed, attribute name/value of access removed/added, and the requester	yes
Modify	Creates an audit record when an Identity is edited through the LCM Edit Identity option. Records the user making the change, the affected IdentityIQ account (Identity name), and the new and previous attribute value in the change.	yes
Password Request Start	Logs an audit entry when a password change request is initiated through LCM	yes
Password Changed	Creates an audit entry when the password for a managed application is changed through IdentityIQ; this includes password changes on the pass-through authentication application through the Forgot Password feature as well as LCM requests to change password	yes
Password Change Failure	Creates an audit entry when a user attempts to change a password for a managed application through IdentityIQ, but the password update fails. Records the error details	yes
Forgot Password Changed	Audit events for this action are currently only generated as Forgot Password Start events (see <i>above</i>).	yes
Expired Password Changed	Audit events for this action are currently only generated as Expired Password Start events (see <i>above</i>).	yes

Auditable Action	Description	On by Default
Provisioning Complete	Creates an audit entry when a provisioning request is set to Complete by the Perform Identity Request Maintenance task; reports the provisioning requester, the identity to which the request pertains, the application and account, the attribute/value for the entitlement, and the operation (e.g. add/remove)	yes
Provisioning Failure	Logs an audit record when a provisioning request ends in failure	yes
Registration Started	Creates an audit event when a user who does not have an IdentityIQ account requests a new IdentityIQ account through the self-service registration process	yes
Reject Line Item	Generates an audit entry when a user rejects a pending access request item (role or entitlement); records the requester, the approver, and several pieces of information about the access requested (account name, username, requested access, etc.)	yes
Role Added	Creates an audit event when a role is added through an LCM request; records the requester, the affected Identity, and the role added	yes
Role Removed	Creates an audit event when a role is removed as a result of an LCM request; records the requester, the affected Identity, and the role removed	yes
Roles Request Started	Does not currently create AuditEvents automatically; LCM workflow could be modified to add AuditEvents for this action if desired	yes
Unlock	Creates an audit event when a locked application account is unlocked through LCM	yes
Access Request Started	Records an audit event when an LCM Access Request is created, showing the application, account, attribute name, and attribute value in the request	yes
Authentication-related Audits		

↑ Top

Auditable Action	Description	On by Default
Identity Locked	Records the username in an audit entry when a user's IdentityIQ account gets temporarily locked, either because they entered an incorrect password too many times or because they entered incorrect challenge question answers too many times	yes
Identity Unlocked	Records the username in an audit entry when a user's locked IdentityIQ account gets unlocked	yes
Authentication Answer Incorrect	Creates an audit event showing the username when a user attempts to authenticate through challenge questions (Forgot Password functionality) and fails to answer the required number of questions correctly	yes
Miscellaneous		
Password Policy Changed	Creates an audit event when an application's password policy has changed; records the source (who created the new password policy), the application, the password policy name (attribute value), and the policy attributes (minimum number of characters, maximum number of letters, etc.)	yes
Server Up/Down	Creates an audit event when an application server shuts down or starts up (Target = either "Server Up" or "Server Down")	yes
API Configuration Changed	Creates an audit event when an OAuth client configuration (for SCIM API authentication) is created or edited; records the Source, application, and attributes, along with the type of change that occurred	yes

Link Attribute Changes

The **Link Attribute Changes** tab lists account attributes that can be audited, meaning a change in the attribute value creates an audit event. (Links represent application accounts in IdentityIQ.) Only attribute changes that are the result of user requests – LCM requests or Identity Cube edits – are tracked in attribute-change audit events; attribute changes that come from

aggregations do not generate audit records. Consequently, only editable attributes -- those marked as **Edit Mode: Permanent** or **Temporary** in the **gear** icon > **Global Settings** (or in version 6.4 and prior, **System Setup**) > **Account Mappings** page -- are available for auditing.

If no account attributes are marked as editable, the **Link Attribute Changes** tab does not appear on the **Audit Configuration** page. By default, no attribute changes are audited, but once they are selected for auditing on this **Link Attribute Changes** page, audit log entries are generated any time those attribute values change in response to a user action.

To request auditing on editable account attributes, click **gear** icon > **Global Settings** (or in version 6.4 and prior, **System Setup**) > **Audit Configuration** > **Link Attribute Changes** and select the desired account attributes.

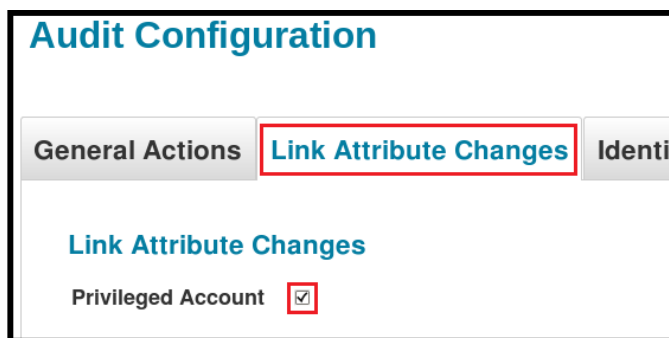


Figure 2: Audit Configuration: Link Attribute Changes Tab

Auditable attributes are recorded in the AuditConfig object XML in the AuditAttributes list, noted with class="Link". (This differentiates them from auditable identity attributes, discussed in the next section.) When the attribute has been selected for auditing, the attribute enabled="true" is added to the AuditAttribute element.

```
<AuditAttributes>
```

```
...
```

```
  <AuditAttribute class="Link" displayName="Privileged
  Account" enabled="true"
  name="privileged"/>
```

```
</AuditAttributes>
```

Identity Attribute Changes

The **Identity Attribute Changes** tab specifies the attributes on the Identity object which can be audited for changes. By default, the attributes listed here are pre-defined to be available for auditing:

- Assigned Roles
- Capabilities
- Authorized Scopes
- Password
- Assigned Scope
- Correlation Status

Additionally, any extended identity attributes configured as editable through the UI (marked as **Edit Mode: Permanent** or **Temporary** on the **gear** icon > **Global Settings** (or in version 6.4 and prior, **System Setup**) > **Identity Mappings** page) are automatically available for auditing, though they are not audited automatically. To turn on auditing for the preconfigured attributes or the editable identity attributes, click **gear** icon > **Global Settings** (or in version 6.4 and prior, **System Setup**) > **Audit Configuration** -> **Identity Attribute Changes** and select the attributes to be audited.

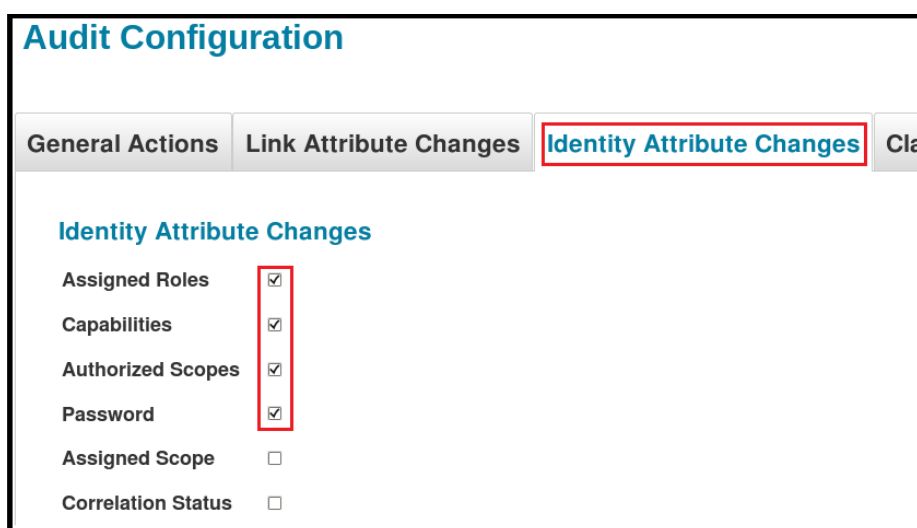


Figure 3: Audit Configuration: Identity Attribute Changes Tab

Only attribute changes that are the result of user requests – LCM requests or Identity Cube edits – are tracked in attribute-change audit events; attribute changes that come from aggregations do not generate these audit records.

In the AuditConfig object XML (viewable through the debug pages), these are listed as AuditAttribute elements with class="Identity". When the attribute has been selected for auditing, the attribute enabled="true" is added to the

AuditAttribute element.

```
<AuditAttributes>

...

  <AuditAttribute class=&quot;Identity&quot;
    displayName=&quot;Department&quot;
    name=&quot;department&quot; enabled=&quot;true&quot;/>

</AuditAttributes>
```

NOTE: Editable extended identity attributes are only added to the AuditAttributes list when the audit configuration is edited through the UI, so an examination of the AuditConfig XML may not always show those AuditAttribute elements. They can be manually added to the XML or can be auto-added by viewing and saving the **Identity Attribute Changes** page of the **Audit Configuration** through the UI. Since audit records are only generated by manual edits, any ineligible (i.e. non-editable) Identity attributes added to the AuditConfig will generate no audit records, and IdentityIQ will automatically delete them from the AuditConfig when any changes are saved from the **Identity Attribute Changes** page in the UI.

Class Actions

Some installations may want to audit any changes to objects of specific classes in their system. On the **Class Actions** tab, auditing can be configured for any create, update, or delete action on any of the configured classes. For example, audit records can be created when a user edits a role, creates a policy, or creates or changes an email template.

Out of the box, the classes listed below are configured to allow auditing of their create/update/delete events, though by default, none of these events are audited:

- ActivityDataSource
- Application
- AuditConfig
- Bundle (displayed on the Class Actions tab as "Role")
- Category
- Capability
- Configuration

[↑ Top](#)

- EmailTemplate
- GroupFactory
- JasperTemplate
- MessageTemplate
- MiningConfig
- ObjectConfig
- Policy
- RequestDefinition
- RightConfig
- Rule
- ScoreConfig
- TaskDefinition
- TaskSchedule
- TimePeriod
- UIConfig
- Workflow

Most top-level sailpoint.object.* classes can be configured for auditing. To add additional classes to the auditable class list, add a new AuditClass element to the AuditConfig XML through the debug pages.

```
<AuditConfig name=&quot;AuditConfig&quot; ... >  
  
...  
  
<AuditClasses>  
  
...  
  
  <AuditClass name=&quot;IntegrationConfig&quot;/>  
  
</AuditClasses>  
  
</AuditConfig>
```

NOTE: Class auditing cannot be enabled for the AuditEvent class. It also should not be enabled for classes that are system-managed and not editable by users, since audit events are not generated for system actions. Finally, it should not be enabled for classes that are serialized inside another class (e.g. DashboardLayout, Link) if the enclosing class is also audited, since this can result in duplicate audit events.

To turn on auditing for any of the configured classes, click **gear** icon > **Global Settings** (or in version 6.4 and prior, **System Setup**) > **Audit Configuration** > **Class Actions** and select **Create**, **Update**, or **Delete** for each class to be audited.

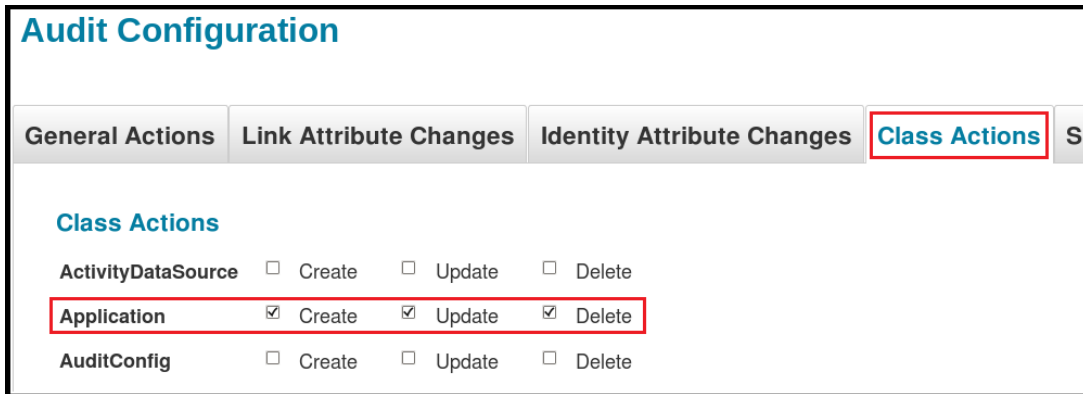


Figure 4: Audit Configuration: Class Actions Tab

NOTE: These class audit options should be turned on only as needed; on some classes, they can result in a large number of audit events. For example, if class auditing were turned on for Identity updates, every update from any source – the UI, an aggregation, or any other task – would create an audit record. (Identity is not included as a auditable class by default.)

SCIM Resource Actions

SCIM (System for Cross-Domain Identity Management) is a standard that defines a schema and API to create, read, update, and delete identity and identity-related information. This standard creates a common language, by which a client system can communicate with many different servers in the same way.

Out of the box, the objects listed below are configured to allow auditing when these objects are read/created/updated/deleted in IdentityIQ through the SCIM API, and by default, the create/update/delete events for Users (i.e. Identities) and Accounts are audited, as well as the Create events for LaunchedWorkflow:

- User
- Application
- Entitlement
- Role
- Account
- Workflow

↑ Top

- TaskResult
- LaunchedWorkflow
- PolicyViolation
- CheckedPolicyViolation

NOTE: Not all of these resource actions exist for all versions of IdentityIQ; this list will continue to evolve as the SCIM functionality expands.

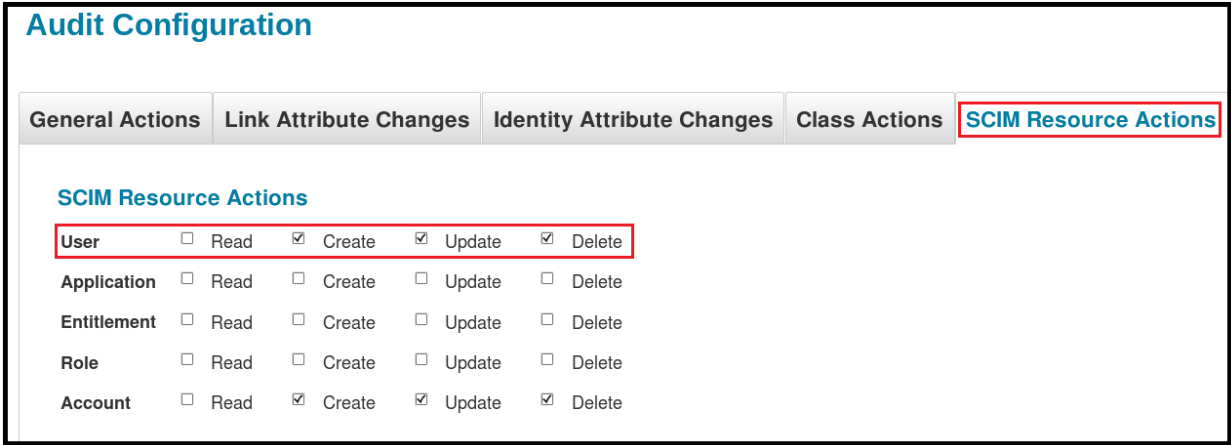


Figure 5: Audit Configuration: SCIM Resource Actions Tab

More information on the IdentityIQ SCIM API can be found here: [IdentityIQ SCIM API](#).

Custom Audit Records

Audit events can also be generated by workflows, rules, and scripts running in the IdentityIQ instance. Often, these are used to create custom audit records needed for a specific installation’s audit requirements. The available fields in an AuditEvent can be populated as deemed appropriate to capture the important data about the events being audited.

The recommended best practice is to create audit configuration entries (as AuditActions in the AuditConfig object) for these custom events so auditing can be turned on and off for custom events the same way it can be for built-in events. A custom AuditConfig entry looks like this:

```
<AuditConfig name="AuditConfig">
  <AuditActions>
```

↑ Top

...

```
<AuditAction displayName=&quot;Certification Exclusion&quot;
  enabled=&quot;true&quot;
  name=&quot;CertificationExclusion&quot;/>
```

```
</AuditActions>
```

...

NOTE: Both Name and DisplayName are used in the UI and should therefore be descriptive and readable; name may be a shorter version of DisplayName. The Name value is shown on the Advanced Analytics **Audit Search Results** page as the **Action** value; the DisplayName is displayed on the **Audit Configuration** page as well as in the **Audit Event Details** pop-up window that appears when an item in the search results list is clicked to show all its details.

Once a custom AuditAction has been saved in the AuditConfig object, it appears on the **General Actions** tab of the **Audit Configuration** UI page and can be turned on or off from there as well as through the XML. Items appear on that page in the order in which they are specified in the AuditConfig XML.

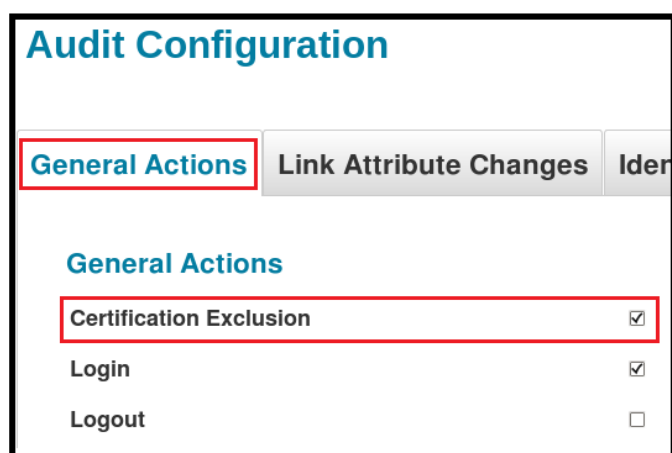


Figure 6: Custom Audit Config Entry

Auditing from Rules or Scripts

Rules and scripts can create audit records in a few different ways by calling methods available in the Auditor class.

Logging a Complete AuditEvent Object

One way to create an audit entry is to build an AuditEvent and pass it to the Auditor class's log() method to save the record. The signature for that method is:

```
static public void log(AuditEvent event)
```

This method does not validate whether the action is enabled before creating the specified audit entry, so the BeanShell in the rule or script should use the Auditor.isEnabled() method to check whether the AuditAction has been enabled before logging the event.

```
import sailpoint.server.Auditor;
```

```
String action = "ActionName";
```

```
if ( Auditor.isEnabled(action) ) {
```

```
    [Proceed with auditing processing]
```

```
}
```

The components that go into an AuditEvent vary depending on the action being recorded. AuditEvents generally includes a Source, an Action, and a Target; the rest of the fields can be set or left blank as required for creating a meaningful record of the specific event. These methods are available in the AuditEvent class for creating and populating the AuditEvent being recorded.

AuditEvent Class Method	Description
AuditEvent()	Basic constructor; populates no fields
AuditEvent(String source, String action)	Constructor; populates source and action attributes from provided string values
AuditEvent(String source, String action, String target)	Constructor; populates source, action, and target with the provided string values
AuditEvent(String source, String action, SailPointObject obj)	Constructor; populates source and action from provided strings and extracts the name or ID from the object to set as the target attribute

↑ Top

<code>setSource(String source)</code>	Sets the source attribute to the provided string; source is the Identity performing the action or can be "system" for audit records pertaining to automated tasks
<code>setTarget(String target)</code>	Sets the target attribute to the provided string; usually the name of a SailPointObject (e.g. an Identity, a Certification, a Role, etc.) on which the audited action occurred
<code>setAction(String action)</code>	Sets the action attribute to the provided string; action describes what happened; this string should match the corresponding AuditConfig's name attribute
<code>setApplication(String app)</code>	Sets the application attribute on the event to the provided string (application name); only applies to actions that relate to a specific application (e.g. provisioning activities)
<code>setInstance(String instance)</code>	Sets the instance attribute to the provided instance name; only applicable for applications with instances defined
<code>setAccountName(String accountName)</code>	Sets the accountName attribute in the event to the provided account name (should match the native identity value for the Link to which the audited action applies)
<code>setAttributeName(String attribute)</code>	Sets the AttributeName attribute in the event to the provided attribute name; used to specify the affected attribute when the audited event is setting a single value on an account or identity
<code>setAttributeValue(String value)</code>	Sets the AttributeValue attribute in the event to the provided attribute value; used in conjunction with AttributeName to specify the value for the affected attribute when the audited event is setting a single value on an account or identity
<code>setString1(String string), setString2(String string), setString3(String string), setString4(String string)</code>	Set the values of the string1-string4 attributes; these can be any strings that are meaningful for the audit event; can be helpful to include a short label for these values
<code>setAttributes(Attributes<String, Object> attr)</code>	Sets the attributes map in the event to the provided map; overwrites any existing values in the map; attributes map can contain any name-value pairs that are meaningful for the audit event
<code>setAttribute(String name, Object value)</code>	Adds the provided name-value pair to the event's attributes map

<code>setTrackingId(String trackingId)</code>	Sets the value of the trackingId attribute; can be used to connect audit events together in groups; often, workflowCase ID is stored here to correlate items that occurred in a single workflow Not displayed in Advanced Analytics data about audit events
<code>setInterface(String s)</code>	Indicates application module source for the event; expected to be a sailpoint.object.Source enumeration name (see JavaDocs) (e.g. <code>event.setInterface(Source.LCM.toString());</code>) Not displayed in the Advanced Analytics data about audit events

This example rule (a certification exclusion rule) creates an audit event when certification items are excluded from the certification. The exclusion criterion in this example is that the identity is a contractor (rather than an employee).

```
import sailpoint.server.Auditor;
```

```
import sailpoint.object.AuditEvent;
```

```
System.out.println("&quot;Entering Exclusion Rule.&quot;;);
```

```
if
(identity.getAttribute("&quot;status&quot;").equals("&quot;Contractor&quot;")) {
```

```
    System.out.println("&quot;Not certifying contractor : &quot; +
identity.getDisplayName());
```

```
    String reason = "Excluding Contractors";
```

```
itemsToExclude.addAll(items);
```

```
items.clear();
```

```
String action = "&quot;CertificationExclusion&quot;;
```

```
if (Auditor.isEnabled(action)) {
```

```
    AuditEvent event = new AuditEvent();
```

```
    event.setSource(certification.getCreator());
```

```
    event.setTarget(certification.getName());
```

```
    event.setAction(action);
```

[↑ Top](#)

```
        event.setString1("&quot;Identity: &quot; +  
        identity.getDisplayName());  
  
        event.setString2("Reason: " + reason);  
  
        Auditor.log(event);  
  
    }  
  
    return reason;  
  
}  
  
return null;
```

NOTE: The Auditor.log(event) method saves the record but does not commit the changes. Certification generation already includes transaction commitment, so a commit is not required here, but there may be circumstances where a commit must be explicitly included in the rule or workflow to ensure that the auditEvent gets saved to the database.

```
context.commitTransaction();
```

Specifying Event Details to Auditor LogAs() Method

The Auditor class's LogAs() method can create and save an AuditEvent from string components passed to it. The signature of the LogAs method is:

```
static public boolean logAs(String actor, String action, String target,  
    String arg1, String arg2, String arg3, String arg4)
```

The "actor" parameter becomes the AuditEvent's "source" attribute and the "arg1" – "arg4" parameters map to the "string1" – "string4" attributes. The "action" and "target" parameters become the AuditEvent object's "action" and "target" values. This method does not populate the other attributes in the AuditEvent, so it cannot be used when the AuditEvent should contain values for the attributes: application, instance, attribute name, attribute value, or the attributes map.

This method checks whether the action is enabled before creating the AuditEvent, so the rule or script does not have to do that check before calling it. However, this also means that the AuditAction for events logged with this method *must* be added to the AuditConfig for the entries to be generated. The return value from the method is the Boolean value indicating whether the action was enabled for auditing or not.

This is another example of a certification exclusion rule. It creates an identical AuditEvent to the one created by the rule that called Auditor.log(event) in the previous section, but it uses the logAs() method instead.

```
import sailpoint.server.Auditor;

import sailpoint.object.AuditEvent;

import sailpoint.object.Identity;

// Exclude Contractors from this certification

if
(identity.getAttribute(&quot;status&quot;).equals(&quot;Contractor&quot;
)) {

    String reason = "Excluding Contractors";

    String action = &quot;CertificationExclusion&quot;;

    itemsToExclude.addAll(items);

    items.clear();

    Boolean logged = Auditor.logAs( certification.getCreator(), action,
        certification.getName(), &quot;Identity: &quot; +
        identity.getDisplayName(), &quot;Reason: &quot; + reason, null,
        null );

    if (!logged)

        log.debug(&quot;Logging disabled for CertificationExclusion
events&quot;);

    return reason;
```

[↑ Top](#)

```
}
```

```
return null;
```

NOTE: The Auditor.logAs() method saves the record but does not commit the changes. Certification generation already includes transaction commitment, so a commit is not required here, but there may be circumstances where a commit must be explicitly included in the rule or workflow to ensure that the AuditEvent gets saved to the database.

Calling Other Auditor.log() Methods

The Auditor class also offers several signatures for the log() method that can be called with a smaller subset of data values. A source for the event is not specified to these methods, so they set the context user as the source for the event; all other values for the event that are not provided as arguments to the method are left blank (null). These are the available signatures for the log method:

```
static public boolean log(String action)
```

```
static public boolean log(String action, String target)
```

```
static public boolean log(String action, String target, String arg1)
```

```
static public boolean log(String action, String target, String arg1, String arg2)
```

```
static public boolean log(String action, String target, String arg1, String  
arg2, String arg3)
```

Like the logAs method, these do not set the attributes: application, instance, attribute name, attribute value, or the attributes map. They also check whether the action is enabled for auditing or not before creating the AuditEvent entry, and they return the Boolean value indicating whether the action was auditable or not.

If this line were substituted into the rule example above in place of the logAs() method call, it would create an almost identical audit entry; the only difference is the source value would be "Scheduler" instead of the certification creator's name:

```
Boolean logged = Auditor.log(action, certification.getName(), &quot;Identity:
&quot; + identity.getDisplayName(), &quot;Reason: &quot; + reason);
```

Auditing in Workflows

Workflows can create audit events through scripts and rules as described above, but they can also use provided methods in the Standard Workflow Handler and other workflow libraries to perform auditing functions. Values are passed to the library methods through step arguments which become part of the workflowContext; the workflowContext is automatically passed to every library method when it is invoked. The workflow excerpt below, taken from the Scheduled Role Activation workflow, shows an example of a call to the audit() method in the Standard Workflow Handler.

```
<Workflow name="Scheduled Role Activation" ... >

...

<Step action="audit" icon="Audit" name="Audit Success">

  <Arg name=&quot;source&quot; value=&quot;ref:launcher&quot;/>

  <Arg name=&quot;action&quot; value=&quot;script:(activate) ?
&quot;activate&quot; : &quot;deactivate&quot;&quot;/>

  <Arg name=&quot;target&quot; value=&quot;ref:roleName&quot;/>

  <Arg name=&quot;string1&quot; value=&quot;script:((activate) ?
&quot;Activated &quot; : &quot;Deactivated &quot;) + roleName&quot;/>

  <Arg name=&quot;string2&quot; value=&quot;Success&quot;/>

  <Transition to=&quot;Refresh Identities&quot;
when=&quot;ref:doRefresh&quot;/>

  <Transition to="end"/>

</Step>

...

</Workflow>
```

The action="audit" element in the step tells IdentityIQ to run the Standard Workflow Handler's audit() method; the standardWorkflowHandler methods are all available to any workflow. The <arg> elements are passed to the method through the workflowContext. The audit() method can accept these ↑ Top

arguments: source, action, target, and string1 – string4; this example omits string3 and string4, so they are left null in the audit event that is created. This method only populates some of the attributes of an auditEvent; it does not populate the application, instance, attributeName, attributeValue, or attributes map.

This table shows the workflow library methods that are available for auditing functionality:

Class Library	Method	Description
Standard Workflow Handler	public Object audit (WorkflowContext wfc)	Passed a source, action, target, string1 – string4 value in the workflow context, creates an audit event from those attributes
Identity Library	public Object auditLCMStart (WorkflowContext wfc)	Passed an approvalSet and flow name in the workflow context, creates an audit event to mark the start of an LCM workflow for each item in the approvalSet
	public Object auditLCMCompletion (WorkflowContext wfc)	Passed an approvalSet and flow name in the workflow context, creates an audit event to mark the completion of each item in the approvalSet
	public Object auditLCMEvents (String eventType, WorkflowContext wfc)	Passed an approvalSet and flow name in the workflow context, creates an audit event to mark the completion of each item in the approvalSet
Role Library	public Object auditRoleDifferences (WorkflowContext wfc)	Passed a source, action, target, and string1 value, creates audit events – one for each attribute difference between role states (workflow vs database); string2-4 values for the audit event are calculated in this diff process

Custom workflow libraries can be built for an installation to support other auditing actions (or other functions in general).

Auditing Examples

Several of the built-in IdentityIQ workflows include auditing functionality (usually in the approval step) implemented either through library calls or through calls to rules or rule libraries which contain auditing logic. Use these as a reference for creating any desired custom auditing logic.

The rules and rule libraries can be viewed through the debug pages, making them available as models for creating customized auditing rules, or they can be called by other rules or workflows to perform the specific auditing actions they provide. Rule libraries are stored as Rule objects but contain multiple independent methods instead of a single thread of logic.

Viewing Audit Events

The Advanced Analytics Audit Search page provides user visibility to audit events recorded by the system. Click **Intelligence** (or in 6.4 and prior, **Analyze**) > **Advanced Analytics** and change Search Type to **Audit** to access that page. Any user with the Auditor, Compliance Officer, or System Administrator Capability can see this page.

Click **Run Search** without setting any criteria to view the entire set of audit events in the system or specify filters to narrow the results.

Advanced Analytics

Search Type
Audit

Search Criteria ?

Audit Attributes

Action		▼	Source		▼
Application		▼	Instance		▼
Attribute Name		▼	Attribute Value		▼
Target		▼	Account Name		▼

Filter by: Date

Start Date ☐
End Date ☐

Run Search
Clear Search

Figure 7: Advanced Analytics Audit Search Filter Page

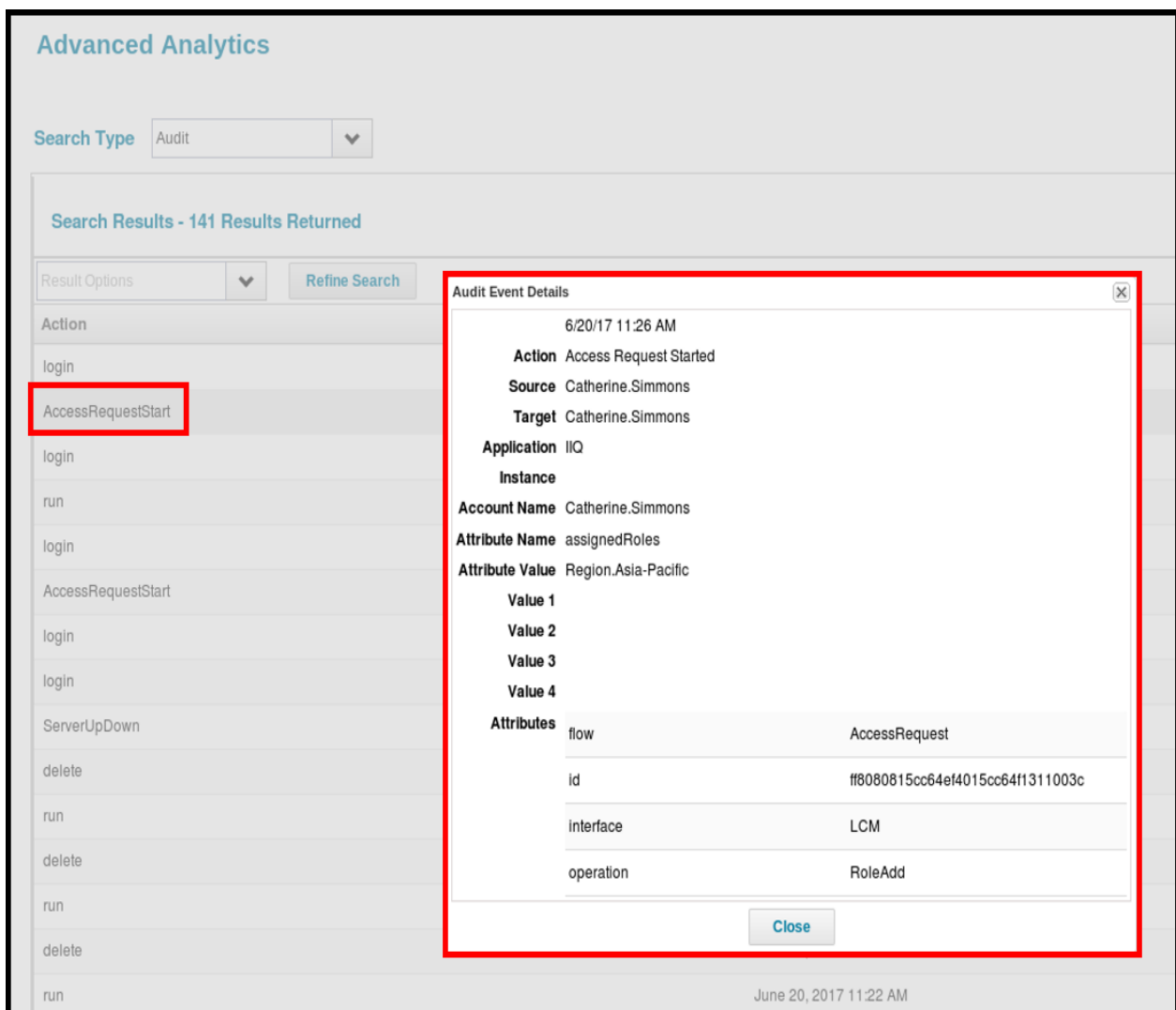
The table below lists the audit log fields available for display through the Advanced Analytics search. Not all fields may be populated for all audit events: the fields populated in the audit table depend on the action being audited.

Field	Description
Account Name	Unique identifier of the account to which the action applied; only populated when Application is populated; for actions on Application IIQ, this is the Identity's name attribute
Action	Audit event type, or what occurred to trigger the audit event (e.g. Create, ApproveLifecycleEvent, Forward, Import, Change, Run)
Application	The application to which the action applied; some internal IdentityIQ actions that do not affect managed applications (e.g. identity creation and line item approval events) show IIQ as the application while others (e.g. forwarding) do not show an application

[↑ Top](#)

Attribute Name	The account attribute involved in the audited action; only populated for application account actions (so both application and account name will also be populated)
Attribute Value	The attribute value in the audited action; usually this is the value for the attribute named in Attribute Name, but some audit transactions record an attribute value with no attribute name
Date	Creation date for the audit event
Instance	The application instance to which the action applied; only applicable if instances are used
Interface	The interface that was used to perform the action (e.g. LCM or UI); not all audited actions have an Interface
Source	The Identity, system user, or task performing the action (e.g. forwarding user, workitem approver, or task launcher such as Janet Washington or The Administrator; RequestHandler; HR Aggregation task)
Target	System entity that was the object of the action (e.g. Identity name, workitem number, certification name, import filename); most but not all audited actions have a Target
Value 1, Value 2, Value 3, Value 4	Additional information about the audited event; usually used when Account name, attribute name, and attribute value are not used e.g.: In Forwarding events, these are sender, recipient, how the forwarding occurred (manual or rule), and the ID of the work item, respectively. In the AuditEvent XML and spt_audit_event database table, these are named string1 – string4.
Attributes	Only visible through the UI by clicking each individual item in the search results list; displays the contents of the audit event's attributes map, which often contains comments about the event and may contain other attributes, depending on the event

By default, only the **Action**, **Date**, **Source**, and **Target** fields are displayed on the main grid; all attributes for each line item can be seen one at a time by clicking the record in the grid.



Advanced Analytics

Search Type: Audit

Search Results - 141 Results Returned

Result Options: Refine Search

Action
login
AccessRequestStart
login
run
login
AccessRequestStart
login
login
ServerUpDown
delete
run
delete
run
delete
run

Audit Event Details
6/20/17 11:26 AM
Action Access Request Started
Source Catherine.Simmons
Target Catherine.Simmons
Application IIQ
Instance
Account Name Catherine.Simmons
Attribute Name assignedRoles
Attribute Value Region.Asia-Pacific
Value 1
Value 2
Value 3
Value 4
Attributes

flow	AccessRequest
id	ff8080815cc64ef4015cc64f1311003c
interface	LCM
operation	RoleAdd

[Close](#)

June 20, 2017 11:22 AM

Figure 8: Click Grid Record to see Event Details

To display more fields in the main grid, select them in the **Fields to Display** section of the **Audit Search Criteria** page.

Advanced Analytics

Search Type: ▼

Search Criteria ?

Audit Attributes

Action	<input type="text"/>	Source	<input type="text"/>
Application	<input type="text"/>	Instance	<input type="text"/>
Attribute Name	<input type="text"/>	Attribute Value	<input type="text"/>
Target	<input type="text"/>	Account Name	<input type="text"/>

Filter by: Date

Start Date ☐ End Date ☐

Fields to Display ?

Audit Fields

- ☐ Account Name
- ☒ Action
- ☐ Application
- ☐ Attribute Name
- ☐ Attribute Value
- ☒ Date
- ☐ Instance
- ☐ Interface
- ☒ Source
- ☒ Target
- ☐ Value 1
- ☐ Value 2
- ☐ Value 3
- ☐ Value 4

Figure 9: Choose Additional Fields to Display in Main Results Grid

Other Options for Audit Log Viewing

View XML Objects or Database Table Records

Audit records are stored in IdentityIQ as AuditEvent objects and they are stored in the `spt_audit_event` table in the database. In addition to the Advanced Analytics option for viewing the records through the UI, an administrator can view the objects through the debug pages or the `iiq` console, and a user with database access can query the database directly for them.

Create Custom Reports

Additionally, custom reports can be developed for auditing purposes. The IdentityIQ 6.0+ reporting infrastructure makes that an easy task. The XML below represents a simple custom report that shows the Date, Source, Action, Target, and String 1–4 values for each audit event. It allows the user to specify a date range for the events to report. The corresponding form to accept that [Top](#)

entered date range is shown below the report TaskDefinition XML. These can be modified to include different report columns, to alter the sort order, or to offer additional filter criteria.

Custom Audit Report TaskDefinition XML:

```

<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE TaskDefinition PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<TaskDefinition executor="sailpoint.reporting.LiveReportExecutor" name="Audit Report"
  <Attributes>
    <Map>
      <entry key="report">
        <value>
          <LiveReport title="Audit Report">
            <DataSource objectType="AuditEvent" type="Filter">
              <QueryParameters>
                <Parameter argument="creationDateStart" operation="GT" property="created"/>
                <Parameter argument="creationDateEnd" operation="LT" property="created"/>
              </QueryParameters>
            </DataSource>
            <ReportForm>
              <Reference class="sailpoint.object.Form" name="Audit Report Form"/>
            </ReportForm>
            <Columns>
              <ReportColumnConfig field="created" header="Date" property="created" sort="asc"/>
              <ReportColumnConfig field="action" header="Action" property="action" sort="asc"/>
              <ReportColumnConfig field="source" header="Source" property="source" sort="asc"/>
              <ReportColumnConfig field="target" header="Target" property="target" sort="asc"/>
              <ReportColumnConfig field="string1" header="String 1" property="string1" sort="asc"/>
              <ReportColumnConfig field="string2" header="String 2" property="string2" sort="asc"/>
              <ReportColumnConfig field="string3" header="String 3" property="string3" sort="asc"/>
              <ReportColumnConfig field="string4" header="String 4" property="string4" sort="asc"/>
            </Columns>
          </LiveReport>
        </value>
      </entry>
    </Map>
  </Attributes>
  <Description>Displays audit records generated between specified dates.</Description>
  <RequiredRights>
    <Reference class="sailpoint.object.SPRight" name="FullAccessWorkItemReport"/>
  </RequiredRights>
  <Signature>
    <Inputs>
      <Argument name="creationDateStart" type="date">
        <Description>rept_input_cert_report_create_dt_start</Description>
      </Argument>
    </Inputs>
  </Signature>

```

```

</Argument>
    <Argument name="creationDateEnd" type="date">
<Description>rept_input_cert_report_create_dt_end</Description>
</Argument>
</Inputs>
</Signature>
</TaskDefinition>

```

Custom Audit Report Form XML:

```

<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Form PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<Form name="Audit Report Form">
    <Section columns="2" label="Audit Properties" name="customProperties">
        <Field displayName="Creation Start Date" helpKey="Include all AuditEvents">
        <Field displayName="Creation End Date" helpKey="Include all AuditEvents">
    </Section>
</Form>

```

Audit Best Practices

Though auditing can be useful – even critical for record-keeping at times – it must be used carefully. Auditing involves extra database writing activities which will necessarily impact system performance, particularly when applied to actions that happen frequently or in high volumes. The value of the audit records must be weighed against the performance impact of creating them.

The recommended best practice around auditing is therefore to use it sparingly to capture only those events which are required to meet legal compliance or deemed necessary by the organization's auditors or


management. Later, desired-but-not-required auditing can be added separately, evaluating the performance impact of these records against their value to the organization as each is added.

advanced_analytics
audit audit_config austin_configuration configuration

 34 Kudos

Submit Content Feedback

Version history

Revision #: 2 of 2
Last update: Jan 25, 2023 04:27 PM
Updated by: **cathy_mallet** 

[View Article History](#)

Contributors

 **cathy_mallet**

Powered by
Khoros 

- [About](#) [Careers](#) [Support](#) [Trust Center](#) [Security](#) [Privacy](#) [Cookie Notice](#)
- [Terms of Use](#) [Legal](#)



© 2023 SailPoint Technologies, Inc. All Rights Reserved.

[↑ Top](#)