



System Administration

Version: 8.4

Revised: September 2023

This document and the information contained herein is SailPoint Confidential Information

Copyright and Trademark Notices

Copyright © 2023 SailPoint Technologies, Inc. All Rights Reserved.

All logos, text, content, including underlying HTML code, designs, and graphics used and/or depicted on these written materials or in this Internet website are protected under United States and international copyright and trademark laws and treaties, and may not be used or reproduced without the prior express written permission of SailPoint Technologies, Inc.

“SailPoint Technologies,” (design and word mark), “SailPoint,” (design and word mark), “Identity IQ,” “IdentityNow,” “SecurityIQ,” “Identity AI,” “Identity Cube,” and “SailPoint Predictive Identity” are registered trademarks of SailPoint Technologies, Inc. “Identity is Everything,” “The Power of Identity,” and “Identity University” are trademarks of SailPoint Technologies, Inc. None of the foregoing marks may be used without the prior express written permission of SailPoint Technologies, Inc. All other trademarks shown herein are owned by the respective companies or persons indicated.

SailPoint Technologies, Inc. makes no warranty of any kind regarding these materials or the information included therein, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. SailPoint Technologies shall not be liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Patents Notice. <https://www.sailpoint.com/patents>

Restricted Rights Legend. All rights are reserved. No part of this document may be published, distributed, reproduced, publicly displayed, used to create derivative works, or translated to another language, without the prior written consent of SailPoint Technologies. The information contained in this document is subject to change without notice.

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c)(1) and (c)(2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Regulatory/Export Compliance. The export and re-export of this software is controlled for export purposes by the U.S. Government. By accepting this software and/or documentation, licensee agrees to comply with all U.S. and foreign export laws and regulations as they relate to software and related documentation. Licensee will not export or re-export outside the United States software or documentation, whether directly or indirectly, to any Prohibited Party and will not cause, approve or otherwise intentionally facilitate others in so doing. A Prohibited Party includes: a party in a U.S. embargoed country or country the United States has named as a supporter of international terrorism; a party involved in proliferation; a party identified by the U.S. Government as a Denied Party; a party named on the U.S. Department of Commerce’s Entity List in Supplement No. 4 to 15 C.F.R. § 744; a party prohibited from participation in export or re-export transactions by a U.S. Government General Order; a party listed by the U.S. Government’s Office of Foreign Assets Control as ineligible to participate in transactions subject to U.S. jurisdiction; or any party that licensee knows or has reason to know has violated or plans to violate U.S. or foreign export laws or regulations. Licensee shall ensure that each of its software users complies with U.S. and foreign export laws and regulations as they relate to software and related documentation.

Contents

- Using the Administrator Console 1**
 - Manage Task Results 1
 - Manage Provisioning Transaction Results 5
 - Monitoring Your Environment 7

- About Debug Pages 21**
 - Read-Only Access to Debug Pages and Administrative Information 22
 - Auditing Changes Made in the Debug Pages 23

- Partitioning 24**
 - How Partitioning Works 24
 - Loss Limits 25
 - Configuring Partitioning Request Objects 25

- Alerts 27**
 - Alerts Page 27
 - Create Alert Definition 28
 - Edit Alert Definitions 30

- About Data Extract 33**
 - Configuring Data Extraction 33
 - Configuring Data Transformation 35
 - Importing Data Extract and Transform Configurations 49
 - Provisioning a Data Extract Task Destination Queue 49
 - Enabling and Configuring Data Extract 49
 - Setting Up Data Extract Task 50
 - Working with the Data Extract Message Broker 51
 - Running a Data Extract Task 54
 - Viewing Data Extract Task Results 54
 - Validating Data Extract Results 55
 - Intercept Delete Functionality 56
 - Data Extract Auditing 56
 - Sample Transform YAML Config 56

Allowed Properties	62
IdentityIQ Object Model and Usage	64
Using Data Extract with Sensitive Data	65
Rules and Scripts in IdentityIQ	66
Types of Rules and Scripts	66
How Rules Are Created	67
Type Attributes in Rules	67
Using the Rule Editor	67
Rule Re-Use	68
Restricting Access to the Rule Editor	68
Importing Rule XML	69
Sample Rule Files	69
Rule Libraries	69
API Reference for Rules	70
Testing Rules	70
Working With Incident Codes	71

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.

- [Manage Task Results](#)
- [Manage Provisioning Transaction Results](#)
- [Monitoring Your Environment](#)

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 a task is running long and diagnostics are needed.

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

Postponing Tasks

Tasks which are scheduled to run in the future can be postponed for a selected amount of time.

1. Click the **Scheduled** tab to see future scheduled tasks.
2. Click the **calendar** button (Postpone Task) in the Actions column.
3. Choose the date and time when the task should run again and click **Submit**.

The system will automatically prevent any instances of that task from running until the designated time.

After the designated time has passed, the task will resume on its normal schedule. It will not be run immediately at the time specified in the postponement.

Terminating a Task

Tasks which are currently executing can usually be terminated from the IdentityIQ UI.

1. Click the **Active** tab on the **Administrator Console Tasks** page to see currently running tasks.
2. If a task is running longer than expected or otherwise needs to be immediately terminated, click the **X** button in the Actions column to send a terminate request to the task.

All of SailPoint's out-of-the-box tasks have logic in them that will respond to a terminate request and shut down gracefully at the next logical breakpoint. Custom tasks should also be written to handle a terminate request.

This terminate action can also be done from the Task Results page in **Setup > Tasks > Task Results** by right-clicking the task and clicking **Terminate**. However, permissions which grant access to the Administrator Console are different than the permissions required for the **Setup > Tasks** page, so some users may only have access to this feature from the Administrator Console. See the **IdentityIQ Tasks** documentation for more information.

Requesting a Stack Trace

When troubleshooting abnormal task behavior (most notably, long-running tasks), it may be helpful to get a stack trace of the task's current execution.

1. Click the **Active** tab of the **Administrator Console Tasks** page to see currently running tasks.
2. Click the `</>` button in the **Actions** column on any running task row to generate a stack trace for that task. A dialog box appears, to confirm that the stack trace has been requested and will be available soon.
3. The stack trace will be generated and the `</>` button will be colored green when it is available.
4. Click the green `</>` button to view the stack trace in a dialog box. You can generate another stack trace for the same task, if it is still executing, with the **Request New Stack Trace** button in that dialog box.

Note that workflows also generate task result objects which, are included on this page. Since the Stack Trace option does not apply to a workflow, this option is not enabled for those entries in the list.

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.

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.

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

Completed Tab

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

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

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.

To access the Provisioning Transaction table, click the **gear** icon > **Administrator Console** > **Provisioning**.

Four pages of data are available for review: **All**, **Failure**, **Success**, and **Pending**. The first page (All), which is displayed by default, shows all logged provisioning transactions. The other three pages display transactions which ended in failure, success, and pending statuses (retry) respectively, depending on the logging configuration, which is described in the section below.

Managing the Contents of the Provisioning Transaction Table

By default, the Provisioning Transaction table only displays transactions that have resulted in a failure condition, but it can be configured to display all provisioning transactions which are processed through IdentityIQ.

To change this default, change the logging level for Provisioning Transactions:

1. Click the **gear** icon > **Global Settings** > **IdentityIQ Configuration** > **Miscellaneous**.
2. In the **Provisioning Transaction Log Settings** section, change **Maximum Log Level** to **Retry** or **Success**.
 - **Retry** means the system will log provisioning transactions that return a Failure result or a Retry result (an error message indicating a temporary condition that means a later retry of the provisioning operation will likely succeed and should therefore be auto-retried after a delay interval).
 - **Success** means the system will log all provisioning transactions, regardless of their provisioning result status values.
 - Days before provisioning transaction event deletionNote.
3. Optionally, set a **Days before provisioning transaction event deletion** value. If you set the **Maximum Log Level** to **Success**, the provisioning transaction log will record high volumes of records. Therefore, it is

particularly important in that case to also set the **Days before provisioning transaction event deletion** value to the number of days you want to retain these records so they will be automatically purged after that time. *Leaving this value as the default "0" means these records will never be deleted*, which can fill your database quickly. Even when using a Retry or Failure maximum log level, this value should be set to purge records you no longer need.

4. To turn off provisioning transaction logging entirely, clear the **Enable Provisioning Transaction Log** box.
5. **Save** your changes.

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.

Reporting on Provisioning Transactions

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

The information on this page is also available in two reports available from the **Intelligence > Reports** menu: the **Provisioning Transaction Object Report** and the **Detailed Provisioning Object Report**. See the **Reports** documentation for more information.

Click the information icon for any transaction to view detailed information. The Transaction Details window displays all available details about the selected provisioning transaction, including attribute-level information about the request and any applicable error messages.

The Transaction Details window provides very detailed information, including the reasons for a Failed or Pending status. After viewing, you can take the appropriate actions to correct the reasons for the failure or delay.

Forcing Retries for Retryable Errors

Retryable errors are recorded as **Pending** transactions in the Provisioning Transactions table. Though these transactions will automatically retry after a configured interval, an administrator can override the retry interval and force a transaction to retry immediately. For example, suppose a network error causes a retryable error. Ten minutes later, the problem is fixed but the wait period on the retry is set for the transaction to retry after two hours. The administrator can speed up the process by forcing the retry to occur immediately after the network issue is resolved.

For any Pending transaction, a **Retry** button is shown in the **Actions** column. These retryable transactions appear on both the **All** page and the **Pending** page, and can be forced to retry from either page.

To force an immediate retry of a pending transaction, click its **Retry** button. The **Retry Launched** pop-up window appears, confirming that the retry has begun.

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

Overriding Failed Transactions with Manual Work Items

Transactions which appear as **Failures** are considered permanent failures and will never be retried automatically by the system. Likewise, they cannot be forcibly retried from the Provisioning Transactions table user interface. However, they can be reprocessed manually by creating a manual work item assigned to a person to complete the request outside of IdentityIQ's automated operations. In short, the transaction reprocessing is done just as if it were a provisioning request for a disconnected system (an application with no automated provisioning channel from IdentityIQ).

The administrator can view failed transactions on the **All** and **Failure** pages. To reprocess a failed transaction through a manual work item, click **Override** in the **Actions** column.

Next, choose the user to whom the manual work item should be assigned, and enter comments to communicate with that person; the comments are included in the manual work item. The choices for assignee are the **Application Owner** for the target application, yourself, or **Other** (another user). If you choose **Assign to Other**, you will be prompted to select the target user from a list.

Finally, click **Override** on this pop-up window to complete the failure override process and generate the manual work item. An Override Success message appears, confirming that the override has been completed and the manual work item has been generated.

The original transaction remains in the Provisioning Transactions table with a **Failure** status, for historical information purposes, though it no longer offers the Override option because the override has already been done. A new Provisioning Transactions table entry gets created for the override manual work item. Both the new manual record and the failed auto record reflect the number, so they can be connected through that information as needed.

Note: Just like with manual work items for provisioning to disconnected systems (systems with no automated provisioning channel from IdentityIQ), override work items are treated as successfully completed in the Provisioning Transactions table when the manual work item generated. Consequently, the **Success** override transaction appears in the Provisioning Transactions table immediately.

Monitoring Your Environment

The Environment Monitoring console in the Administrator console provides insight into each defined application's health and the status of your modules and extensions via a single dashboard. This helps administrators of both on-premises and cloud-based installations get insights into a number of environment statistics for both hosts and applications, and diagnose connectivity issues within the environment.

Access the Environment Monitoring console by following these steps:

1. Select the **gear icon**.
2. Select **Administrator Console**.
3. From the left-hand menu, select **Environment**.

Use the **gear icon** on the right side of the title bar to define global settings for all hosts in IdentityIQ. These settings are used for all hosts that have not explicitly overridden the defaults.

The Environment console has tabs for Hosts, Applications, and SailPoint Modules and Extensions.

On each tab, select the **Columns** button at the upper right side of the page to choose and arrange the information displayed on the pages. When you select the **Add Column** button, any other available columns appear as dropdown options. Use the search field to locate specific servers.

For details about each tab, see:

- [Hosts](#)
- [Applications](#)
- [SailPoint Modules and Extensions](#)

Hosts

The Hosts tab in the Environment console displays all of the hosts associated with an IdentityIQ instance, along with statistics about the state of each host in the selected column.

A search box in the upper right lets you search for servers. This is a "starts with" search rather than a full-text search, and it is not case-sensitive.

You can add, remove, or reorder the columns on this tab. Columns may include:

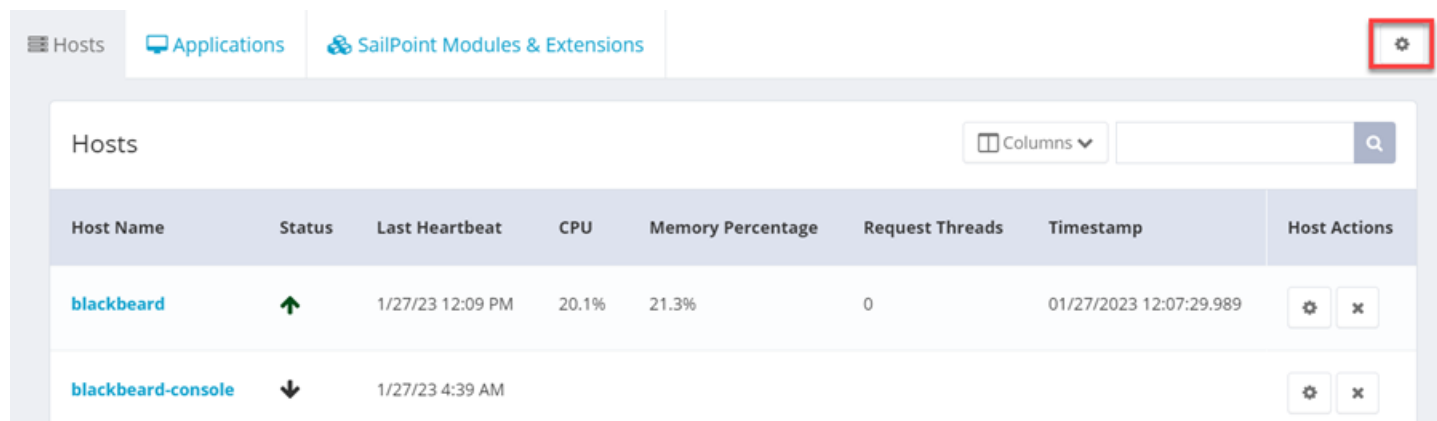
- **Host Name** – name of the machine hosting the IdentityIQ instance.
- **Status** – a color-coded arrow icon indicating whether the host is up or down.
- **Last Heartbeat** – the last time the host verified its functionality. Each host periodically sets a date in the server object to verify it is still running. If the verification date does not advance for a specified period of time, the other servers detect and mark the server as having crashed.
- **CPU** – a percentage representation of the workload sustained by the host machine CPU.
- **Memory Percentage** – the percentage of the maximum allowed memory which is actively in use by the application server's Java virtual machine.

- **Request Threads** – the number of request processor threads running on this host.
- **Timestamp** – the last time the monitoring service ran.
- **Database Response Time** – the number of milliseconds it last took to connect to the database and query from it.
- **Task Threads** – the number of background tasks running on this host.
- **Memory** – the amount of memory (in bytes) actively in use by the application server's Java virtual machine.
- **Host Actions** – action buttons to configure or delete hosts.

Note: For hosts running on Unix operating systems, there is an option, Open Files—Unix Only, that displays the number of open files to avoid running out of file handles. This count is on the operating system, not the application server. This column can only be enabled manually from the UI Configuration object on the debug pages.

Global Host Configuration

You can set global Environment Monitoring options for all your hosts by selecting the gear in the title bar. Settings you choose here are used for all hosts, but you can also set specific options for individual hosts (described below) to override many of these global settings on a per-host basis.







These are the options you can configure globally for your hosts:

- **Polling Interval** – the number of seconds between monitoring cycles.
- **Statistics Retention** – the number of days monitoring statistics will be retained. Enter a zero here to retain statistics indefinitely.

- **CPU** – a percentage representation of the workload sustained by the host machine CPU.
- **Task Threads** – the number of background tasks running on this host.
- **Request Threads** – the number of request processor threads running on this host.
- **Database Response Time** – the number of milliseconds it last took to connect to the database and query from it.
- **Memory** – the amount of memory (in bytes) actively in use by the application server's Java virtual machine.
- **Memory Percentage** – the percentage of the maximum allowed memory which is actively in use by the application server's Java virtual machine.

Host-Specific Configuration and Monitoring

Each host can be configured by clicking the **gear** in the **Host Actions** column for that host.

Host Name	Status	Last Heartbeat	CPU	Memory Percentage	Request Threads	Timestamp	Host Actions
blackbeard	↑	1/27/23 12:09 PM	20.1%	21.3%	0	01/27/2023 12:07:29.989	 
blackbeard-console	↓	1/27/23 4:39 AM					 

When you select the gear icon, the Host Setting dialog appears with two tabs, Services and Configuration, enabling you to specify the services running on each host and configure host monitoring.

Services

The Services tab lets you toggle to enable or disable the services running on a specific host. Many of these services can be managed through the UI. The one exception is the Request service, which cannot be fully shut down.

When a service is shut down, the host still processes requests that have been specifically targeted to that host, but does not pick up generic, or untargeted, requests.

Monitoring Service

The Monitoring service captures and saves the latest statistics, promotes any statistics-related extended attributes to the Server object, and prunes any statistics that have outlived the configured retention period. The Monitoring service

populates the **task threads**, **request threads**, and **cpu usage** statistics as well as statistics for **memory usage**, **memory usage percentage**, and **database response time**.

Reanimator Service

The Reanimator service helps manage "hung" tasks, or tasks that appear to be running even though they have stopped. An un-partitioned task can sometimes fail without properly updating the state of the TaskResult. This can leave the task in a hung state. The most common causes for this kind of issue are a temporary loss of connection to the database or a brief database server failure. Regardless of the underlying reason, the Reanimator service performs the task of resetting requests or tasks so they can resume.

The service can also help with terminating a task or request that is not configured to resume upon being orphaned or failing. If the task or request is not configured to resume, then when the service detects a task in a hung state, it automatically marks it as terminated.

Note: If you are using Email Task Alerts in any of your task definitions, be aware that when a task hangs and is restarted by the reanimator service, email task alert notifications may not be sent.

The Reanimator service runs by default on all hosts. Although it is unlikely that you would need to switch it off, it is possible to do so. For example, if you have a dedicated UI host, you might not need this service running there.

To disable the Reanimator service on a specific host:

1. On the **Hosts** tab, select the **gear** icon beside the specific host on which you want to disable the service.
2. On the **Services** tab of the Host Configuration dialog, use the slider to switch off the Reanimator service.

Configuration

The Configuration tab lets you set up host-specific monitoring configuration. What you set here overrides the global defaults for Polling Interval, Statistics Retention, and to enable and disable given retained statistics, such as CPU or task threads.

The Configuration tab also lets you select applications to monitor. Any applications you select for monitoring are contacted each time the monitoring service runs and a health check status is recorded.

The **Use Default Settings** button at the bottom of the Configuration tab clears your Monitoring Configuration for this host and reverts this host to the global defaults. See *Monitoring Applications*.

Note: When you restore default settings, any applications you have selected for Application Monitoring are retained; restoring default settings does not remove any applications you have selected for monitoring.

Note: Administrators can assign a given host to handle specific types of requests. See *Specifying Hosts to Handle Requests*.

Viewing Host Statistics

Select a name in the Host Name column to show all server statistics captured for that host, grouped in time-based Statistic Snapshots. A host's snapshots can be cycled using the previous and next arrows to the left and right of the header or individually selected using the dropdown list.

Choosing Which Host Statistics to Display

You can choose which statistics to show on the Hosts tab and in which order they appear.

1. To edit what is shown on the Hosts tab, select the **Column** button.
2. To add columns, select **Add Column** and choose the new column(s) to include.
3. To change the order of the statistics, use the **arrow buttons** on each column tile, or **drag the tiles** into the order you prefer.
4. To remove a column from the Hosts tab, select **Remove** on the column tile.
5. **Save** your changes before exiting.

Deleting Hosts

Use the **X** button in the **Host Actions** column to delete hosts. Deleting a host removes the associated Server object and any related configuration or statistics for the given host.

The host no longer appears in the list of hosts after deletion. However, if the underlying server is still running, the host will reappear the next time its heartbeat service runs. All configuration settings for a regenerated host use the defaults for its list of services and for the monitoring service configuration.

Specifying Hosts to Handle Requests

In a multi-host environment, you can specify which hosts can process specific types of requests, by including a single host or list of hosts in a RequestDefinition object. This can help with performance, allowing you to dedicate specific machines and threads for processing request types that are, for example, operating at a high volume or require more resources.

1. When logged in as an administrator, select the **wrench** icon dropdown at the top of the screen, then select **Object**.
2. In the Debug pages, choose RequestDefinition in the **Select an Object** field of the Object Browser.

3. From the list of RequestDefinition objects, select the object that you want to modify in order to specify a host or list of hosts.
4. Add host entry to the RequestDefinition attribute map. The value can be set to a single host, or multiple hosts separated by commas. For example:

```
<Attributes>  
  <Map>  
    <entry key="hosts" value="hostA,hostB,hostC"/>  
    <entry key="maxThreads" value="1"/>  
  </Map>  
</Attributes>
```

5. Select **Save** to save the updated RequestDefinition object or **Close** to cancel changes.

Applications

The Application tab in the Environment console provides a view from an application-up perspective. Application monitoring is configured **per host**, giving each a list of applications you want that specific host to monitor. Each application is listed, along with its type and a summary of all statuses reported by all configured hosts. Monitoring can be run from any number of servers on any subset of applications.

Select an application name to display a panel containing detailed information about the application's statuses. The panel shows each host that has reported a status for the application, as well as the status – up or down – and time of the most recent ping.

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

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

Global Application Configuration

You can set global monitoring options for all your hosts and applications by selecting the gear in the title bar. Settings you choose here are used for all hosts and associated applications in the same way.

Application Name	Type	Status
ADDirectDemodata	Active Directory - Direct	0 ↑ 0 ↓
Active_Directory	Delimited File Parsing Connector	0 ↑ 0 ↓

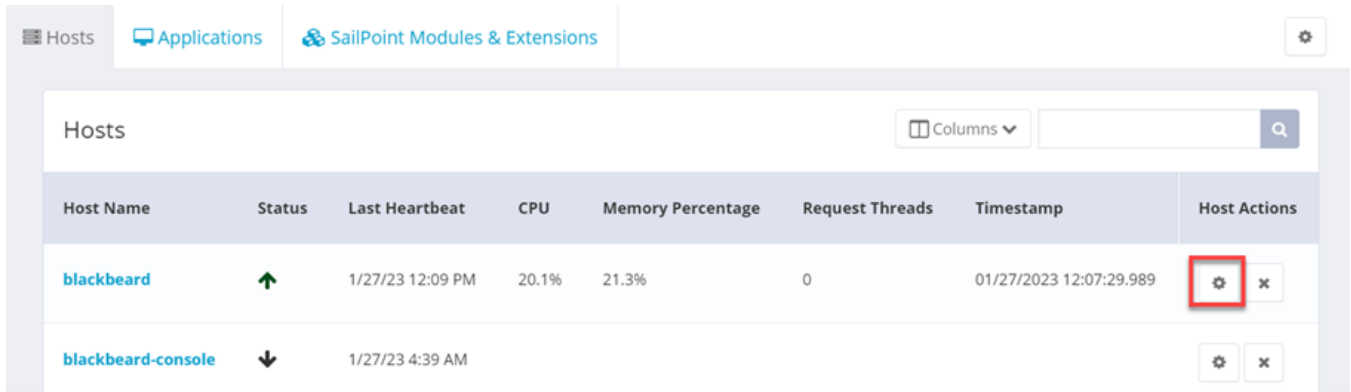
These are the options you can configure globally:

- **Polling Interval** – the number of seconds between monitoring cycles.
- **Statistics Retention** – the number of days monitoring statistics will be retained. Enter a zero here to retain statistics indefinitely.
- **CPU** – a percentage representation of the workload sustained by the host machine CPU.
- **Task Threads** – the number of background tasks running on this host.
- **Request Threads** – the number of request processor threads running on this host.
- **Database Response Time** – the number of milliseconds it last took to connect to the database and query from it.
- **Memory** – the amount of memory (in bytes) actively in use by the application server's Java virtual machine.
- **Memory Percentage** – the percentage of the maximum allowed memory which is actively in use by the application server's Java virtual machine.

Note: For hosts running on Unix operating systems, there is an option, **Open Files – Unix Only**, that displays the number of open files to avoid running out of file handles. This count is on the operating system, not the application server. This column can only be enabled manually from the UI Configuration object on the debug pages.





Setting Up Application Monitoring

To set up an application to be monitored, navigate to the **Hosts** tab, then complete the following:



Hosts

Columns

Host Name	Status	Last Heartbeat	CPU	Memory Percentage	Request Threads	Timestamp	Host Actions
blackbeard	↑	1/27/23 12:09 PM	20.1%	21.3%	0	01/27/2023 12:07:29.989	 
blackbeard-console	↓	1/27/23 4:39 AM					 

1. On the Hosts page, select the **gear** icon in the Host Actions column for the host that you want set up to monitor applications.
2. Select the **Configure** tab for that host.
3. In the **Application Monitoring** section, choose the applications to monitor; you can type in application names, or use the dropdown listing browse for applications. You can choose as many hosts as you like.

Host Settings

Services Configuration

General

Polling Interval
Number of seconds between monitoring cycles
300

Statistics Retention
Number of days monitoring statistics should be retained
7

Application Monitoring

Monitored Applications

ADDirectDemodata
Active_Directory
AdminsApp
Composite_ERP_Global_App_Users
Composite_ERP_Global_DB

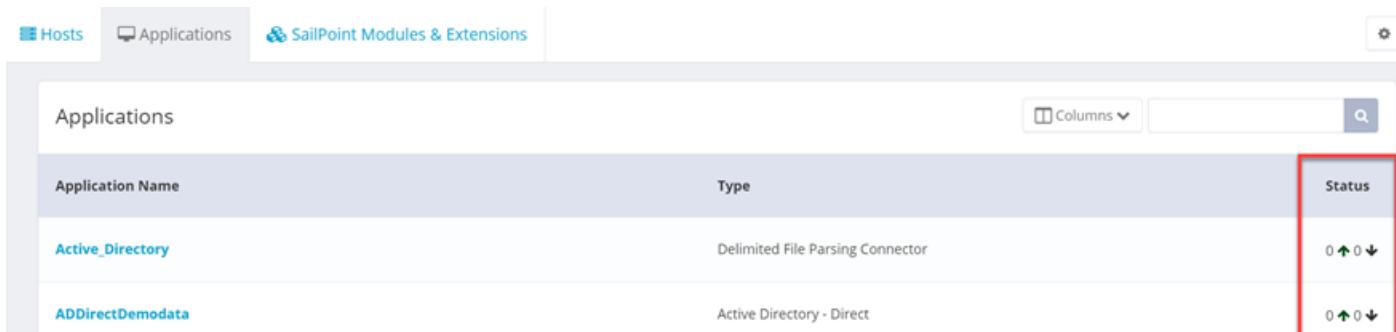
Request Threads

4. Select **Save**.

Monitoring Applications

1. Select the **gear menu > Administrator Console > Environment**.
2. Select the **Applications** tab.

3. All your applications are listed. The applications that are being monitored have **numbers in the status column**, indicating the number of hosts showing the application as being responsive (up arrow) and the number of hosts showing the application as down (down arrow). To search for specific applications, use the **Search** box.



4. Select an application name to see details:
 - **Status** – an arrow indicates whether the application is up (running) or down.
 - **Last ping** – a timestamp of the most recent ping to the application.
5. To re-ping the application and refresh the status, select the **Refresh** icon.


Environment Monitoring Objects

The IdentityIQ object model includes some objects related to Environment Monitoring.

Monitoring Statistic Object

An object type, MonitoringStatistic, defines what you can monitor on all your hosts. The objects of this type do not hold any of the monitoring data; they simply define which statistics can be monitored. You can customize the Environment Monitoring UI via changes to this object, that is, by adding or removing individual MonitoringStatistic objects.


Object Browser

Object Browser				
MonitoringStatistic	▼	Filter by Name or ID		Configuration Objects ▼
<input type="checkbox"/>	Id	Name ▲	Created	Modified
<input type="checkbox"/>	ac10046e86021515818602355c380134	applicationResponseTime	1/30/23 4:24 AM	
<input type="checkbox"/>	ac10046e86021515818602355c130131	cpu	1/30/23 4:24 AM	
<input type="checkbox"/>	ac10046e86021515818602355c420135	databaseResponseTime	1/30/23 4:24 AM	
<input type="checkbox"/>	ac10046e86021515818602355c490136	memoryUsage	1/30/23 4:24 AM	
<input type="checkbox"/>	ac10046e86021515818602355c520137	memoryUsagePercentage	1/30/23 4:24 AM	
<input type="checkbox"/>	ac10046e86021515818602355c570138	openFileCount	1/30/23 4:24 AM	
<input type="checkbox"/>	ac10046e86021515818602355c2b0132	quartzThreads	1/30/23 4:24 AM	
<input type="checkbox"/>	ac10046e86021515818602355c320133	requestProcessorThreads	1/30/23 4:24 AM	

Server Statistic Object

Objects of type ServerStatistic contain the statistics for monitored hosts. The number of ServerStatistic objects you can see in the Debug pages varies depending on how long you have configured Environment Monitoring to retain statistics and how frequently the monitoring service runs.

Object Browser

Object Browser				
ServerStatistic	▼	Filter by Name or ID		Configuration Objects ▼
<input type="checkbox"/>	Id	Name ▲	Created	Modified
<input type="checkbox"/>	ac10046e860210c281860245afeb1453	cpu	1/30/23 4:42 AM	
<input type="checkbox"/>	ac10046e860210c2818602580f191473	cpu	1/30/23 5:02 AM	
<input type="checkbox"/>	ac10046e860210c2818602537975146b	cpu	1/30/23 4:57 AM	
<input type="checkbox"/>	ac10046e860210c28186024ed8371463	cpu	1/30/23 4:52 AM	

Global ServiceDefinition for Monitoring

The ServiceDefinition object contains the default monitoring configuration for all servers; you can override the configuration settings defined here on a server-by-server basis by settings in the monitoringConfig entry in the Server object, as described in the section below.

Object Browser

Object Browser				
ServiceDefinition		Filter by Name or ID	Q	Configuration Objects
<input type="checkbox"/>	Id	Name ^	Created	Modified
<input type="checkbox"/>	ac10046e8602151581860235d28f02cc	BundleProfileRelation	1/30/23 4:25 AM	1/30/23 2:31 PM
<input type="checkbox"/>	ac10046e8602151581860235f543035c	CAMSync	1/30/23 4:25 AM	
<input type="checkbox"/>	ac10046e8602151581860235d51102d9	FullText	1/30/23 4:25 AM	1/30/23 4:25 AM
<input type="checkbox"/>	ac10046e86021515818602355c70013b	Heartbeat	1/30/23 4:24 AM	
<input type="checkbox"/>	ac10046e86021515818602355c75013c	Monitoring	1/30/23 4:24 AM	1/30/23 4:25 AM
<input type="checkbox"/>	ac10046e86021515818602355c85013f	PluginSync	1/30/23 4:24 AM	
<input type="checkbox"/>	ac10046e86021515818602355c890140	Reanimator	1/30/23 4:24 AM	

Server-Specific Monitoring Settings in the Server Object

Each server can define its own specific monitoring configuration. If a monitoring configuration is defined in the **Server** object, it will override the global default configuration for this host. The Server object can include an entry named **monitoringConfig** in its attribute Map.

```
<entry key="monitoringConfig">
  <value>
    <Map>
      <entry key="retentionPeriodDays">
        <value>
          <Integer>30</Integer>
        </value>
      </entry>
      <entry key='monitoringStatistics'>
        <value>
          <List>
            <String>cpu</String>
          </List>
        </value>
      </entry>
    </Map>
  </value>
</entry>
```

```
</value>  
</entry>
```

Application Monitoring in Server Objects

Information about applications that are being monitored is stored in the Server object for each host, in the **monitoredApplications** entry key that is part of the **monitoringConfig**:

```
<entry key="monitoringConfig">  
  <value>  
    <Map>  
      <entry key="monitoredApplications">  
        <value>  
          <List>  
            <String>ADDirectDemodata</String>  
            <String>Composite_ERP_Global_App_Users</String>  
            <String>Active_Directory</String>  
            <String>Composite_ERP_Global_DB</String>  
          </List>  
        </value>  
      </entry>  
    </Map>  
  </value>  
</entry>
```

SailPoint Modules and Extensions

The SailPoint Modules and Extensions tab provides a list of all installed modules and extensions, a brief description, and a summary of all statuses reported.

Select a module or extension name to see a list of reported statuses and the corresponding Last Ping timestamp.

When the **Problem Detected** icon is present, select it for more information about the nature of the problem.

About Debug Pages

The Debug pages provide administrative functions, including advanced object editing capabilities for IdentityIQ system administrators, as well as a place to find detailed information about your IdentityIQ installation.

Debug pages include:

- **About** – basic information about the IdentityIQ installation such as server host, the current IdentityIQ version and schema version, locale and time zone for the client, and an array of Java system properties.
- **Object** – the most frequently used feature of the Debug pages, providing access to the XML representations of data and configuration objects in the IdentityIQ system
- **Memory** – view the Free, Total, and Max JVM memory
- **Caches** – five buttons for managing the Hibernate Caches
- **Count** – count of objects by class
- **Beans** – the Managed Beans page displays all JMX beans registered with the application server
- **Threads** – view all Java threads and their current state; a good resource for diagnosing performance problems
- **Call Timings** – statistics about the time taken to perform certain database activities
- **Logging** – view or change the path to the Log4j properties file
- **Database** – view basic database pooling information
- **Connections** – see the connections from the database connection pool that the server is currently using
- **ActiveMQ Monitoring** – view broker information, connectors, broker statistics, destinations, and subscriptions

See the **Debug Pages**Compass article for details about each page.

On Debug's Object Browser page, you can view and edit XML representations of data and configuration objects in the IdentityIQ system. The Debug pages also provide UI-based access to many of the actions you can run in the IdentityIQ console.

Debug pages are accessed via the wrench menu, if available, or a hidden URL: `http://[hostname:port]/[contextroot]/debug` (for example, `http://localhost:8080/identityiq/debug`) and can only be used by those with the SystemAdministration capability. As of IdentityIQ version 8.4, users may be granted read-only access rights to view the Debug pages without taking actions or making changes. See [Read-Only Access to Debug Pages and Administrative Information](#)

For more details about Debug, see **Read-Only Access to Debug Pages**, and the **Debug Pages** article on Compass.

Caution: There is no rollback mechanism for edits that are saved in the Debug pages. For this reason it is strongly recommended that you export an object (the process is described in the **Debug Pages** article on Compass), copy it, make changes to the copy, and then import the updated version, rather than editing objects directly in the Debug page.

Read-Only Access to Debug Pages and Administrative Information

System administrators may grant the capability Debug Pages Read Only Access to non-admin users. Those who are granted standard read-only access to IdentityIQ Debug pages are able to view not just the Object Browser, but also the other pages that have traditionally been reserved for system administrators, including About, Memory, Caches, Count, Beans, Threads, Call Timings, Logging, Database, Connections, and ActiveMQ Monitoring.

Read-only access allows users to see the XML code for configured objects. They can copy or download the XML, but cannot save changes or upload.

Many of the Debug pages have actionable options that are only enabled for those with system administrator capabilities. For those with read-only access, options such as Run Rules and Run Garbage Collector are present but disabled.

Important: Read-only access allows users to view all objects in the Object Browser; this may include some sensitive data. Keep this in mind when determining who should have access.

Configuring Fine-Grained Read-Only Debug Access

There are multiple individual rights that control fine-grained access. Your installation of IdentityIQ is configurable and administrators can group a subset of read-only debug rights into capabilities appropriate for your organization's needs. For example, if you have a DBA who only needs to see a subset of Debug items related to databases, then use the more fine-grained rights to create a custom capability.

Important: Read-only access allows users to view all objects in the Object Browser; this may include some sensitive data. Keep this in mind and consider configuring custom capabilities as needed.

See **Rights and Capabilities for Identities**.

View-only access to the Debug pages controls whether or not you can see the page – if users try to reach a page without having either system administrator capabilities or read-only access rights, they see an Access Denied message.

Useful When

Read-only access is useful when developers need to see IdentityIQ objects' XML in order to write custom code, troubleshoot, or debug without a system administrator partnering with them. They can open any object on the Object Browser.

Auditing Changes Made in the Debug Pages

When enabled, changes made on Debug pages may be audited. To enable logging, navigate to **gear > Global Settings > Audit Configuration > General Actions** and select the **Debug Object Browser Change** checkbox.

Audit items log the following information:

- Date/time
- Source – identity that made the change
- Target – object class that was edited such as identity, bundle, configuration, etc.

The audit log does not detail what the changes were. Use your organization's internal versioning or tracking if you need to track the specific changes made.

To view Debug changes:

1. Navigate to **Intelligence > Advanced Analytics**.
2. Select **Audit** from the Search Type dropdown.
3. Under Search Criteria > Audit Attributes, select the Action dropdown, then select **DebugObjectBrowserChange**.
4. Select the **Run Search** button.
5. Audit results may be exported in pdf, csv, or cef formats by selecting the corresponding button at the upper right side of the search results.

Partitioning

Partitioning is used to break operations into multiple pieces, or partitions, allowing data processing to split across multiple hosts, and across multiple threads per host. The overall goal with partitioning is to increase processing throughput and speed.

Partitioning is supported for account and account group aggregation tasks, for identity refresh tasks, for generation of manager and targeted certifications, and for role propagation.

Some connector types support partitioning at the application level. To use partitioning for account and account group aggregation, you must configure the application for partitioning, and enable partitioning when defining an account aggregation or identity refresh task.

If partitioning is enabled in an aggregation task that is acting on a connector application that does **not** support partitioning, partitioning will take place at the IdentityIQ database level and not at the connector level; this is referred to as **generic partitioning** or **task-level partitioning**.

- Applications are configured as part of the Account Settings on the Configuration tab of the Application Configuration page. See the **Application Configuration** documentation.
- For task details, see the **Tasks** documentation.
- For certification details, see the **Certifications** documentation.

Note: Partitioning is not available on all tasks or certifications. Partitioning is available for Account Aggregation, Account Group Aggregation, Identity Refresh, Perform Identity Request Maintenance, and Perform Maintenance tasks, and for Manager and Targeted Certification generation. Partitioning is also not available on all application types. Partitioning is controlled by both the configuration of the applications you are using and the configuration of the applications used to communicate with those applications.

How Partitioning Works

Each partition is 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, each instance must be assigned a unique `iiq.hostname` and have a unique Server object.

The Server objects include a heartbeat service that 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 [Using the Administrator Console](#).

See also:

- [Loss Limits](#)
- [Configuring Partitioning Request Objects](#)
- [Partitioning Best Practices](#) on SailPoint's Compass community.

Loss Limits

Some of the features which support [Partitioning](#) also include an option to set loss limits for the identities or accounts being processed by a task. The loss limit sets the maximum number of identities or accounts that will be reprocessed in case of a sudden termination of a partitioned refresh.

In a partitioned task, each time the task reaches the loss limit – that is, it has processed a number of accounts or identities that match the value of the loss limit – it commits a list of the accounts to a `requestState` object. If the task should happen to fail, due perhaps to a server or database going down, the task will check the `requestState` object when it resumes, so that it knows which accounts have already been processed. This means the task doesn't have to re-process the entire partition. A lower loss limit number will result in less duplicated work following a crash, but may slow down the task due to increased database contention.

Loss limit data that is stored in the `requestState` object is base-64 encoded and so is not human-readable.

RequestState objects are not retained in the IdentityIQ database past their usefulness; in other words, once a loss limit has been reached, the object for that particular segment is automatically deleted.

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** dropdown list.

Alerts

Alerts are created using IdentityIQ File Access Manager (FAM) based on activity data – actions users take on resources that are part of an application that FAM is monitoring. FAM can be configured to create alerts when the user action is considered unexpected, potentially risky, or inappropriate. It is possible to configure alerts for any behavior. You can choose to use this functionality more broadly (e.g. for non-risky or non-problematic activities that someone wants to use as a process trigger).

This integration additionally enables you to trigger actions in IdentityIQ in response to an alert. Specifically, alerts aggregated into IdentityIQ can be used to drive three different response actions. A single alert can trigger more than one response action:

- Launch a certification
- Launch a workflow
- Send an email notification

See:

- [Alerts Page](#)
- [Alert Definitions Page](#)
- [Edit Alert Definitions](#)

Alerts Page

Use the Alerts page to view existing alerts for your enterprise. To limit the number of alerts displayed in the table, use the filtering options.

Alert Page Details

Name

The name of the alert.

Source

Application associated with the alert.

Native Id

Native identifier of the application with which the alert is associated.

Type

Alert type.

Target Type

Type of the object that triggered the alert.

Target Name

Name of the object that triggered the alert.

Alert Date Start

Date and time at which this alert was triggered.

Alert Date End

Date and time at which this alert expires.

Last Processed Start

Last date and time this alert was triggered.

Last Processed End

Last date and time this alert process finished.

Acted Upon

Select True if this alert matched an alert definition and an alert action was triggered.

Create Alert Definition

The Create Alert Definition page contains the following information:

Details

Name

A descriptive name of this alert. This is the name that displays on the Alerts page.

Display Name

Label that is displayed on the alert.

Description

A brief description of the alert.

Owner

The alert owner, not necessarily the identity who triggered the alert.

Match Rule

Enables more complex matching logic.

+Add

Option to add a **Match Term**.

Source

Application name that triggers the alert.

Attribute

The display name of an account attribute derived from the attribute and its associated application.

Value

Value for the selected attribute that will trigger an alert during alert processing.

Action Type

Action to be taken when the alert is created. This can either be a notification, certification, or a workflow, or a combination of the available actions.

Email Template

Template used for the notification email. If none is selected, a system default is used.

Email Recipients

List of users to receive the alert notification.

How to Create an Alert Definition

Alerts are created using the Alert Definitions tab. Use this procedure to create new alert.

Procedure

1. Click the **Alert Definitions** tab on the Alerts page.
2. Click **+New**.
3. Enter the alert information.
4. Click **Save** to save the alert and return to the Alerts page.

Edit Alert Definitions

Use the Edit Alert Definition to edit existing rules. The Edit Alert Definitions page contains the following information:

Name

A descriptive name of this alert. This is the name that displays on the Alerts page.

Display Name

Label that is displayed on the alert.

Description

A brief description of the alert.

Owner

The alert owner, not necessarily the identity who triggered the alert.

Source

Application name that triggers the alert.

Attribute

The display name of an account attribute derived from the attribute and its associated application.

Value

Value for the selected attribute that will trigger an alert during alert processing.

Match Rule

Enables more complex matching logic.

Action Type

Action to be taken when the alert is created. This can either be a notification, certification, or a workflow, or a combination of the available actions.

Workflow

Defines the workflow structure and steps involved in the workflow processing.

Email Template

Template used for the notification email. If none is selected, a system default is used.

Email Recipients

List of users to receive the alert notification.

[How to Edit an Alert Definition](#)

[How to Filter Alerts](#)

How to Edit an Alert Definition

Alerts are edited in the **Alert Definitions** tab. Use this procedure to edit existing alerts.

Procedure

1. Click the **Alert Definitions** tab on the Alerts page.
2. Select an alert and click **Edit** in the Actions column.
3. Enter the alert information.
4. Click **Save** to save the alert and return to the Alerts page.

How to Filter Alerts

Use the filtering options to limit the number of alerts displayed in the table. You can filter by any field. Use this procedure to filter through existing alerts.

1. Click the **Alert** tab on the Alerts page.
2. Click **Filter**.
3. Enter filtering criteria, such as type, source, and alert start and end dates.

4. Click **Apply** to save the filter options. The filter button turns green to alert you that filtering conditions have been applied.
5. To clear filters, click the Filter button to open the filter criteria section, and click **Clear**.

About Data Extract

Data Extract allows organizations to periodically extract data from the IdentityIQ database and store it in a format that common business intelligence (BI) tools can use. IdentityIQ administrators create and configure a Data Extract Task, which calls the functionality to extract and transform data and defines the message destination, a queue where data is available to be picked up by BI systems.

Note: Nothing in IdentityIQ consumes the messages from an ActiveMQ message broker, but customer organizations can write their own client to do so.

Administrators can configure criteria for both the extraction and transformation tasks to customize which types of objects are extracted and define which properties of those objects to include.

Data extract also tracks sufficient context to keep a log of the extraction events.

Running Data Extract requires you to complete the following:

1. [Configuring Data Extraction](#)
2. [Configuring Data Transformation](#)
3. [Importing Data Extract and Transform Configurations](#)
4. [Provisioning a Data Extract Task Destination Queue](#)
5. [Enabling and Configuring Data Extract](#)
6. [Setting Up Data Extract Task](#)

Configuring Data Extraction

Data extraction defines what to extract. It is configured by creating an Extract YAMLConfig and importing it to IdentityIQ. You may use the IdentityIQ console to generate extract configurations, then modify them as needed. See **IdentityIQ Console Commands**.

Data Extract yields image objects that contain imageFields with data in them, as well as metadata about the image, such as the object type it represents and the timing of the image. The ObjectSelector is used to select objects for extraction. Administrators define key data in the extract configuration, including:

- Types of objects to extract and transform
- What attributes to include or exclude for each object type

- Filter criteria for selecting objects
- Name of the queue to publish to
- Name of the transform configuration

Note: You can exclude any property or attribute for any object you have defined from going through Data Extract.

Note: Attributes that are known to be encrypted or secret cannot be extracted by default. If your implementation is storing an Extended Attribute defined by your organization and if you define that attribute as part of a Data Extract YAMLConfig, then it will be extracted as normal. If an encrypted attribute were included, it would be extracted in its encrypted state unless you take extra steps to decrypt it. See Using Data Extract with Sensitive Data.

Major classes matching the objects found in the Debug Object browser can be extracted. Best practice is to exclude AccessHistory classes and Intercepted Deletes from extraction, although this is not enforced by IdentityIQ. The following objects are not extractable in any circumstance:

- AuthenticationAnswer
- RemoteLoginToken
- SAMLToken

Make sure the extractedObjects from the Extract YAMLConfig all have a corresponding imageConfigDescriptor and that each has a valid objectClassName.

The extract configuration may look like this example:

```
1extractedObjects:
2  identity:
3    resultLimit: 10
4  role:
5  certification:
6    filterString: phase ==
Certification.Phase.End
7transformConfigurationName:
ExtractTransformConfig
8messageDestination: extractedObjectsQueue
```

transformConfigurationName – the name of the configuration object describing the transformation of the objects extracted.

messageDestination – the name of a queue to place the message into. You may use a script if you want to dynamically decide what queue to write to. It is important to correctly define a message destination; without something reading from the queue, it will fill up and negatively impact performance.

Note: IdentityIQ supports ActiveMQ only. Whether you use embedded or external ActiveMQ, message destinations do **not** have expiry defined by IdentityIQ. Be sure to configure what will read from them before you start running data extracts. The internal processing queues used by the tasks do have expiration times set after the first run to ensure that failed execution do not block subsequent runs. If there is a time out event during a data extract task, the task will fail, allowing the cause to be fixed and the data to be re-extracted.

extractedObjects– a map of values. The key is the string names of the object types described in the transformation configuration, for example, identity, role, and certification. If that is all that is present, the data extract process will look up the classname from the transformation configuration object and find the classname to know which object to read from the database.

The value can contain an object with the following properties:

- **filterString** – a normal IdentityIQ filter string to be used for the queries
 - You can use a filter string to pull specific audit table rows based a particular action, so the usage would look like this:

```
className:
```

```
filterString: property == "<filter value>"
```

Example <filter value> would be name.startsWith("B")

- **deleteTransformFormat** – indicates this YAMLConfig is interested in intercepted deleted objects (optional)
 - You can set the output for the transform to brief, full, or none. Brief cannot be compared; full captures can be compared.
 - **Full** captures a full copy of what is intercepted including data such as ID, name, created date, modified date, and acknowledgement of child records.
 - **Brief** contains only the ID and name.
 - **None** gives no indication that a record was deleted.

Configuring Data Transformation

The Transform configuration tells the system how to format the extracted data. Objects and properties extracted from the IdentityIQ database may be transformed into something that can easily be exported in a JSON format to be more friendly to BI tools. The Transform YAML configuration defines how you want that done by listing the configuration for

each object and how to do the actual JSON transformations. View YAML configurations from the Debug > Object Browser page.

You may generate a Transform YAMLConfig in the IdentityIQ console. See [IIQ Console Commands](#). The following example creates the base default Transform config for the Identity object called IdentityTransformConfig:

```
>dataextract generatetransform --classes Identity --write IdentityTransformConfig
```

Customize your transform YAMLs according to your organization's needs. If you want to write your own transformers, use the existing IdentityIQ plugin interface and it can be called by Data Extract. See **Working with Plugins in IdentityIQ**.

You might use transformers for things like converting a timestamp to a specific format, or converting a string that refers to another object to an extract reference to that object, or converting the xml attributes to JSON containing a select set of attributes.

Each transformer is just one operation, but there are many ways to customize the JSON output by stacking transformers such that the output of one becomes the input of the next. Properties are transformed first – for example, transforming a UNIX time into a formatted date string – then object transformers are applied, such as adding a composite key or a hash value. Transformers are applied in the order listed.

A transform configuration may look like this example:

```
identity:
  includeHash: true
  objectClassName: sailpoint.object.Identity

imagePropertyConfigDescriptors:
-   property: id
-   property: name
-   property: managerStatus
-   attribute: email
```

Adding a hash by setting includeHash to True lets you quickly know if extracted objects are the same or different. Properties and attributes are those that you would like to export; this example includes id, name, manager status, and email. You may also export extended attributes, which are those that are defined only for your implementation.

Property transformers apply to a single property. For instance, you can use a property transformer to change the date format. Just add dateFormat (see Java's SimpleDateFormat) and optionally timeZone as shown in the example below:

```
1 imageConfigDescriptors:
2   identity:
3     objectClassName: sailpoint.object.Identity
4     imagePropertyConfigDescriptors:
5       - property: id
6       - property: name
7       - property: created
```



```
8     type: date
9     dateFormat: EEEE MMMM d yyyy ('julian day' - DD) - h:m:s a
10    timeZone: CST

1 dataextract transform identity james.smith deExamples
```

The result is a new format for the date – note that this example adjusts to CST instead of UTC (5 hours earlier):

```
1{
2  "created": "Wednesday September 7 2022 (julian day - 250) - 9:29:20 AM",
3  "name": "James.Smith",
4  "id": "ac130b1283181a728183185b0138065d"
5}
```

At the end of a list of specified transformers, you may use special YAML properties to make it easier to add 'built-in' transformers. This can be helpful if you want to add wrappers to convert values to ImageValues or want to use hashExclude to exclude values from a hash.

For an example of a complete Transform YAML Configuration, see [Sample Transform YAML Config](#).

Data Transformer Cookbook

This section contains some "recipes" for how to export object properties.

Here is a template for the configuration object (be sure to change the name):

```
1<?xml version='1.0' encoding='UTF-8'?>
2<!DOCTYPE YAMLConfig PUBLIC "sailpoint.dtd" "sailpoint.dtd">
3<YAMLConfig name="deExamples">
4  <Attributes>
5    <Map>
6      <entry key="ExportConfiguration">
7        <value>
8          <Script language="yaml">
9            <Source>
15             </Source>
16           </Script>
17         </value>
18       </entry>
19     </Map>
20   </Attributes>
21</YAMLConfig>
```

Using the console commands we can see how these objects will be transformed. The console command for this is:

```
dataextract transform <type> <name or id> <configuration object name>
```

Properties

In the configuration above, replace the part that says <CONFIGURATION GOES HERE> with this:

```
1imageConfigDescriptors:
2  identity:
3    objectClassName: sailpoint.object.Identity
4    imagePropertyConfigDescriptors:
5      - property: id
6      - property: name
```

The part that says "identity" is the friendly name of the object type, and the objectClassName is the actual Java class name of the object. Import the xml file, and use this configuration to export an identity:

```
1dataextract transform identity james.smith deExamples
```

Result (only the imageFields part)

```
1"imageFields": {
2  "name": "James.Smith",
3  "id": "ac10f8718187187d818187b962930671"
4}
```

Two properties were described, and two properties appear in the output.

Properties can be renamed easily using newName. Here is an example of renaming id to identityIQ_id:

```
1imageConfigDescriptors:
2  identity:
3    objectClassName: sailpoint.object.Identity
4    imagePropertyConfigDescriptors:
5      - property: id
6      - newName: identityIQ_id
7      - property: name
```

Result (rename)

```
1{
2  "identityIQ_id": "ac130b1283181a728183185b0138065d",
3  "name": "James.Smith"
4}
```

Date

Adding date created.

Example

```
1imageConfigDescriptors:
2  identity:
3    objectClassName: sailpoint.object.Identity
4    imagePropertyConfigDescriptors:
5      - property: id
6      - property: name
7      - property: created
1dataextract transform identity james.smith deExamples
```

Result

```
1{
2  "created": 1662560960824,
3  "name": "James.Smith",
4  "id": "ac130b1283181a728183185b0138065d"
5}
```

Notice the created field. This is a Java date object. It gets converted to a Unix date (long).

Date with default formatting

By just specifying that the type is date, default formatting is applied to the date (ISO 8601).

Example

```
1imageConfigDescriptors:
2  identity:
3    objectClassName: sailpoint.object.Identity
4    imagePropertyConfigDescriptors:
5      - property: id
6      - property: name
7      - property: created
8      type: date
1{
2  "created": "2022-09-07T14:29Z",
3  "name": "James.Smith",
4  "id": "ac130b1283181a728183185b0138065d"
5}
```

Date with formatting

You can change the date format. Just add dateFormat (see Java's SimpleDateFormat) and optionally timeZone.

Example

```
1imageConfigDescriptors:
2  identity:
3    objectClassName: sailpoint.object.Identity
4    imagePropertyConfigDescriptors:
5      - property: id
6      - property: name
7      - property: created
8      type: date
9      dateFormat: EEEE MMMM d yyyy ('julian day' - DD) - h:m:s a
10     timeZone: CST
1dataextract transform identity james.smith deExamples
```

Result

New format for the date, and note that it's in CST instead of UTC (5 hours earlier).

```
1{
2  "created": "Wednesday September 7 2022 (julian day - 250) - 9:29:20 AM",
3  "name": "James.Smith",
4  "id": "ac130b1283181a728183185b0138065d"
5}
```

ImageRef

If the property is a SailPoint object, it will become an imageref object.

Example

```
1imageConfigDescriptors:
2  identity:
3    objectClassName: sailpoint.object.Identity
4    imagePropertyConfigDescriptors:
5      - property: id
6      - property: name
7      - property: capabilities
```

Result

Notice that the object under capabilities has an id, name, and objectType. It's now an imageref.

```
1dataexport export identity wanda.watkins deExamples
1{
2  "capabilities": [
3    {
4      "id": "ac130b1283181a728183185a87b100bf",
5      "name": "SystemAdministrator",
```

```
6   "objectType": "sailpoint.object.Capability"  
7 }  
8 ],  
9 "name": "James.Smith",  
10 "id": "ac130b1283181a728183185b0138065d"  
11}
```

xmlString

Example

```
1imageConfigDescriptors:  
2  identity:  
3    objectClassName: sailpoint.object.Identity  
4    imagePropertyConfigDescriptors:  
5      - property: id  
6      - property: name  
7      type: xmlString
```

Result

```
1{  
2  "id": "ac130b1283181a728183185b0138065d",  
3  "name": "<name>James.Smith</name>"  
4}
```

Attributes

Example

Extract 'location' from the attributes map.

```
1imageConfigDescriptors:  
2  identity:  
3    objectClassName: sailpoint.object.Identity  
4    imagePropertyConfigDescriptors:  
5      - property: id  
6      - property: name  
7      - attribute: location
```

Result

```
1{
```

```
2 "name": "James.Smith",
3 "attributes": {
4   "location": "Austin"
5 },
6 "id": "ac130b1283181a728183185b0138065d"
7}
```

Normally attribute will put all attributes under the attributes container. If you want the name to be different, use the `attributesContainer` property on the image property config descriptor.

Example

```
1imageConfigDescriptors:
2  identity:
3    objectClassName: sailpoint.object.Identity
4    attributesContainer: myAttributes
5    imagePropertyConfigDescriptors:
6      - property: id
7      - property: name
8      - attribute: location
```

Result

```
1{
2  "name": "James.Smith",
3  "myAttributes": {
4    "location": "Austin"
5  },
6  "id": "ac130b1283181a728183185b0138065d"
7}
```

Notice the `myAttribute` property instead of `attributes`. There is an additional field for when a record's attributes column is not named "attributes." These differing column names are supported using `attributeSourceName`.

`extendedAttributes` and `extendedIdentityAttributes` work in the same manner.

Note: If you set the attributes container to nothing, the attribute will be a top-level property of the json. For example:

```
1imageConfigDescriptors:
2  identity:
3    objectClassName: sailpoint.object.Identity
4    attributesContainer:
5    imagePropertyConfigDescriptors:
```

```
6 - property: id
7 - property: name
8 - attribute: location
```

Result

```
1{
2  "name": "James.Smith",
3  "location": "Austin",
4  "id": "ac130b1283181a728183185b0138065d"
5}
```

Hashing

To add a hash value to an object use includeHash like this:

```
1imageConfigDescriptors:
2  identity:
3    includeHash: true
4    objectClassName: sailpoint.object.Identity
5    imagePropertyConfigDescriptors:
6      - property: id
7      - property: name
8
```

Result

```
1{
2  "sha1": "D6306DF4EE9DC1A261D3E1BC727998DA8A3797B3",
3  "name": "James.Smith",
4  "id": "ac130b1283181a728183185b0138065d"
5}
```

The property called "sha1" contains the hash. This can be change to another field by adding the hashProperty attribute to the ImageConfigDescriptor for identity.

Example

```
1imageConfigDescriptors:
2  identity:
3    includeHash: true
4    hashProperty: my_sha1_hash
```

```
5  objectClassName: sailpoint.object.Identity
6  imagePropertyConfigDescriptors:
7    - property: id
8    - property: name
```

Result

```
1{
2  "name": "James.Smith",
3  "id": "ac130b1283181a728183185b0138065d",
4  "my_sha1_hash": "D6306DF4EE9DC1A261D3E1BC727998DA8A3797B3"
5}
```

Notice the new name is my_sha1_hash.

Suppose that you want to exclude a property from the hash calculation – for example, created and modified. Just add hashExclude to the property config descriptor and set it to true.

Example

```
1imageConfigDescriptors:
2  identity:
3    includeHash: true
4    hashProperty: my_sha1_hash
5    objectClassName: sailpoint.object.Identity
6    imagePropertyConfigDescriptors:
7      - property: id
8      - property: name
9      - property: created
10     hashExclude: true
11     - property: modified
12     hashExclude: true
```

Result

```
1{
2  "created": 1662560960824,
3  "name": "James.Smith",
4  "modified": 1662562037284,
5  "id": "ac130b1283181a728183185b0138065d",
6  "sha1": "D6306DF4EE9DC1A261D3E1BC727998DA8A3797B3"
7}
```

Notice the created and modified properties are in the json, but the hash is identical.

Decrypt

You can decrypt values. See [Using Data Extract with Sensitive Data](#).

Caution: It is your responsibility to ensure that only data that do not violate security policies within your environment are included as extractable or decryptable.

Example

```
1imageConfigDescriptors:
2  identity:
3    objectClassName: sailpoint.object.Identity
4    imagePropertyConfigDescriptors:
5      - property: id
6        newName: identityIQ_id
7      - property: name
8      - property: password
9      decrypt: true
```

Result

```
1{
2  "password": "xyzyzy",
3  "identityIQ_id": "ac130b1283181a728183185b0138065d",
4  "name": "James.Smith"
5}
```

Objects

With type object, you can specify an object name to expand an object in place. Normally it would be an identity ref, but suppose you want to expand it in place.

Note: In this example, we need to also specify transformType of elements because this is a list. We don't want to transform the list, we want to transform each item in the list.

Example

```
1imageConfigDescriptors:
2  identity:
3    objectClassName: sailpoint.object.Identity
4    imagePropertyConfigDescriptors:
5      - property: id
6      - property: name
7      - property: links
```

```

8     type: object
9     objectName: link
10    transformType: elements
11
12    link:
13      objectClassName: sailpoint.object.Link
14      imagePropertyConfigDescriptors:
15        - property: id
16        - property: applicationName
17        - property: nativeIdentity

```

Result

```

1{
2  "name": "James.Smith",
3  "links": [
4    {
5      "id": "ac130b1283181a728183185b1b310862",
6      "applicationName": "HR_Employees",
7      "nativeIdentity": "1a"
8    },
9    {
10     "id": "ac130b1283181a728183185b53c811c8",
11     "applicationName": "Active_Directory",
12     "nativeIdentity": "100"
13   },
14   {
15     "id": "ac130b1283181a728183185be2e417e4",
16     "applicationName": "ADDirectDemodata",
17     "nativeIdentity": "CN=James Smith,OU-
U=Austin,OU=Americas,OU=DemoData,DC=test,DC=sailpoint,DC=com"
18   },
19   {
20     "id": "ac130b1283181a728183185c81011917",
21     "applicationName": "RealLDAPWithDemoData",
22     "nativeIdentity": "CN=James Smith,OU-
U=Austin,OU=Americas,OU=DemoData,DC=test,DC=sailpoint,DC=com"
23   }
24 ],
25 "id": "ac130b1283181a728183185b0138065d"
26}

```

TransformType

This is documented in the Objects section above.

Exclude Boolean false

For the sake of brevity in the json, you may want to exclude Boolean values that are false.

Example

```
1excludeBooleanFalse: true
2imageConfigDescriptors:
3  identity:
4    objectClassName: sailpoint.object.Identity
5    imagePropertyConfigDescriptors:
6      - property: id
7      - property: name
8      - property: inactive
```

Result

```
1{
2  "name": "James.Smith",
3  "id": "ac130b1283181a728183185b0138065d"
4}
```

This is what the results would have looked like without excludeBooleanFalse

```
1{
2  "inactive": false,
3  "name": "James.Smith",
4  "id": "ac130b1283181a728183185b0138065d"
5}
```

Classifications

This is really the same thing as object – it's just another object. But since it comes up in a lot of IdentityIQ objects, here is an example of how to extract classifications.

This configuration will extract the property called classifications. It's a list of ObjectClassification objects. Each object classification object has a property called classification. It's a Classification object. This example uses names for the imageConfigDescriptors that match the class names. While that is not strictly required, is a best practice.

Example

```
1imageConfigDescriptors:
2  Bundle:
3    imagePropertyConfigDescriptors:
4      - property: "id"
5      - property: "name"
6      - property: "displayName"
7      - property: "description"
```

```

8   - property: "disabled"
9   - property: "modified"
10  - property: "created"
11  - property: "owner"
12  - property: "classifications"
13    type: object
14    objectName: ObjectClassification
15    transformType: elements
16    objectClassName: "sailpoint.object.Bundle"
17
18  ObjectClassification:
19    imagePropertyConfigDescriptors:
20      - property: id
21      - property: classification
22      type: object
23      objectName: Classification
24
25  Classification:
26    imagePropertyConfigDescriptors:
27      - property: name
28      - property: id
29      - property: origin
30      - property: displayName
31      hashExclude: true
1> dataextract transform Bundle "sailpoint dev" bconfig | jq

```

Result

```

1{
2  "owner": {
3    "id": "ac130b1283aa11858183aa01a19100ea",
4    "name": "spadmin",
5    "objectType": "sailpoint.object.Identity"
6  },
7  "classifications": [
8    {
9      "id": "ac130b1283cd1d008183cdfce6700105",
10     "classification": {
11       "origin": "JDBCDirectDemoData",
12       "name": "Special7",
13       "id": "ac130b1283aa11858183aa02fefb1b58"
14     }
15   }
16 ],
17 "created": 1665004612038,
18 "name": "sailpoint dev",
19 "modified": 1665608246939,
20 "disabled": false,
21 "id": "ac130b1283aa11858183aa0229c60805"
22}

```

Importing Data Extract and Transform Configurations

Import your Extract and Transform YAMLConfigs using the IdentityIQ user interface (see **Import from File** in the *System Configuration Guide*) or the console.

To import through the IdentityIQ console:

1. Open a command prompt and launch the IdentityIQ console. See **Launching the Console** in the *IdentityIQ Console Guide*.
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.

Provisioning a Data Extract Task Destination Queue

The Data Extract task destination queue value can be a string that represents the ActiveMQ messageDestinations name, or it can be a beanshell script like this example:

```
messageDestination: |
script:
import sailpoint.object.Configuration;

Configuration config = Configuration.getAccessHistoryConfig();
return config.get("destinationQueue");
```

See [Configuring Data Extraction](#).

Enabling and Configuring Data Extract

Prior to running the Data Extract task, you need to enable that task.

1. Navigate to **gear > Global Settings > Data Extract Configuration**.
2. Select the **Enable Data Extract Tasks** checkbox.
3. Set the internal processing queue.
4. Set the internal processing message expiration time, which is the length of time in minutes that a message should be retained in the internal processing queue.
5. Select **Save Changes**.

Setting Up Data Extract Task

The Data Extract Task selects objects from the IIQ database, processes each object, and writes a message for each to a destination queue. The first time Data Extract runs, it completes a full extract for the defined objects. Subsequent task runs extract a delta based on what has changed since the last time it ran.

You can set up this task to run on your instance:

1. Configure a Data Extraction YAMLConfig for the task provides what types of objects to extract. See [Configuring Data Extraction](#).
2. Set a message destination to dispatch messages to. See [Provisioning a Data Extract Task Destination Queue](#).
3. Configure a transformConfigurationName YAMLConfig to describe how to extract the types from the first YAMLConfig extractedObjects. See [Configuring Data Transformation](#).
 - a. Make sure the extractedObjects from the first YAMLConfig all have a corresponding imageConfigDescriptor and that each has a valid objectClassName
4. Navigate to **Setup > Tasks**.
5. Select the **New Task** dropdown in the upper right corner.
6. From the dropdown list, select **Data Extract**.

Note: When upgrading to version 8.4 from another version of IdentityIQ, if you do not see the Data Extract option, then make sure you followed the upgrade process by importing upgradeObjects. If it's a clean installation, then you need to reimport init.xml.

7. On the New Task screen, enter a Name for your task and add any other optional field information you would like.
8. Under Data Extract Options, select a Data Extract YAMLConfig.
9. Select **Save**, **Save & Execute**, **Cancel**, or **Refresh**.
 - a. Executing the task looks at what objects are configured to be exported, applies the filter criteria and any limits that you have set and translates all of those objects into JSON documents, and writes them to a JMS queue.
 - b. If executed, review the Task Results, which display all the differences as well as the attribute statistics. See [Viewing Data Extract Task Results](#).

You can schedule this task to run on a regular cadence. See [How to Schedule a Task](#) in the *Tasks Guide*.

If you configure different YAML configurations for different object types, you can also configure separate tasks to run at different intervals. For example, YAML 1 may be configured for Object X and YAML 2 for Object Y. Task 1 for YAML 1 may be scheduled to run every week, while Task 2 for YAML 2 may be scheduled to run every day.

Working with the Data Extract Message Broker

[Viewing Overall Broker Status on the ActiveMQ Monitoring Page](#)

[Starting and Stopping Connectivity to an External Broker](#)

[Configuring Message Broker](#)

[Starting and Stopping an Embedded Message Broker](#)

Viewing Overall Broker Status on the ActiveMQ Monitoring Page

Data Extract may use an embedded ActiveMQ message broker, but organizations are encouraged to set up external ActiveMQ instances. When using an external broker, use that broker's monitoring tool for greater administrative capabilities.

Use the ActiveMQ monitoring page to view the high-level status of brokers that are registered in the configuration, and view messages processing during the data extract task.

1. Navigate to **Debug menu > ActiveMQ Monitoring**.
2. The following information is included on the ActiveMQ Monitoring page:
 - a. Broker information – ID, name, VM URL, status, and data directory
 - b. Connectors – connector name and binding address
 - c. Broker statistics – memory usage, storage usage, total message count, average message size, counts for total enqueue and dequeue, total counts for producer and consumer
 - d. Destinations – table lists destinations by name and includes size, enqueue messages, dequeue messages, and average enqueue time.

Starting and Stopping Connectivity to an External Broker

Connectivity information, along with set up and administration of the different queues you want to use for statistics, is found in Broker Configuration. You may want to stop the broker to complete maintenance on an external broker implementation or in the case of a disaster recovery failover.

To start or stop the broker:

1. Navigate to **gear > Global Settings > Messaging Configuration**.
2. Select or deselect the checkbox for Enable Messaging.
3. Select **Save Changes**.

If the message broker is disabled and you navigate to something that depends on it, such as Access History or Data Extract, a banner warning states, "Messaging is disabled. You can enable messaging in Global Settings > Messaging Configuration."

Configuring Message Broker

To configure an embedded or external message broker for Data Extract, navigate to **gear > Global Settings > Messaging Configuration** and select the checkbox to **Enable Messaging**.

There are sections for:

- Broker Type
 - Embedded – the embedded ActiveMQ broker that is included with IdentityIQ.
 - External – an external ActiveMQ broker that you maintain.
- Embedded Broker Settings
 - Lock Keep Alive Period (required) – the amount of time in milliseconds that the lock should be kept alive.
 - Lock Acquire Sleep Interval (required) – duration in milliseconds for how long the broker will wait before trying to retry acquiring a lock. For Lease Database Locker, this value should be larger than the Lock Keep Alive Period.
 - Max Page Size (required) – the maximum number of messages to page in from the store at one time.
 - DLQ Message Expiration in Days (required) – duration in days that messages in the dead letter queue (DLQ) will be expired and removed.
- Connection Settings
 - Client Connection String (required) – a string that defines how clients connect to the broker, in the format of 'failover:(tcp://<host>:<port>)?<options>'
 - Enable Failover Optimization – if enabled, the system will use the last known broker url first when connecting to the broker.

- Consumer Username (required) – username for read-only connections. For the embedded broker, it may take a few minutes for the change to take effect.
- Consumer Password (required) – password for read-only connections. For the embedded broker, it may take a few minutes for the change to take effect.
- Producer Username (required) – username of the connection that will write messages onto the queues. For the embedded broker, it may take a few minutes for the change to take effect.
- Producer Password (required) – password of the connection that will write messages onto the queues. For the embedded broker, it may take a few minutes for the change to take effect.
- Admin Username (optional for external brokers, required for embedded brokers) – username of the connection that creates queues if a queue is not available. This setting is required for Embedded Brokers. For the Embedded Broker, it may take a few minutes for the change to take effect.
- Admin Password (optional for external brokers, required for embedded brokers) – password of the connection that creates queues if a queue is not available. This setting is required for Embedded Brokers. For the Embedded Broker, it may take a few minutes for the change to take effect.
- Statistics Settings
 - Broker Statistics Queue Name (optional for external brokers unless you are using the Debug > ActiveMQ Monitoring page, required for embedded) – name of the ActiveMQ queue on which overall statistics are stored.
 - Destinations Statistics Queue Name (optional for external brokers unless you are using the Debug > ActiveMQ Monitoring page, required for embedded) – name of the ActiveMQ queue on which destination statistics are stored.
 - Subscriptions Statistics Queue Name (optional for external brokers unless you are using the Debug > ActiveMQ Monitoring page, required for embedded) – name of the ActiveMQ queue on which subscription statistics are stored.

Note: Selecting an external broker disables the embedded broker settings.

Starting and Stopping an Embedded Message Broker

The ActiveMQ embedded message broker is run as an IIQ service that can be enabled or disabled for each host in the Administrator Console UI. The shutdown message is broadcast to embedded brokers, but they may take some time to shut down. Attempts to turn them back on will not be accepted until the shutdown is completed.

To start or stop the broker:

1. Navigate to **gear > Administrator Console > Environment**.
2. Select the **gear** button in the Host Actions column for the hostname. This launches a pop-up dialog.
3. EmbeddedBroker is listed on the Services tab and can be toggled on or off.
4. Select **Save**.

If the message broker is disabled and you navigate to something that depends on it, such as Access History or Data Extract, a banner warning states, "Messaging is disabled. You can enable messaging in Global Settings > Messaging Configuration."

Running a Data Extract Task

Use the IdentityIQ console to run a data extraction task, run on demand from the Tasks page, or schedule Data Extract to run at a time or cadence you choose. See **Tasks Overview** and **How to Schedule a Task** in the *Tasks Guide*.

Viewing Data Extract Task Results

View Data Extract task results at **Setup > Tasks > Task Results**. The top section displays the total number of Extracted Objects and Extracted Objects Dispatched. The values should be the same. Also included are the total number of Deleted Objects, Deleted Objects Dispatched, and a breakdown of the types of deleted objects dispatched and the total for each. The total for all objects listed in this section equals the number of Deleted Objects Dispatched in the top section.

Note: If an object is deleted, you will receive an update on the message queue for the deleted object so that you can update your data repository. Under Extracted Objects Dispatched Types is a breakdown of types of objects dispatched and the total for each. The total for all objects listed in this section equals the number of Extracted Objects Dispatched in the top section.

Details can be expanded or collapsed. The expanded view shows a list of partitions with columns for Name, Host, and Status. Each partition can be expanded or collapsed. When expanded, each shows number of Extracted Objects and Extracted Objects dispatched, including any deletion objects processed.

The Deleted Objects Processed are handled in a unique way – they are processed in parallel to Data Extract and are deleted after sending. If a Data Extract task fails for any reason, there is a chance that the deleted objects may still be processed and a notification of the event may be on the destination queue. As the task is asynchronous, the failure could result in data being in several locations throughout the workflow, including the destination queue. The message consumer will handle any duplicates.

Validating Data Extract Results

If your implementation uses the embedded ActiveMQ message broker, you can verify extracted messages in the `ACTIVEMQ_MSGS` table. There is not a way to monitor temporary queues.

If your implementation uses external ActiveMQ, you can see the following happen as a Data Extract task runs:

- `OBJECT_SELECTOR` messages are written in the inbound queue.
 - This is the list of object IDs / names to extract.
 - The partitions / threads will read messages from this queue then process and write to the destination queue.
- `EXTRACTED_OBJECT` messages are written to the destination queue with the name specified in your Extract YAMLConfig.

To verify the content of `EXTRACTED_OBJECT` messages, you can do the following:

1. Run Task with Transform / Extract config to extract all objects of all types (generate transform config for all objects) and verify messages.
2. Run Task with Transform / Extract config to extract all objects for single object type.
3. Run Task with Transform / Extract config to extract all objects for multiple object types.
4. Run Task with Transform / Extract config with `filterString` to restrict which objects of a particular type to extract.

Verify that the number of messages match:

- **Using Embedded Message Broker:** In the task result, verify that the number of message in the table for destination queue (see Debug > ActiveMQ Monitoring, scroll down to Destinations) matches the number of Extracted Objects/Extracted Objects Dispatched (see Task Results).
- **Using External Broker:** Verify that the number of messages in the temporary queue match the number in the destination queue. Additionally, the task result should show the same number for Extracted Objects as for Extracted Objects Dispatched.

Spot check messages:

1. Generate the default Transform YAMLConfig for identity. Add `filterString` to Extract YAMLConfig to extract single identity. Run the task using that config and view the dispatched message content, verify data looks correct.

2. Remove some properties or attributes from the Transform YAMLConfig and repeat. View dispatched message content, verify that it's correct.
3. Repeat with some other tasks.

Intercept Delete Functionality

When an identity is removed from a workgroup, they effectively lose permissions or access to certain functionality that is associated with the workgroup. That change in permissions is now reflected in the identity cube and can be viewed by running the Data Extract task and reviewing the Task Results. This is driven by the Created By and Modified By attributes.

Data Extract Auditing

Enable Data Extract auditing by navigating to **gear > Global Settings > Audit Configuration**. When you Run Task in the audit configuration, the task shows up as an audit event viewable at **Intelligence > Advanced Analytics > Search Type: Audit, Action: Run Task**.

Sample Transform YAML Config

```

1 <YAMLConfig name="ExampleTransformConfig" type="Transform">
2   <YamlText>
3     <![CDATA[
4       excludeBooleanFalse: true
5       imageConfigDescriptors:
6
7         identity:
8           includeHash: true
9           hashProperty: smartHash
10          objectClassName: sailpoint.object.Identity
11          objectTransformerConfigDescriptors:
12            - type: sailpoint.dataextract.conversion.ExtendedIdentityAttributes
13              args:
14                excludedKeys: ""
15                saveAs: "extendedAttributesIdentity"
16          imagePropertyConfigDescriptors:
17            - property: id
18            - property: name
19            - property: displayName
20            - property: displayableName
21            hashExclude: true
22            - property: managerStatus
23              newName: isManager
24              type: bool
25            - property: administrator
26              type: object
27            objectName: identityRef

```

```
28         - property: manager
29           type: object
30           objectName: identityRef
31         - property: inactive
32         - property: bundles
33           newName: detectedRoles
34           type: object
35           objectName: bundleRef
36           transformType: elements
37         - property: assignedRoles
38           type: object
39           objectName: bundleRef
40           transformType: elements
41         - property: roleAssignments
42           type: object
43           objectName: roleAssignment
44           transformType: elements
45         - property: id
46           newName: entitlements
47           type: none
48           propertyTransformerConfigDescriptors:
49             - type: sailpoint.dataextract.conversion.IdentityIdEntitlements
50               args:
51                 objectName: identityEntitlement
52         - property: attributes
53           type: none
54           propertyTransformerConfigDescriptors:
55             - type: sailpoint.dataextract.conversion.AttributesWithExclusions
56               args:
57                 excludedKeys: displayName,inactive
58                 objectConfigTypes: system,standard,unclear
59         - property: extendedAttributes
60           type: none
61           propertyTransformerConfigDescriptors:
62             - type: sailpoint.dataextract.conversion.AttributesWithExclusions
63               args:
64                 objectConfigTypes: extended
65         - property: attributes
66           newName: extendedAttributesNamed
67           type: none
68           propertyTransformerConfigDescriptors:
69             - type: sailpoint.dataextract.conversion.AttributesWithExclusions
70               args:
71                 objectConfigTypes: named
72         - property: links
73           newName: accounts
74           type: object
75           objectName: link
76           transformType: elements
77
78     role:
79       includeHash: true
80       hashProperty: smartHash
81       objectClassName: sailpoint.object.Bundle
```

```
82     objectTransformerConfigDescriptors:
83       - type: sailpoint.dataextract.conversion.ExtendedIdentityAttributes
84       args:
85         excludedKeys: ""
86         saveAs: "extendedAttributesIdentity"
87     imagePropertyConfigDescriptors:
88       - property: id
89       - property: name
90       - property: displayName
91       - property: inheritance
92       newName: supers
93       type: object
94       objectName: bundleRef
95       transformType: elements
96       - property: permits
97       newName: permitteds
98       type: object
99       objectName: bundleRef
100      transformType: elements
101      - property: requirements
102      newName: requireds
103      type: object
104      objectName: bundleRef
105      transformType: elements
106      - property: activationDate
107      - property: deactivationDate
108      - property: attributes
109      type: none
110      propertyTransformerConfigDescriptors:
111        - type: sailpoint.dataextract.conversion.AttributesWithExclusions
112        args:
113          excludedKeys: displayName
114          objectConfigTypes: system,standard,unclear
115        - property: extendedAttributes
116        type: none
117        propertyTransformerConfigDescriptors:
118          - type: sailpoint.dataextract.conversion.AttributesWithExclusions
119          args:
120            objectConfigTypes: extended
121        - property: attributes
122        newName: extendedAttributesNamed
123        type: none
124        propertyTransformerConfigDescriptors:
125          - type: sailpoint.dataextract.conversion.AttributesWithExclusions
126          args:
127            objectConfigTypes: named
128
129    link:
130      attributesContainer:
131      objectClassName: sailpoint.object.Link
132      objectTransformerConfigDescriptors:
133        - type: sailpoint.dataextract.conversion.UniqueKeyGenTransformer
134        args:
135          fields:
```

```
136         - applicationId
137         - nativeIdentity
138         saveAs: "__key"
139         separator: "|"
140     imagePropertyConfigDescriptors:
141         - property: id
142         - property: applicationId
143         - property: applicationName
144         - property: nativeIdentity
145         - property: iiqLocked
146         - property: iiqDisabled
147         - property: attributes
148         type: none
149     propertyTransformerConfigDescriptors:
150         - type: sailpoint.dataextract.conversion.AttributesWithExclusions
151         args:
152             excludedKeys: displayName,IIQLocked,IIQDisabled
153             objectConfigTypes: system,standard,unclear
154         - property: extendedAttributes
155         type: none
156     propertyTransformerConfigDescriptors:
157         - type: sailpoint.dataextract.conversion.AttributesWithExclusions
158         args:
159             objectConfigTypes: extended
160         - property: attributes
161         newName: extendedAttributesNamed
162         type: none
163     propertyTransformerConfigDescriptors:
164         - type: sailpoint.dataextract.conversion.AttributesWithExclusions
165         args:
166             objectConfigTypes: named
167     certificationEntity:
168         objectClassName: sailpoint.object.CertificationEntity
169     imagePropertyConfigDescriptors:
170         - property: targetId
171         - property: targetName
172         - property: type
173         - property: completed
174         type: unixtime
175     identityEntitlement:
176         objectClassName: sailpoint.object.IdentityEntitlement
177     imagePropertyConfigDescriptors:
178         - property: appName
179         - property: name
180         - property: value
181         - property: type
182         - property: grantedByRole
183         type: bool
184     certification:
185         objectClassName: sailpoint.object.Certification
186     imagePropertyConfigDescriptors:
187         - property: id
188         - property: name
189         - property: type
```

```

190         - property: phase
191         - property: finished
192         type: unixtime
193         - property: signed
194         type: unixtime
195         - property: certifiers
196         - property: entities
197         type: object
198         objectName: certificationEntity
199         transformType: elements
200     identityRef:
201         imagePropertyConfigDescriptors:
202         - property: name
203         - property: id
204         - property: displayName
205         hashExclude: true
206
207     bundleRef:
208         imagePropertyConfigDescriptors:
209         - property: name
210         - property: id
211         - property: type
212         - property: displayName
213         hashExclude: true
214
215     roleAssignment:
216         imagePropertyConfigDescriptors:
217         - property: roleId
218         - property: roleName
219         - property: comments
220         - property: assigner
221         - property: date
222         - property: source
223         - property: negative
224         - property: startDate
225         - property: endDate
226         - property: assignmentId
227     ]}]>
228 </YamlText>
229 </YAMLConfig>

```

Sample Transform YAML Config with Explanation

Here is an example of an actual Transform YAML Config.

```

1 imageConfigDescriptors:
2   identity:
3     objectClassName: sailpoint.object.Identity
4     imagePropertyConfigDescriptors:
5       - property: id

```



```

6     type: none
7     propertyTransformerConfigDescriptors:
8     - type: sailpoint.dataextract.conversion.ImagePropertyTransformer
9       args:
10        propertyType: string
11     - property: name
12       type: none
13     propertyTransformerConfigDescriptors:
14     - type: sailpoint.dataextract.conversion.ImagePropertyTransformer
15       args:
16        propertyType: string
17     - property: firstname
18       type: none
19     propertyTransformerConfigDescriptors:
20     - type: sailpoint.dataextract.conversion.ImagePropertyTransformer
21       args:
22        propertyType: string
23     - property: lastname
24       type: none
25     propertyTransformerConfigDescriptors:
26     - type: sailpoint.dataextract.conversion.ImagePropertyTransformer
27       args:
28        propertyType: string
29     - property: modified
30       type: none
31     propertyTransformerConfigDescriptors:
32     - type: sailpoint.dataextract.conversion.ImagePropertyTransformer
33       args:
34        propertyType: date

```

You can view an example of how this will get exported by entering the following in the IIQ console:

```

1dataextract transform identity james.smith deExampleOne
2- or - if you have jq installed and want the pretty version
3dataextract transform identity james.smith deExampleOne | jq

```

Here is an excerpt of the output run through jq for formatting:

```

1{
2  "firstname": "James",
3  "name": "James.Smith",
4  "modified": "2022-09-07T14:47Z",
5  "id": "ac130b1283181a728183185b0138065d",
6  "lastname": "Smith"
7}

```

There are some shortcuts so you don't have to specify every piece of information for each property. While the Type property in the example above is set to "none," its default value is "auto." This property will append the

ImagePropertyTransformer with an argument of "auto," or whatever the Type is set to, so our configuration could be simplified to this:

```
1 imageConfigDescriptors:
2   identity:
3     objectClassName: sailpoint.object.Identity
4     imagePropertyConfigDescriptors:
5       - property: id
6       - property: name
7       - property: firstname
8       - property: lastname
9       - property: modified
```

With no Type specified, the automatically appended ImagePropertyTransformer tries to infer the Type and use that.

One important reminder is that the Type of every property is defaulted to Auto, which causes the ImagePropertyTransformer to be appended. If Data Extract can't determine what type to use, it will just use the object's toString method.

Allowed Properties

The subsections below detail properties allowed in Data Extract Transform configurations.

TransformationConfigDescriptor

Map<String, ImageConfigDescriptor> imageConfigDescriptors – map of ImageConfigDescriptor objects that is keyed by the object type's "friendly name," for example identity.

boolean excludeBooleanFalse – (default false) – if set to true, this will exclude any Boolean values that are equal to false during the transformation to JSON.

ImageConfigDescriptor

String objectClassName – the name of the image object's Java class. For example sailpoint.object.Identity.

List<ImagePropertyConfigDescriptors> imagePropertyConfigDescriptors – list of descriptors for the image property config.

boolean includeHash – (default false) – true if you would like to append a property to the JSON object that contains the sha1 hash of the JSON object.

String hashProperty – (default "sha1") – the name of the property that gets appended to the JSON object containing the sha1 hash of the JSON object. This will only be used if includeHash is set to true.

String attributesProperty – (default "attributes") – the name of the property in the source object that will serve as the attributes map. This will almost certainly always be "attributes."

String attributesContainer – (default "attributes") – the container to put the attribute values under in the JSON output.

String extendedAttributesContainer – (default "extendedAttributes") – the container to put the extendedAttribute values under in the JSON output.

String extendedIdentityAttributesContainer – (default "extendedIdentityAttributes") – the container to put the extendedIdentityAttribute values under in the JSON output.

List<ObjectTransformerConfigDescriptor> objectTransformerConfigDescriptors – list of transformers that can be invoked on the resulting ImageValueObject after all property transformations are complete.

ImagePropertyConfigDescriptor

String property – the name of the property to export from the IdentityIQ object.

String newName – the name given to the property in the resulting JSON document

String attribute – if attribute is used instead of property it assumes the property is attribute, and the value will be extracted from the attribute property of the source object.

String extendedAttribute – if extendedAttribute is used instead of property it assumes the property is an extendedAttribute, and the value will be extracted from the extendedAttribute property of the source object.

String attribute – if extendedIdentityAttribute is used instead of property it assumes the property is extendedIdentityAttribute, and the value will be extracted from the extendedIdentityAttribute property of the source object.

String timeZone – specifies a timeZone for the ImagePropertyTransformer that is appended for date formats. The default is UTC.

String dateFormat – specifies a SimpleDateFormat string for the ImagePropertyTransformer that is appended for date formats. The default is 'yyyy-MM-dd'T'HH:mm'Z'

boolean decrypt – appends the DecryptionTransformer. This will be appended before the ImagePropertyTransformer if the type parameter is specified.

boolean hashExclude – adds a parameter to the ImagePropertyTransformer (if type is not set to none) that will wrap the resulting object in a NonHashableImageValue. These are excluded from hash calculations for ImageValueObjects. If you do set type to none, you can still accomplish the same thing, but you will need to wrap the object in the NonHashableImageValue. One way to do this is by manually appending the ImagePropertyTransformer to the list of transformers, and using the argument to the constructor to tell it to do this wrapping.

String type – (default is auto. Valid values: none, auto, string, number, bool, list, date, unixtime, imageref, xmlString.) – appends the ImagePropertyTransformer and use the value as the first argument to its constructor. If it is set to None, it will not append any transformer.

String objectName – if type is set to object, objectName is used to transform the sub-object.

String transformType – (default is object. Valid values: elements, object.) – if the property value is a list, the transformType determines how it is passed into the transformers. If the transform type is elements, each element will be passed into the transformers, and then added to a list. If the type is object, the entire list will be passed into the transformer.

List<PropertyTransformerConfigDescriptors> propertyTransformerConfigDescriptors – a list of PropertyTransformerConfigDescriptors. It's important to note that if the type property of the PropertyConfigDescriptor is set to none, then only transformers in this list will be called in the order they are listed. Remember that type appends transformers as do a couple other things. It's fine to mix these, but just remember that more transformations can happen than what are in this list unless the type is set to none.

PropertyTransformerConfigDescriptor

String type – the java class name of the transformer. For example: sailpoint.dataexport.conversion.ImagePropertyTransformer

String pluginName – optional – name of a plugin to load the transformer from.

List<String> args – the arguments passed into the constructor of that transformer.

Transformers

DecryptionTransformer – decrypts encrypted values

ExportAttributeTransformer – extracts an attribute from the attributes map. While it may be easier to use attribute instead of property, this gives complete control.

ExportAttributeXPathTransformer – extracts an attribute from the attributes xml document using an XPath query. While it may be easier to use attribute instead of property, this gives complete control.

ExportHashTransformer – adds a sha1 hash to an object.

ImagePropertyTransformer – wraps a type in an ImageValue object. This will frequently be the last transformer in the chain, since all transformations must ultimately result in an ImageValue derived object.

ImageValueObjectTransformer – transforms image value objects by looking up their YAML configurations. This can be used to clean up the YAML file, or to embed objects into other objects. It can also be used for things like the link object contained in identity objects.

IdentityIQ Object Model and Usage

Data Extract allows you to configure which objects to extract. See *IdentityIQ Object Model and Usage* in Compass for details on core objects and key areas of application functionality.

Using Data Extract with Sensitive Data

By default, attributes that are known to be encrypted or sensitive cannot be extracted from standard SailPointObject fields. If your implementation is storing an Extended Attribute defined by your organization and if you define that attribute as part of a Data Extract YAMLConfig, then it will be extracted as normal.

If an encrypted attribute were included in a data extraction, it would be extracted in its encrypted state unless you take extra steps to decrypt it.

To allow your configuration to extract encrypted data, modify the system configuration.

1. In the Debug pages, open the **SystemConfiguration** object.
2. Locate the **encryptedClassFieldConfig** section.
3. Locate the object you would like to make extractable and/or decryptable. They may look like this:

```
<entry key="extractable" value="false"/>
```

```
<entry key="decryptable" value="false"/>
```

4. Update the extractable and/or decryptable values to true or false as needed.

Caution: It is your responsibility to ensure that only data that do not violate security policies within your environment are included as extractable or decryptable.

The following tables are not extractable in any circumstance:

- AuthenticationAnswer
- RemoteLoginToken
- SAMLToken

Caution: If customers define these objects within an attributes column, they may be extracted.

Rules and Scripts in IdentityIQ

Rules in IdentityIQ allow the addition of custom business logic at specific points within the execution flow of the product. To enable this, IdentityIQ allows system integrators to write rules in the BeanShell scripting language. The BeanShell language is based on Java and can use all Java classes that are available to IdentityIQ, including custom code.

- [Types of Rules and Scripts](#)
- [How Rules Are Created](#)
- [Type Attributes in Rules](#)
- [Using the Rule Editor](#)
- [Rule Re-Use](#)
- [Restricting Access to the Rule Editor](#)
- [Importing Rule XML](#)
- [Sample Rule Files](#)
- [Rule Libraries](#)
- [API Reference for Rules](#)
- [Testing Rules](#)

Types of Rules and Scripts

Rules and scripts are used in many places within IdentityIQ, including but not limited to:

- Modifying IdentityIQ behavior
- Validating and customizing aggregated data
- Filtering and modifying data during provisioning
- Advanced correlation of application accounts to Identity Cubes

In addition to these standard purposes, rules can implement automated tasks and perform advanced actions not contained in the standard features of IdentityIQ.

Scripts are similar to rules; the main difference between them is where they are stored. Scripts are embedded in the objects that use them, such as policy objects and workflows, whereas rules are separate objects that are referred to by the objects that use them.

How Rules Are Created

Rules are created within IdentityIQ in one of two ways: through the UI-based Rule Editor, and by importing rule XML objects.

See [Using the Rule Editor](#) and [Importing Rule XML](#) for details.

Type Attributes in Rules

Specialized rules must have a type attribute so that they can be selected from dropdown menus throughout the application. Each specialized type also has its own set of input and output variables and, optionally, a return value.

In each area where you can select a rule, you are limited to choosing (or creating) rules of the type that is appropriate to the feature. For example, in the Edit Policy page, the Policy Owner rule dropdown only lets you choose rules of the type "PolicyOwner."

The rule type is shown in both the Rule Editor, and in the rule's signature in the Debug pages.

Using the Rule Editor

Rules are associated with system activities on a variety of pages (such as the Schedule Certification, Edit Application, and Edit Policy pages), and throughout the IdentityIQ user interface. At these points, an existing rule can be selected to attach to the activity. Most, but not all, of these UIs also provide an entry point to the Rule Editor, for writing new rules or editing existing rules. Some UIs, such as the Rapid Setup Configuration pages, allow you to select rules but do not provide a direct link to the Rule Editor.

To edit an existing rule, first select the rule, then click the [...] button to open the Rule Editor.

To create a new rule, click the [...] button without selecting an existing rule, and a blank Rule Editor window is opened where you can create a new rule.

Rule Editor Fields

The Rule Editor uses these fields to define the rule and its business logic:

Copy From Existing Rule

If you want to reuse existing code in your new rule, select an existing rule to copy.

Rule Name

Enter a unique name for the new rule, that clearly describes the purpose. It is useful to include the type of rule in the name, such as "Customization – My App," or "Exclusion – Inactive Users."

Description

Enter text that describes the purpose of the rule and what it does.

Arguments and Returns

These sections list the input variables (Arguments) that can be used in the script, and the return information (Returns). Click on any of the arguments or returns to show more detailed information. These fields cannot be changed in the Rule Editor.

Rule Type

The rule type determines the purpose of the rule and where it can appear in dropdown lists in the IdentityIQ UI. In most cases, the Rule Editor does not allow you to change the Rule Type.

Rule Editor

In the main editor of this UI, enter the BeanShell code for the rule. The rule cannot be saved until some code is entered. Note that no validation is performed; even if the rule is syntactically incorrect, it is accepted and saved. Incorrect rules may result in tasks, workflows or certifications failing to execute, and in errors and large stack traces in the log files.

Rule Re-Use

A single rule can be reused in many places throughout IdentityIQ; for example, two applications could share the same Build Map Rule. Changes made to the rule in one location will affect the functionality in all locations where it is used, so if the functionality varies between usages, separate rules should be created for each functional need.

Restricting Access to the Rule Editor

The ability to edit inline scripts in Identity IQ requires either the **SystemAdministrator** capability or the **EditScripts** SPRight. The **EditScripts** SPRight is not included in any standard capabilities by default. If you want to give script editing abilities to users other than System Administrators, you must either create a new capability for these users that includes the **EditScripts** SPRight, or add the **EditScripts** SPRight to an existing capability that can be assigned to those users.

Because the ability to write scripts is a potential security risk, it is recommended that the **EditScripts** SPRight is given out in a controlled manner to trusted users only.

For more information on assigning capabilities to users, see the *IdentityIQ Identity Management Guide*.

For more information on rights and capabilities in IdentityIQ, see [IdentityIQ Rights and Capabilities - Definitions](#).

Importing Rule XML

Rules can be written as standalone XML objects and imported into IdentityIQ through the **gear > Global Settings > Import from File** option, or through the IdentityIQ console, using the `iiq console import` command. A single rule XML file can contain one or more rules.

For more information on importing files, see the *IdentityIQ System Configuration Guide* and the *IdentityIQ Console Guide*.

Sample Rule Files

IdentityIQ provides a set of sample rules that you can use as a model for creating your own. The sample rules are in the `examplerules.xml` file, which is located in your IdentityIQ installation directory under `<identityiq_home>\WEB-INF\config\`

In addition, a set of sample rules specifically for Rapid Setup is included with the Rapid Setup feature. Sample Rapid Setup rules are in the `rsexamplerules.xml` file, which is located in your IdentityIQ installation directory under `<identityiq_home>\WEB-INF\config\rapidsetup.`

Rule Libraries

Rule libraries are collections of methods that have been grouped together and stored in IdentityIQ as a Rule object. They contain sets of related but unconnected methods that can be invoked directly by workflow steps or other rules. These are stored as Rule objects, rather than in the compiled Java classes, so that their functionality can be easily modified to suit the needs of each installation.

IdentityIQ includes some rule libraries that are used by the default workflows. Examples of rule libraries are Workflow Library, Approval Library, and LCM Workflow Library, any of which can be viewed through the Debug pages or the `iiq` console.

You can create your own custom libraries to provide additional functionality as needed. To reference a rule library from another rule, include a `<ReferencedRules>` element in the rule XML, naming the rule library in the `<Reference>` element.

The methods within the library can then be invoked from within the rule's `Source` element.

```
<Rule...>
  <ReferencedRules>
    <Reference class='Rule' name='My Library' />
  </ReferencedRules>
  <Source>
    doSomething();
  </Source>
</Rule...>
```

```
</Source>  
</Rule>
```

API Reference for Rules

The IdentityIQ API is documented as Java documentation (javadocs). The documentation is included in every installation of IdentityIQ under `<identityiq_home>\doc\javadoc\`

Testing Rules

In many cases, rules need input. That is, the rules need instances of objects, meaning that live data is the best test environment. However, taking instances of objects is not always possible and could cause problems with the actual data. For example, if a mistake is made in a correlation rule, application accounts could be correlated to the wrong identities or remain uncorrelated.

SailPoint recommends separating development, testing, and production environments. Creating a "sandbox" with its own sample data is a recommended practice that allows for testing rules under realistic conditions with minimal risk to live data.

For debugging purposes, it can be useful to insert `println` statements in your rules' BeanShell code. Be sure to comment out or delete any `println` statements before making the rule live in a production environment.

Working With Incident Codes

Some errors that occur in IdentityIQ display a banner across the top of the UI, with a message similar to this:

"The system has encountered a serious error while processing your request. Please report the following incident code to your system administrator: <incident code>"

Incident code numbers are specific to your environment. You can use IdentityIQ's search function to see more detail about what the incident code is reporting:

1. Make a note of the incident code number.
2. Navigate to **Intelligence > Advanced Analytics**.
3. For **Search Type**, choose **Syslog**.
4. Enter the **Incident Code** and select **Run Search**.

When IdentityIQ retrieves the incident details, you can select the error to get a stack trace for the incident code.

The stack trace is typically very helpful for troubleshooting purposes. If you need to escalate the issue to SailPoint's Support team, submitting the stack trace with a support ticket can help expedite the troubleshooting process.