# Integration Service Platform

Version 1.7.0

# Table of Contents

1

# Chapter 1

# Introduction

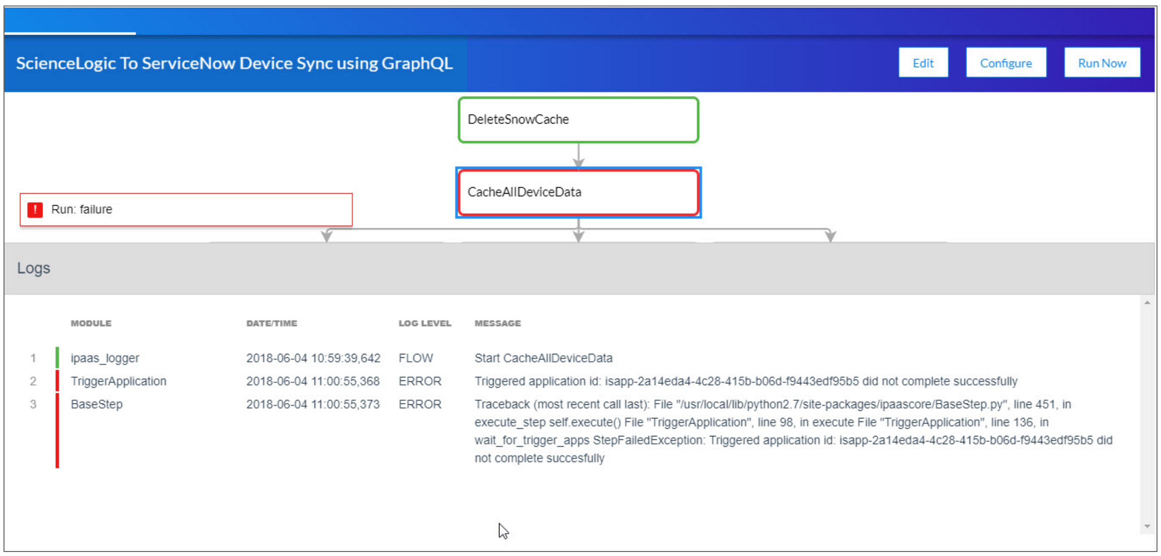
## Overview

The Integration Service provides a generic platform for integrations between SL1 and third-party platforms.

This chapter covers the following topics:

# What is the Integration Service?

The Integration Service enables intelligent, bi-directional communication between the ScienceLogic data platform and external data platforms to promote a unified management ecosystem. The Integration Service allows users to translate and share data between SL1 and other platforms without the need for programming knowledge. The Integration Service is designed to provide high availability and scalability.




The key elements of the Integration Service user interface include the following:

- ***Steps***. A step is a generic Python class that performs an action. Steps accept arguments and can be re-used. The arguments tell the step which variables and values to use when executing. Steps can also pass results to a subsequent step.
- ***Integration Applications***. An integration application is a JSON object that includes all the information required for executing an integration on the Integration Service platform. An integration application includes a list of steps and metadata for those steps. Each step will execute a single action and pass the results to subsequent, dependent steps. The parameters for each step are also defined in the application and can be provided either directly in the step or in the parent integration application.
- ***Configurations***. A configuration is a stand-alone JSON file that contains a set of configuration variables. Configurations live on the Integration Service system and can be accessed by *all integration applications and their steps*.

# What is a Step?

In an Integration Service system, a ***step*** is a generic Python class that performs a single action, such as caching device data:


Steps accept arguments called *parameters*. These arguments specify the values, variables, and configurations to use when executing the step. Parameters allow steps to accept arguments and allow steps to be re-used in multiple integrations. For example, you can use the same QueryREST step to query both the local system and another remote system; only the arguments, (hostname, username, and password) change.

A step can pass the data it generates during execution to a subsequent step. A step can use the data generated by another step.

The Integration Service system analyzes the required parameters for each step and alerts the user if any required parameters are missing before the Integration Service can run the step.

Steps are grouped into the following types:

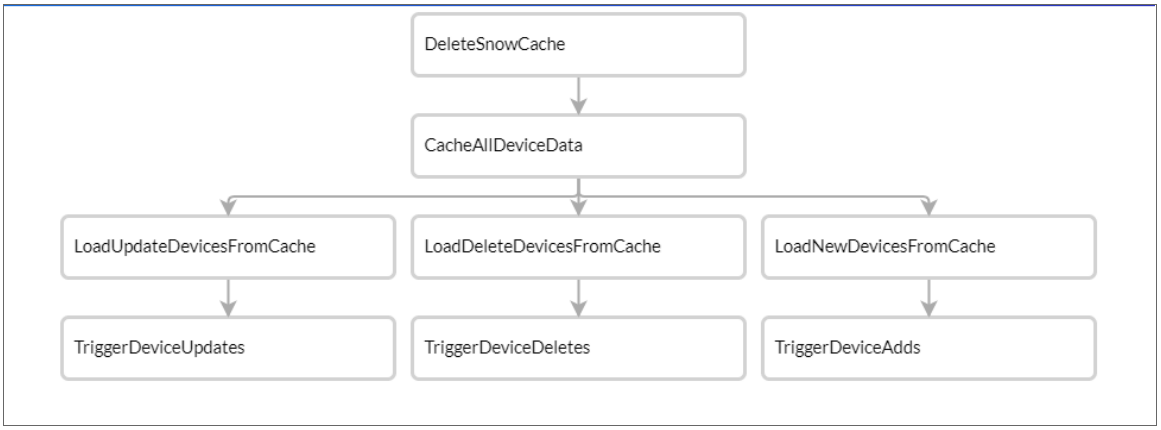
- ***Standard***. Standard steps do not require any previously collected data to perform. Standard steps are generally used to generate data to perform a transformation or a database insert. These steps can be run independently and concurrently.
- ***Aggregated***. Aggregated steps require data that was generated by a previously run step. Aggregated steps are not executed by the Integration Service until all data required for the aggregation is available. These steps can be run independently and concurrently.
- ***Trigger***. Trigger steps are used to trigger other integration applications. These steps can be configured to be blocking or not.

A variety of generic steps are available from ScienceLogic, and you can access a list of step parameters in the ***API endpoint***.

# What is an Integration Application?

In the Integration Service, an ***integration application*** is a JSON file that specifies which steps to execute and the order in which to execute those steps. An integration application also defines variables and provides arguments for each step.

The following is an example of an integration application:




Integration application JSON objects are defined by configuration settings, steps that make up the integration, and application-wide variables to use as parameters for each step. The parameters of each step can be configured dynamically, and each step can be named uniquely while still sharing the same underlying class, allowing for maximum re-use of code.

Integration applications can be executed through the REST API and are processed as an asynchronous task in the Integration Service. During processing the user is provided a unique task ID for the application and each of its tasks. Using the task IDs, the user can poll for the status of the integration application and the status of each individual running step in the integration application.

Executing an integration application from the REST API allows the user to dynamically set one-time parameter values for the variables defined in the integration.

The required parameters of integration applications are strictly enforced, and the Integration Service will refuse to execute the integration application if all required variables are not provided.

# What is a Configuration?

Configuration variables are defined in a stand-alone JSON file called a ***configuration*** that lives on the Integration Service system and can be accessed by all integration applications and their steps.

Each global variable is defined as a JSON object in the configuration. Typically, a configuration object looks like the following:

```
{"name": "var_name", "value":"var_value", "encrypted": true}
```

Each global variable in the configuration has the option of being encrypted. The values of encrypted variables are encrypted within the Integration Service upon upload through the REST API.

Chapter

2

2

# Installing and Configuring the Integration Service

## Overview

This chapter describes how to install and configure the Integration Service, and also how to set up security for the service.

This chapter covers the following topics:

# Prerequisites for the Integration Service

To work with the Integration Service, ScienceLogic recommends that you have knowledge of the following:

- vi or another text editor
- Linux
- Docker. More information on using the Docker command line can be found in the ***Helpful Docker Commands*** section and at https://docs.docker.com/engine/reference/commandline/cli/.
- Python
- Couchbase

In addition, you must give your Docker Hub ID to your ScienceLogic Customer Success Manager to enable permissions to pull the containers from Docker Hub.

# System Requirements

The Integration Service has the following system requirements:

- 8 CPUs
- 12 GB total RAM
- 100 GB total storage

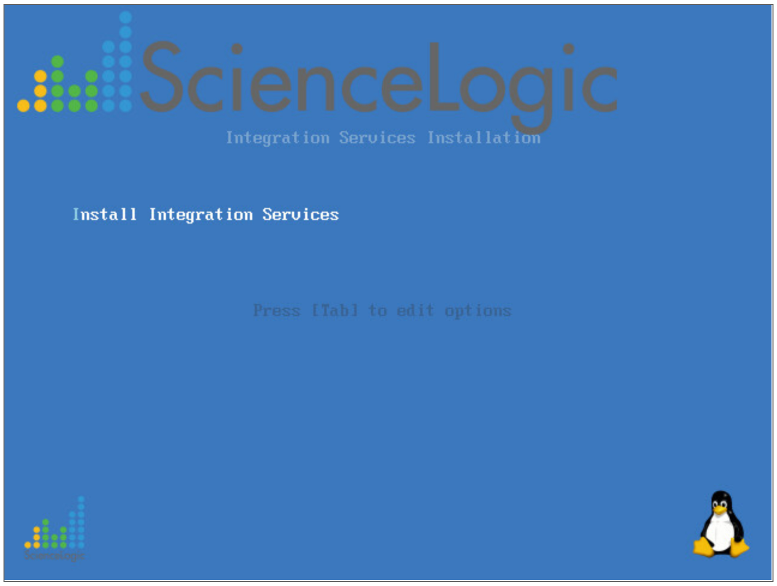
You should also use an **ap2** version of 5.54.9 or later of the new user interface for SL1. For more information, see the ***Introduction to the New User Interface*** manual.

# Installing the Integration Service

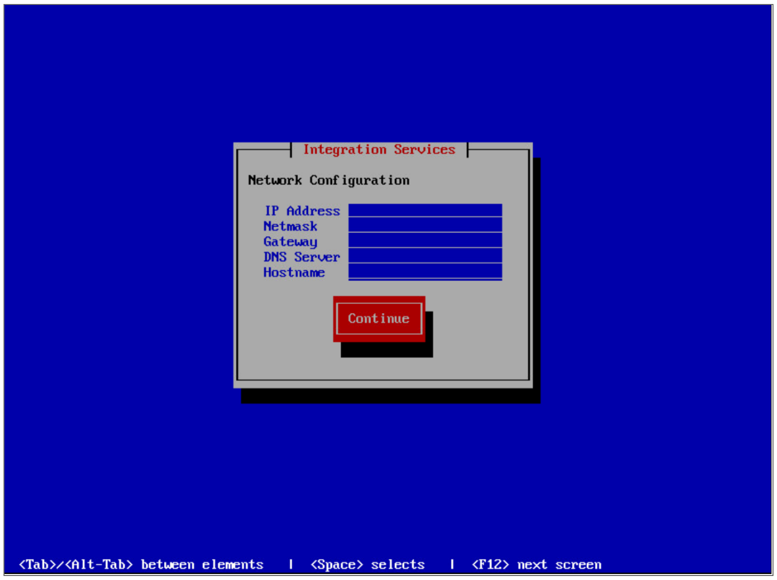
To install the Integration Service:

1. Download the latest Integration Service ISO file to your computer.

2. Using your hypervisor or bare-metal (single-tenant) server of choice, mount and boot from the Integration Service ISO. See the *System Requirements* section for prerequisites for your server. The Integration Service Installation window appears:



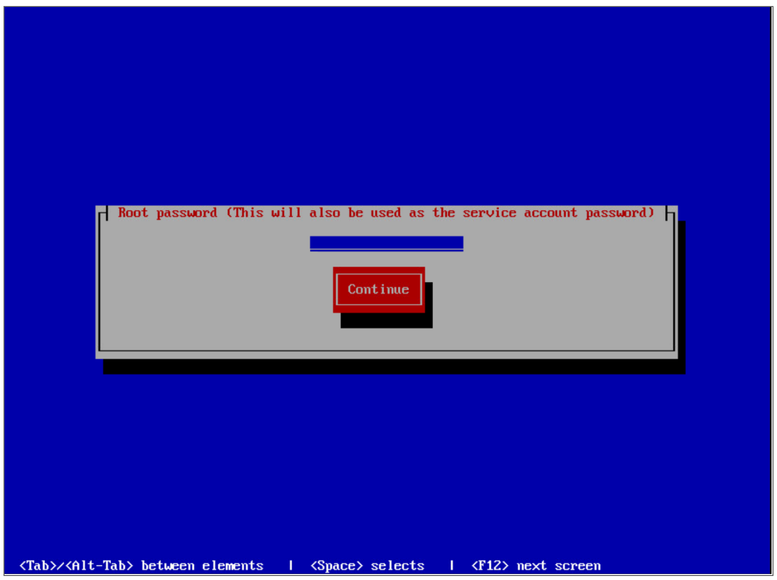

3. Select *Install Integration Service*. After the installer loads, the **Network Configuration** window appears:



4. Complete the following fields:

    - ***IP Address***. Type the primary IP address of the Integration Service server.
    - ***Netmask***. Type the netmask for the primary IP address of the Integration Service server.
    - ***Gateway***. Type the IP address for the network gateway.
    - ***DNS Server***. Type the IP address for the primary nameserver.
    - ***Hostname***. Type the hostname for the Integration Service.

5. Click **[Continue]**. The **Root Password** window appears:



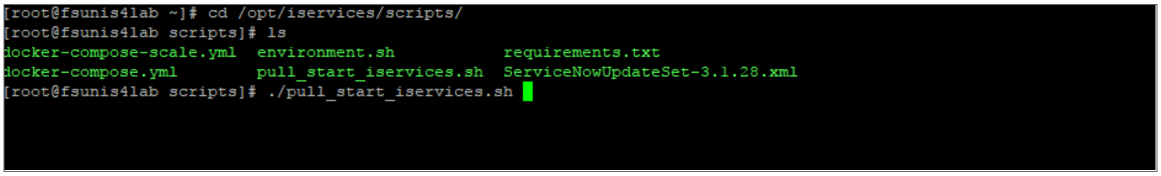

6. Type the password you want to set for the root user on the Integration Service host and press the "Enter" key.

| |
|---|
| **NOTE**: The password cannot contain spaces. |

7. Type the password for the root user again and press the "Enter" key. The Integration Service installer runs, and the system reboots automatically.
8. Click **[Save]**.
9. SSH into the newly-installed system using PuTTY or a similar application.
10. To start services, go to opt/iservices/scripts.

11. Execute the following command:

```
./pull_start_iservices.sh
```






12. Navigate to the Integration Service user interface using your browser. The address of the Integration Service user interface is:

    https://[*IP address entered during installation*]

13. Log in with the default username of *isadmin* and the password you specified above.

To verify that your stack is deployed, view your Couchbase logs by executing the following command using PuTTY or a similar application:

```
docker service logs --follow iservices_couchbase
```

If no services are found to be running, run the following command to start them:

```
docker stack deploy -c docker-compose.yml iservices
```

To add or remove additional workers, run the following command:

```
docker service scale iservices_steprunner=10
```

# Upgrading the Integration Service

If you are already running the Integration Service, perform the following steps to update the service from RPM:

1. Download the RPM and copy the RPM file to the Integration Service system.
2. Either go to the console of the Integration Service system or use SSH to access the server.
3. Log in as ***isadmin*** with the appropriate (root) password.
4. Type the following at the command line:

```
rpm -Uvh full_path_of_rpm
```

where:

- *full_path_of_rpm* is the full path and name of the RPM file.

5. If the upgrade process recommends restarting Docker, run the following command:

```
systemctl restart docker
```

6. After the RPM is installed, re-deploy the Docker stack to update the containers:

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml iservices
```

7. After you re-deploy the Docker stack, the services automatically update themselves. Wait a few minutes to ensure that all services are updated and running before using the system. You can use the visualizer at port 8080 to monitor the progress of the updates .
8. To view updates to each service, type the following at the command line:

```
docker ps
```

You will notice that each service now uses the new version of the Integration Service.

# Changing the Integration Service Password

To change the password for the Integration Service API and database communications:

1. Navigate to Integration Service system or use SSH to access the server.
2. Run the following command as root:

```
/opt/iservices/scripts/ispasswd
```

3. Follow the prompts to reset the iservices password. The password must be 6 characters or more and cannot be the same as the old password.

# Configuring a Proxy Server

To configure the Integration Service to use a proxy server:

1. Either go to the console of the Integration Service system or use SSH to access the Integration Service server.
2. Log in as ***isadmin*** with the appropriate password.
3. Using a text editor like "vi", edit the file **/opt/iservices/scripts/docker-compose-override.yml**.
4. In the "environment" section of the steprunner service, add the following lines:

```
services:
  steprunner:
    environment:
      https_proxy: "<proxy_host>"
      http_proxy: "<proxy_host>"
      no_proxy: "..."
```

5. Save the settings in the file and then run the script **/opt/iservices/compose_override.sh**.

**NOTE:** This script validates the syntax of your settings changes. If the settings are correct, the script applies the settings to your existing compose file.

6. rm and re-deploy the steprunners to use this change by typing the following commands:

```
docker service rm iservices_steprunner
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml iservices
```



# Configuring Security Settings

This topic explains how to change the HTTPS certificate used by the Integration Service, and it also describes password and encryption key security.

## Changing the HTTPS Certificate

The Integration Service API and user interface only accept communications over HTTPS. By default, HTTPS is configured using an internal, self-signed certificate.

You can specify the HTTPS certificate to use in your environment by mounting the following two files in the API and user interface containers:

- /etc/iservices/is_key.pem
- /etc/iservices/is_cert.pem

To specify the HTTPS certificate to use in your environment:

1. Copy over the key and certificate to the Integration Service host.
2. Modify the /opt/iservices/scripts/docker-compose-override.yml file and mount a volume to the "gui" and "contentapi" services. The following image contains an example of the volume specification:

```
volumes:
  - "path/to/is/key:/etc/iservices/is_key.pem"
  - "path/to/is/cert:/etc/iservices/is_cert.pem"
```

3. Run the following script to validate and apply the change:

```
/opt/iservices/scripts/compose_override.sh
```

4. Re-deploy the gui service by running the following commands:

```
docker service rm iservices_gui
docker service rm iservices_contentapi
/opt/iservices/scripts/pull_start_iservices.sh
```

## Using Password and Encryption Key Security

During platform installation, you can specify an Integration Service root password. This root password is also the default isadmin password.

- The root/admin password is saved in a root read-only file here: /etc/iservices/is_pass
- A backup password file is also saved in a root read-only file here: /opt/iservices/backup/is_pass

The user-created root password is also be the default Integration Service password for couchbase (:8091) and all API communications. The Integration Service platform generates a unique encryption key for every platform installation.

- The encryption key exists in a root read-only file here: /etc/iservices/encryption_key
- A backup encryption key file is also saved in a root read-only file here: /opt/iservices/backup/encryption_key

You can use the encryption key to encrypt all internal passwords and user-specified data. You can encrypt any value in a configuration by specifying `"encrypted": true,` when you POST that configuration setting to the API. There is also an option in the Integration Service user interface to select *encrypted*. Encrypted values use the same randomly-generated encryption key.

User-created passwords and encryption keys are securely exposed in the Docker containers using Docker secrets at https://docs.docker.com/engine/swarm/secrets/ to ensure secure handling of information between containers.

**NOTE**: The encryption key must be identical between two Integration Service systems if you plan to migrate from one to another. The encryption key must be identical between High Availability or Disaster Recovery systems as well.

# Helpful Docker Commands

- `docker service ls`. View available services running on the system.
- `docker service ps iservices`. View process status of all services.
- `docker service logs <service-name>`. View logs of a particular service. For example:
  - `docker service logs iservices_couchbase`
  - `docker service logs iservices_steprunner`
- `docker service scale iservices_steprunner=10`. Dynamically scale for more workers.
- `docker stack rm iservices`. Completely remove the services from running.
- `docker stack deploy -c <compose-file> iservices`. Deploy services from a defined Docker compose file.

# Chapter 3

# Backing up Data in the Integration Service

3

## Overview

This chapter describes how to create High Availability configurations to protect the data in the Integration Service. This chapter covers the following topics:

# Configuring a High Availability Environment for the Integration Service

Because the Integration Service uses the Docker Swarm tool to maintain its cluster and automatically re-balance services across nodes, ScienceLogic strongly recommends that you implement the best practices from Docker, Couchbase, and RabbitMQ. The following topics describe those best practices, along with requirements and frequently asked questions.

> **NOTE**: To support automatic failover of the Couchbase database without manual intervention, you must set up at least *three* nodes. You can achieve High Availability with two nodes, and no data will be lost in the event of a single node failure. However, with only two nodes, automatic failover will not function and you must manually perform the failover steps.

## Docker Swarm Requirements for High Availability

After implementing Docker Swarm High Availability, if a node goes down, all the services on that failed node can be dynamically re-provisioned and orchestrated among the other nodes in the cluster. High Availability for Swarm also facilitates network connections with the various other High Availability components.

Docker Swarm requires the following:

- The cluster contains at least three nodes running as managers. With three nodes, there can be a quorum vote between managers when a node is failed over.
- A load balancer with a virtual IP running in front of all nodes in the cluster. The load balancer allows user interface requests to be distributed among each of the hosts in the case one of the hosts fails.

For more information, see the Docker High Availability Documentation.

## Docker Swarm Frequently Asked Questions for High Availability

**What happens if I only use two nodes and one node fails?**

Using only two nodes does not meet Docker's High Availability requirements, so automatic High Availability and failover cannot be guaranteed. In the event of a failure of one out of two nodes, depending on which services fail, the Integration Service system might not be functional until a user logs in and performs some manual actions, such as removing the other failed node from the cluster.

After you perform these manual failover actions, the Integration Service will be back up and running.

**What happens if I use three nodes and two of the nodes fail?**

Docker fault tolerance is limited to one failure in a three-node cluster. If more than one node goes down in a three-node cluster, automatic High Availability and failover cannot be guaranteed, and manual intervention may be required. Adding more nodes is the only way to increase the fault tolerance.

In the event of a two out of three failure, after you perform manual failover actions, the Integration Service system will be back up and running.

For more information about the manual failover steps, see the ***Failover*** section.

## Couchbase Database Requirements for High Availability

Couchbase High Availability ensures that no integration application, configuration, or step data will be lost in the event of a node failure. To support automatic failover, Couchbase requires at least *three* nodes in the high availability cluster.

Each node will have an independent and persistent storage volume that is replicated throughout the cluster. Alternatively, shared storage can be used instead of independent persistent volumes. This replication ensures that data is replicated in all places, and if a single node goes down, no data will be lost.

For more information, see the Couchbase documentation.

## Couchbase Database Frequently Asked Questions for High Availability

**What if I don't have three nodes? If I only use two nodes, what happens during a failure?**

In the event of a failure of one out of two nodes, no data will be lost, because the data is being replicated. With only two nodes, automatic failover will not function, and you will need to perform manual failover actions. For more information about the manual failover steps, see the ***Failover*** section.

**What if I have three nodes and two of them fail?**

In the event of a failure of two out of three nodes, no data will be lost, because the data is being replicated.

If multiple Couchbase data nodes go down at the same time, automatic failover might not occur (not even nodes for quorum to failover). You will then need to perform manual failover steps. After you perform these manual actions, the Integration Service will be operational again. For more information about the manual failover steps, see the ***Failover*** section.

> **NOTE**: If you have three nodes, automatic failover is supported by Docker Swarm and Couchbase. If you have less than three nodes, follow the steps in the ***Failover*** section to manually fail over a system to regain Integration Service functionality.

## RabbitMQ Clustering and Persistence for High Availability

Implementing RabbitMQ High Availability ensures that if any integrations or tasks are waiting in the Rabbit queue, those tasks will not be lost if a node containing the Rabbit queue fails.

> **NOTE**: You can switch between both single-node and cluster options at any time during deployment.

RabbitMQ clustering requires a Docker Swarm configuration with multiple nodes. For more information, see *Configuring Docker Swarm*.

As a best practice for security, enable the user interface only temporarily during cluster configuration. The default user interface login is *guest/guest*.

## RabbitMQ Option 1: Persisting Queue to Disk on a Single Node (Default Configuration)

With this configuration, the Integration Service queue runs on a single node, and the queue is persisted on disk. As a result, if the Integration Service stack is removed and re-deployed, no messages are lost during the downtime. Any messages that exist in the queue before the stack is stopped continue to exist after the stack is re-deployed.

### Potential Risks and Mitigations

Because the queue runs on a single node, if that node fails, then the queue and its related data might be lost.

You can mitigate data loss by persisting the queues on your choice of network shared storage, so that if the queue fails on one node, the queue and its messages can be brought back up on another node.

### Requirements/Setup (Enabled by Default)

- You must define a static hostname for the RabbitMQ host in the docker-compose file. The default is *rabbit_node1.isnet*.
- You must mount a volume to /var/lib/rabbitmq in the docker-compose file.

### Example Compose Definition

```
rabbitmq:
  image: sciencelogic/is-rabbit:3.7.7-1
  hostname: rabbit_node1.isnet
  volumes:
    - "rabbitdb:/var/lib/rabbitmq"
  networks:
    isnet:
      aliases:
        - rabbit
        - rabbit_node1.isnet
```

## RabbitMQ Option 2: Clustering Nodes with Persistent Queues on Each Node

This configuration lets multiple nodes join a RabbitMQ cluster. When you include multiple nodes int he RabbitMQ cluster, all queue data, messages, and other necessary information is automatically replicated and persisted on all nodes in the cluster. If any node fails, then the remaining nodes in the cluster continue maintaining and processing the queue.

Because the RabbitMQ cluster includes disk-persisted queues, if all nodes in the Rabbit cluster fail, or if the service is removed entirely, then no data loss should occur. Upon restart, the nodes will resume with the same cluster configuration and with the previously saved data.

If you include multiple nodes in a RabbitMQ cluster, the Integration Service automatically applies an HA policy of all-node replication, with retroactive queue synchronization disabled. For more information, refer to the [RabbitMQ documentation.]()

### Potential Risks and Mitigations

If you create a Docker Swarm cluster with only two nodes, the cluster might stop functioning if a single node fails. To prevent this situation, include at least *three* nodes in each cluster.

### Requirements/Setup

For a Docker Swarm configuration with multiple independent nodes:

- Both RabbitMQ services must be "pinned" to each of the two nodes. See the **Example Compose Definition** below.
- You must add a new RabbitMQ service to the docker-compose.yml file. This new service should have a hostname and alias following the designated pattern. The designated pattern is: *rabbit_nodex.isnet*, where *x* is the node number. This configuration supports up to 20 clustered nodes by default.
- After you update the docker-compose.yml file, the nodes will auto-cluster when you perform a deployment.

### Example Compose Definition of Two Clustered Rabbit Services

```
rabbitmq:
  image: sciencelogic/is-rabbit:3.7.7-1
  hostname: rabbit_node1.isnet
  volumes:
    - "rabbitdb:/var/lib/rabbitmq"
  networks:
    isnet:
      aliases:
        - rabbit
        - rabbit_node1.isnet
  deploy:
    placement:
      constraints:
        - node.hostname == node-number-1.domain
rabbitmq2:
  image: sciencelogic/is-rabbit:3.7.7-1
  hostname: rabbit_node2.isnet
  volumes:
    - "rabbitdb:/var/lib/rabbitmq"

  networks:
    isnet:
      aliases:
        - rabbit
        - rabbit_node2.isnet
  deploy:
    placement:
        constraints:
          - node.hostname == node-number-2.domain
```

## Checking the Status of a RabbitMQ Cluster

This section contains commands and additional resources for administering your clusters.

To check the status of your clustered RabbitMQ environment:

1. Run `docker ps` and locate the iservices_rabbit container.
2. Run the following command on the RabbitMQ container:

```
docker exec -it [container_id] /bin/bash
```

You can run the following commands for more information:

- `rabbitmqctl cluster_status`. Returns information about the current cluster status, including nodes in the cluster, and failed nodes.
- `rabbitmqctl list_policies`. Returns information about current policies. Ensure that the ha-all policy is automatically set for your cluster.

For additional cluster-related administrative commands, see the RabbitMQ Cluster Management documentation page.

# Configuring Clustering and High Availability

This section describes how to configure clustering and High Availability with Docker Swarm and the Couchbase database, using either two nodes or three or more nodes.

> **NOTE**: This topic assumes you are using Integration Service ISOs for each node, which includes an initial Docker Swarm node configuration. The use of the Integration Service ISO is not required, however. You could instead deploy another node (without using the Integration Service ISO) and configure a Linux operating system based on Red Hat. You could then add that system to the swarm.

## System Requirements for High Availability

On each Docker Swarm node, SSH into the node and then run the following commands to open the proper firewall ports for Docker Swarm:

```
firewall-cmd --add-port=2377/tcp --permanent
firewall-cmd --add-port=7946/tcp --permanent
firewall-cmd --add-port=7946/udp --permanent
firewall-cmd --add-port=4789/udp --permanent
firewall-cmd --reload
```

Also, make sure that the /etc/iservices/is_pass and /etc/iservices/encryption_key are identical on all clustered nodes.

## Configuring Docker Swarm

To configure Docker Swarm for clustering (two nodes or three or more nodes) and High Availability:

1. If you do not already have Integration Service running in your environment, install the Integration Service on a single node. Doing this creates a single-node Docker Swarm manager. For more information, see ***Installing the Integration Service***.
2. SSH to the Docker Swarm manager and run the following command to retrieve the join token. Make note of the token, because you need it to join a node to the swarm:

```
docker swarm join-token manager
```

3. Run the following commands on each Docker Swarm manager that you want to join to the cluster:

```
docker swarm init
docker swarm join --token <join token> <swarm manager ip>:<port>
```

4. Run the following command to verify that the nodes have been added:

```
docker node ls
```

5. If you are using local images and not connecting to Docker Hub, load docker images on the other swarm nodes:

```
docker load -i /opt/iservices/images/is-api:1.x.x.tar
docker load -i /opt/iservices/images/is-couchbase:1.x.x.tar
docker load -i /opt/iservices/images/is-gui:1.x.x.tar
docker load -i /opt/iservices/images/is-worker:1.x.x.tar
docker load -i /opt/iservices/images/rabbitmq:x.tar
docker load -i /opt/iservices/images/redix:4.x.x.tar
docker load -i /opt/iservices/images/visualizer.tar
```

where:

- *x.x* is the version number of the .tar file.

## Configuring the Couchbase Database

In a Couchbase cluster you have a master and one or more worker nodes. At least one worker node is required for a Couchbase cluster.

To add a Couchbase worker node:

1. Add the following line to constrain the Couchbase container to a single Docker Swarm node at the bottom of the **Couchbase** section:

```
deploy:
...
  hostname: couchbase.isnet
  deploy:
    placement:
      constraints:
        - node.hostname == <name of Docker Swarm node>

  networks:
    isnet:
      aliases:
        - couchbase
        - couchbase.isnet

  environment:
    db_host: couchbase.isnet
```

2. Add the couchbase-worker section:

```
couchbase-worker:
  image: repository.auto.sciencelogic.local:5000/is-couchbase:feature-INT-1208-HA-
  IS-Services
  container_name: couchbase-worker.isnet
  volumes:
    - "/var/data/couchbase:/opt/couchbase/var"
  deploy:
    placement:
      constraints:
        - node.hostname == <name of Docker Swarm node>
  networks:
    isnet:
      aliases:
        - couchbase-worker
        - couchbase-worker.isnet

  hostname: couchbase-worker.isnet
  ports:
    - "8095:8091"
  secrets:
    - is_pass
    - encryption_key
  ulimits:
    nofile: 80000
    core: 100000000
    memlock: 100000000
  environment:
    TYPE: 'WORKER'
    AUTO_REBALANCE: 'true'
    db_host: 'couchbase'
  depends_on:
    - couchbase
```

**NOTE**: This deployment makes the Couchbase worker user interface available on port 8095 of the Docker Swarm stack. If the master node goes down, or if the primary Couchbase user interface is not available on port 8091, you can still access the secondary Couchbase user interface through port 8095.

3. Add couchbase-worker to the db_host setting for contentapi:

```
contentapi:
...
  environment:
  ...
    db_host: 'couchbase,couchbase-worker,couchbase-worker2'
```

4. All db_host vars in docker-compose should be in the following format:

```
db_host: 'couchbase,couchbase-worker,couchbase-worker2'
```

5. If using the override file, run the compose-override.sh script to generate the docker-compose.yml file.
6. Deploy the stack with couchbase-worker node:

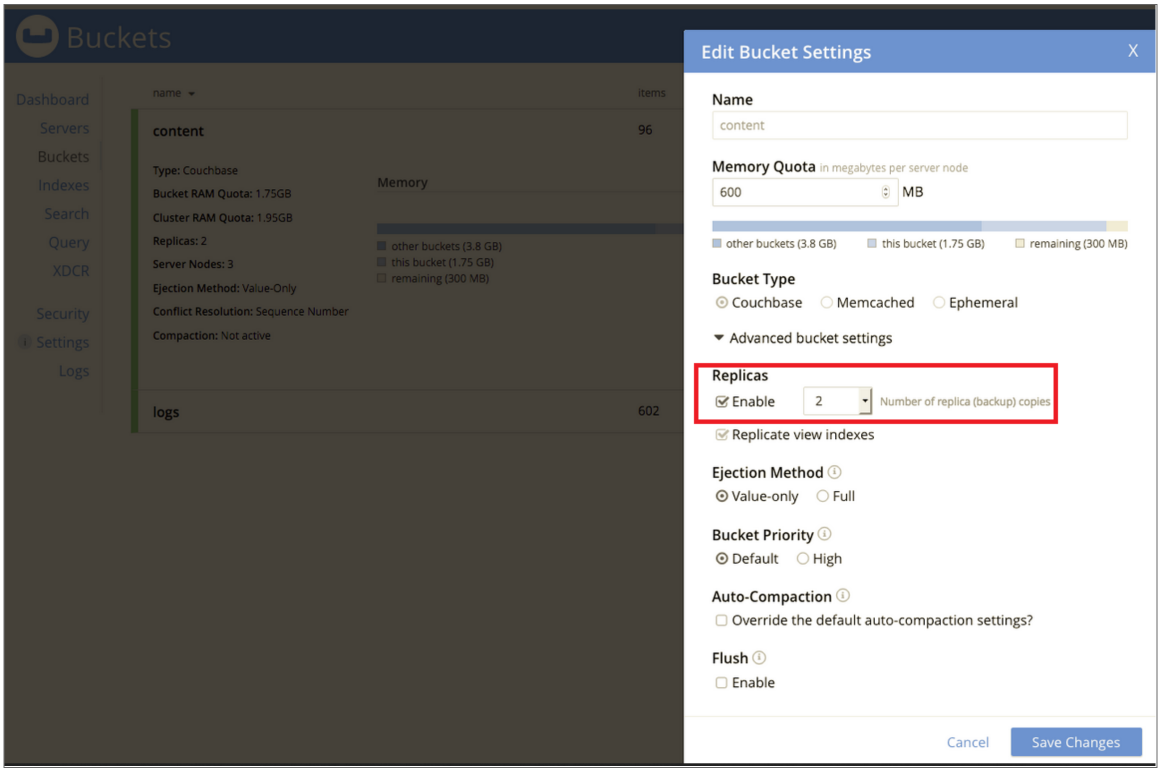
```
docker stack deploy -c <locate to compose file> --with-registry-auth iservices
```

7. After the two-node Couchbase cluster has been successfully deployed and the secondary indexes are successfully added, edit the replicas on couchbase-worker2 to 1 and run the following command:

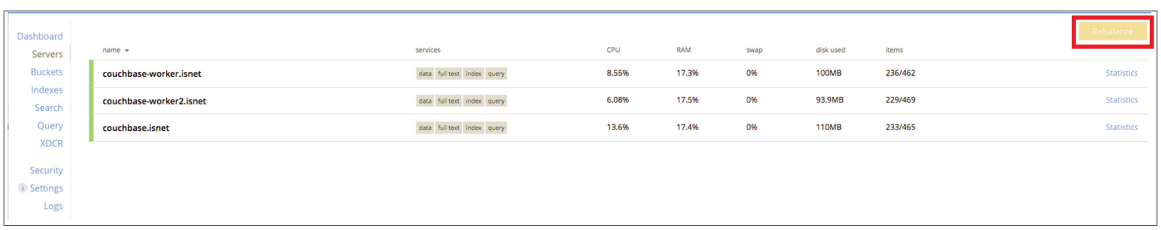
```
docker stack deploy -c <location of compose file> iservices
```

3

8. Set the replicas in the docker-compose-override.yml file as well.
9. After the second worker is added, set the number of replicas to "2" on each bucket (content and logs) in the Couchbase Administrator user interface and click **[Save Changes]**:




10. Rebalance the cluster by navigating to the **Servers** section of the Couchbase Administrator user interface and clicking the **Rebalance** button:




## Code Example: docker-compose-override.yml

The following section includes a complete example of the **/opt/iservices/scripts/docker-compose-override.yml** file for a three-node Couchbase and RabbitMQ clustered deployment:

| |
|---|
| **NOTE**: If shared volumes are available in the cluster, the deploy placement can be omitted and removed. |

```
version: '3.2'
services:
  steprunner:
```

```
    environment:
      db_host: couchbase,couchbase-worker2,couchbase-worker

  scheduler:
    environment:
      db_host: couchbase,couchbase-worker2,couchbase-worker

  couchbase:
    environment:
      db_host: 'couchbase'
    deploy:
      placement:
        constraints:
          - node.hostname == <swarm node hostname>
    networks:
      isnet:
        aliases:
          - couchbase
          - couchbase.isnet
    hostname: couchbase.isnet

  couchbase-worker:
    image: sciencelogic/is-couchbase:1.7.0
    container_name: couchbase-worker
    volumes:
      - "/var/data/couchbase:/opt/couchbase/var"
    ports:
      - "8100:8091"
    deploy:
      placement:
        constraints:
          - node.hostname == <swarm node hostname>
    networks:
      isnet:
        aliases:
          - couchbase-worker
          - couchbase-worker.isnet
    hostname: couchbase-worker.isnet
    secrets:
      - is_pass
      - encryption_key
    environment:
      TYPE: 'WORKER'
      AUTO_REBALANCE: 'true'
      db_host: 'couchbase'
    depends_on:
      - couchbase

  couchbase-worker2:
    image: sciencelogic/is-couchbase:1.7.0
    container_name: couchbase-worker2
    ports:
      - "8100:8091"
    volumes:
      - "/var/data/couchbase:/opt/couchbase/var"
    deploy:
      replicas: 0
      placement:
        constraints:
          - node.hostname == <swarm node hostname>
    networks:
```

3

```
      isnet:
        aliases:
          - couchbase-worker2
          - couchbase-worker2.isnet
    hostname: couchbase-worker2.isnet
    secrets:
      - is_pass
      - encryption_key
    environment:
      TYPE: 'WORKER'
      AUTO_REBALANCE: 'true'
      db_host: 'couchbase'
    depends_on:
      - couchbase

  rabbitmq:
    image: sciencelogic/is-rabbit:3.7.7-1
    hostname: rabbit_node1.isnet
    volumes:
      - "rabbitdb:/var/lib/rabbitmq"
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node1.isnet
      deploy:
        placement:
          constraints:
            - node.hostname == <swarm node hostname>

  rabbitmq2:
    image: sciencelogic/is-rabbit:3.7.7-1
    hostname: rabbit_node2.isnet
    volumes:
      - "rabbitdb:/var/lib/rabbitmq"
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node2.isnet
      deploy:
        placement:
          constraints:
            - node.hostname == <swarm node hostname>

  rabbitmq3:
    image: sciencelogic/is-rabbit:3.7.7-1
    hostname: rabbit_node3.isnet
    volumes:
      - "rabbitdb:/var/lib/rabbitmq"
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node3.isnet
      deploy:
        placement:
          constraints:
            - node.hostname == <swarm node hostname>

  contentapi:
```

```
environment:
  db_host: 'couchbase,couchbase-worker,couchbase-worker2'
```

### Scale iservices-contentapi

To scale out the iservices-contentapi to "3" to distribute the service across the three nodes, run the following command:

```
docker service scale iservices-contentapi=3
```

## Manual Failover

If you have a cluster with three or more nodes that is not configured with automatic failover, you must perform the following manual failover steps.

> **NOTE**: If you can access the Couchbase Administrator user interface (**http://<*IP of Integration Service*>:8091**) on the node that is still running, you can simply click the **[Failover]** button in the Couchbase Administrator user interface instead of manually running the couchbase-cli commands below.

> **NOTE**: In a three-node cluster, a single failed node will be automatically removed, you will still need to perform a re-balance.

To initiate a manual failover:

1. Log in to the Docker Swarm node where the node that is running resides.
2. Remove any failed managers from the cluster by running the following Docker commands:

   ```
   docker swarm init --force-new-cluster
   docker node rm <failed node id>
   ```

3. Run `docker ps` to identify the Container ID of the running Couchbase container.
4. Connect to the Docker container:

   ```
   docker exec -i -t <container id> /bin/bash
   ```

5. Identify the failed node by running the commands:

   ```
   couchbase-cli server-list -c couchbase -u isadmin -p <password>
   couchbase-cli server-list -c couchbase-worker -u isadmin -p <password>
   ```

3

6. One of the previous commands will show a failed node. Copy the IP address and port number of the failed node for step 7.
7. Use the currently running cluster and the failed node's IP address and port to run the following command to failover:

   ```
   couchbase-cli failover -c <couchbase|couchbase-worker> -u isadmin -p <password> --
   server-failover <ip:port> --force
   ```

   For example, if the functioning node is *couchbase-worker*, and the ip:port of the failed service is *10.0.0.4:4379*, then the command would be:

   ```
   couchbase-cli failover -c couchbase-worker -u isadmin -p <password> --server-
   failover 10.0.0.4:4379 --force
   ```

8. Rebalance the cluster using the functioning container name:

   ```
   couchbase-cli rebalance -c <cluster|cluster-worker> -u isadmin -p <password>
   ```

9. In the unlikely event that a failover occurs and no queries can be performed, validate that the indexes exist, and if not, rebuild them. To rebuild the primary indexes, run the following commands:

   ```
   cbq -u isadmin
   CREATE PRIMARY INDEX ON content;
   CREATE PRIMARY INDEX ON logs;
   ```

To recover a Docker Swarm node:

1. Re-deploy the node.
2. Add a new manager node to the swarm stack.

To restore the failed Couchbase node:

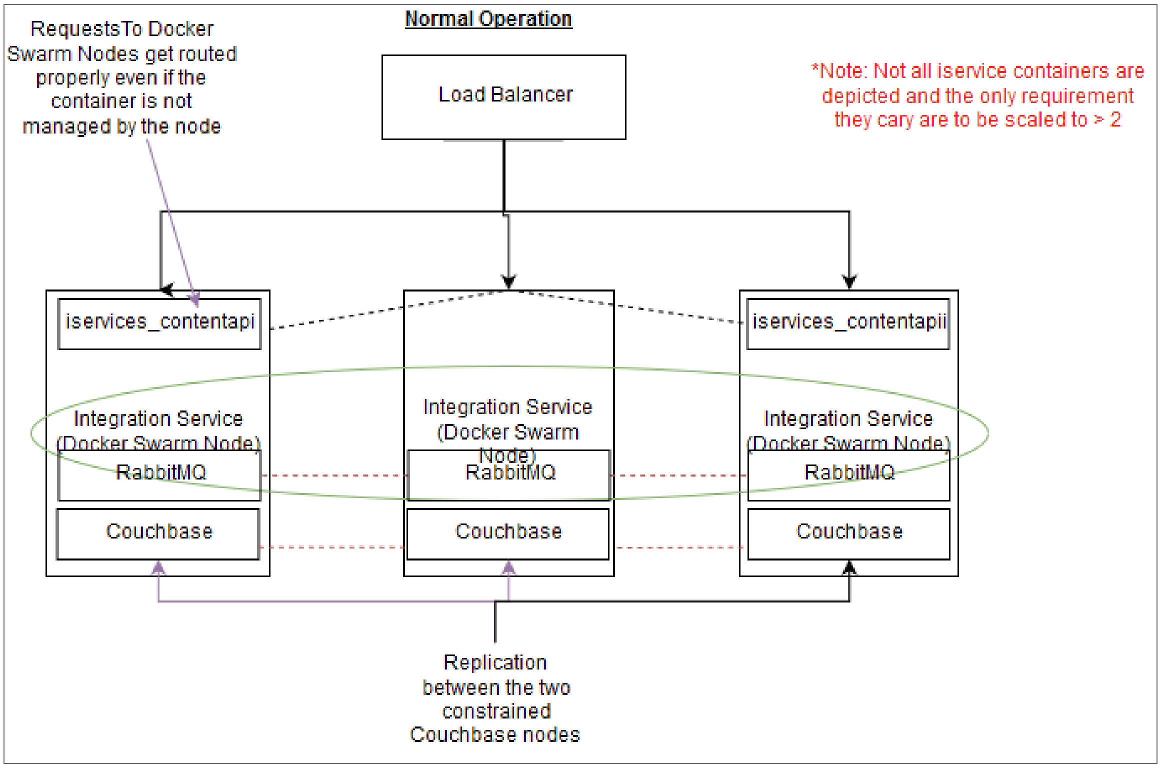
1. Log in to the node where the failed Couchbase cluster node was pinned.
2. Run *one* of the following commands, depending on the Couchbase node being recovered:
   - `docker service scale iservices_couchbase=0`
   - `docker service scale iservices_couchbase-worker=0`
3. If the Docker Swarm node was restored and not rebuilt, remove files from the old container:

   ```
   rm -rf /var/data/couchbase/*
   docker service scale iservices_couchbase scale 1
   ```

   A new node is added to Couchbase that will automatically re-balance the cluster after it is added.

# Additional Configuration Information

The following diagram describes a typical High Availability configuration that uses load balancing, replication, and failover:




## HAProxy Configuration (Optional)

The following example configuration describes using HAProxy as a load balancer:

```
...

frontend http_front
  bind *:80
  bind *:443
  option tcplog
  mode tcp
  tcp-request inspect-delay 5s
  default_backend http_back

backend http_back
  mode tcp
  balance roundrobin
  server master1 <docker swarm node 1 ip>:443 check
  server master2 <docker swarm node 2 ip>:443 check
  server master3 <docker swarm node 3 ip>:443 check
```

# Troubleshooting

## A service is unresponsive, or a service fails to start

If you cannot connect another service to a Docker service, or if the Docker service does not seem to be functioning properly, you can restart the service by running the following commands on the Integration Service system:

```
docker service rm iservices_<service-name>
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml iservices
```

## Docker networking issues

If your Integration Service system encounters Docker-related issues, such as overlay networks that are not accessible or service aliases that are not available, you can restart Docker by running the following command on the Integration Service system:

```
systemctl restart docker
```

> **NOTE**: Running this command restarts the Integration Service system system, and the system will not be available during the restart.

## Scaling services up or down

In some circumstances, you might want to scale *up* a service for more efficiency or to ensure fewer points of failure. In other situations, you might want to scale *down* a service to conserve CPU and memory.

To scale services, run the following command on the Integration Service system:

```
docker service scale iservices_<service-name>=x
```

Examples:

```
docker service scale iservices_steprunner=8
docker service scale iservices_contentapi=1
```

## A Couchbase worker failed to connect to master

A connection failure might happen a few times when a stack is freshly deployed. You can ignore these messages, and the worker should eventually connect to the master.

## A Couchbase worker service does not start

If a Couchbase cluster is not clustered properly, the cluster might have multiple nodes that think they are the master. To resolve this issue, run the following command on the problematic Couchbase service:

```
rm -rf /var/data/couchbase/*
```

**CAUTION:** Running this command clears *all* saved data from the node where that Couchbase service is running. Run this command only if your data is already elsewhere in the Couchbase cluster.

3

# Chapter

# 4

# Managing Integration Applications

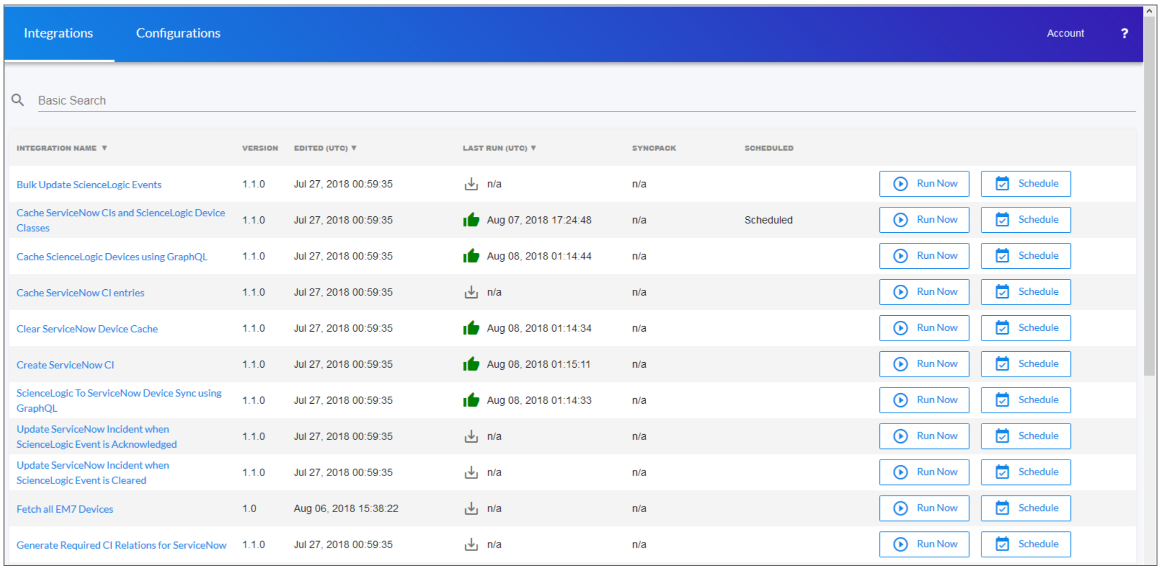
## Overview

This chapter describes how to use the Integration Service Application Registry and Editor to run and schedule integrations.

This chapter covers the following topics:

# Viewing the List of Integration Applications

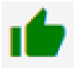
The **Integration Application Registry** page allows you to view a list of integration applications on your system. From this page you can view, run, and schedule applications:




For each integration application, the page displays the following items:

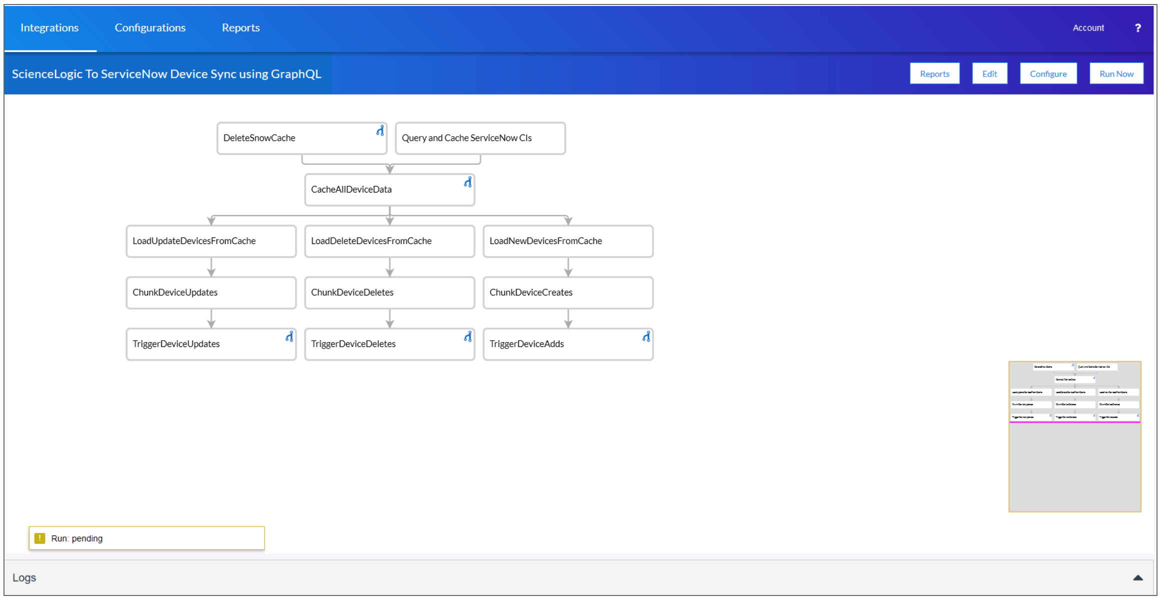
- ***Integration Name***. Name of the integration application. Click the column heading to order the integrations in ascending or descending order by name.
- ***Version***. Version of the integration application.
- ***Edited***. The last time the integration application was edited.
- ***Last Run***. The last time the integration application was run.
- ***SyncPack***. The SyncPack associated with the integration application.
- ***Scheduled***. Indicates if the integration application is scheduled to run.
- ***Run Now***. Click this button to run the integration application.
- ***Schedule***. Click this button to schedule a time to run the integration application.

In the **Last Run** column on the **[Integrations]** tab, you can also view the current status of an integration application:

| Icon | Status |
|---|---|
|  | The integration application ran successfully. |
|  | The integration application is currently running. |
|  | The integration application failed to run successfully. |
|  | The integration application has not been run. |

# Using the Integration Application Editor

When you click the name of an integration application, an **Integration Application Editor** page for that application appears:




In the main pane of the page, the steps for the application are organized as a flowchart. The arrows indicate the order in which the steps will execute when you run the application.

If a step triggers a child application, a branch icon ( ) appears in the upper right-hand corner of the step. You can double-click the branch icon to open the child application, or you can click the branch icon once to display the triggered application's run ID as a link in a pop-up window. If no run ID is present, the branch icon displays "NONE".

The buttons let you view reports for the integration application, and edit, configure, or run the integration application.

In the bottom left-hand corner of the page, you can view the status of the application. In the example above, the status is "Run: pending". Below the status is the **Logs** pane, which displays the logs from a step you selected in the main pane.

In the bottom right-hand corner of the page is a smaller version of the application. You can click and drag on this version to move or scroll through the steps in the main pane.

# Default Steps

The Integration Service system includes some already-defined steps:

- MicrosoftSqlDescribe
- MicrosoftSqlInsert
- MicrosoftSqlSelect
- MySqlDescribe
- MySqlInsert
- MySqlSelect
- QueryGQL
- QueryREST
- stepTemplate

To view the code for one of these steps:

1. Use Postman, cURL, or another REST API tool, or use the API `GET /steps` to download the steps:

   *URL_for_your_Integration_Service_system*/api/v1/steps/*step_name*

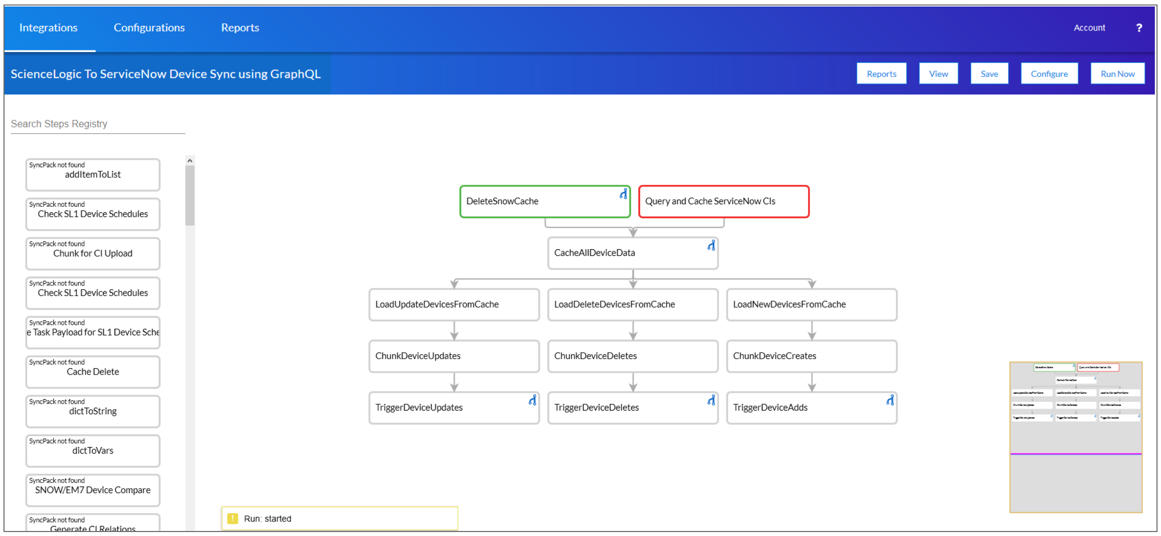
   where:

   - *URL_for_your_Integration_Service_system* is the IP address or URL for the Integration Service.
   - *step_name* is the name of the step you want to view.

# Editing an Integration Application

To edit an integration application:

1. From the **Integration Application Editor** page, click the **[Edit]** button. The **Search Steps Registry** pane appears, with a list of all of the steps that are available for that integration:




2. Scroll through the **Search Steps Registry** pane or use the **Search** field at the top of the pane to find the step you want to add.
3. Click the step you want to add, drag it to the main pane of the **Integration Application Editor** page, and drop it into the integration application .
4. To adjust the position of any step in the integration application, click the step you want to move and drag it to its new location.
5. To redirect the arrows connecting the steps, click an arrow and drag it to reposition it.
6. To remove a step, click the step to select it and press the **[Delete]** key on your keyboard.
7. To save the changes you made to the integration application, click the **[Save]** button.
8. To stop editing and close the **Search Steps Registry** panel, click the **[View]** button.

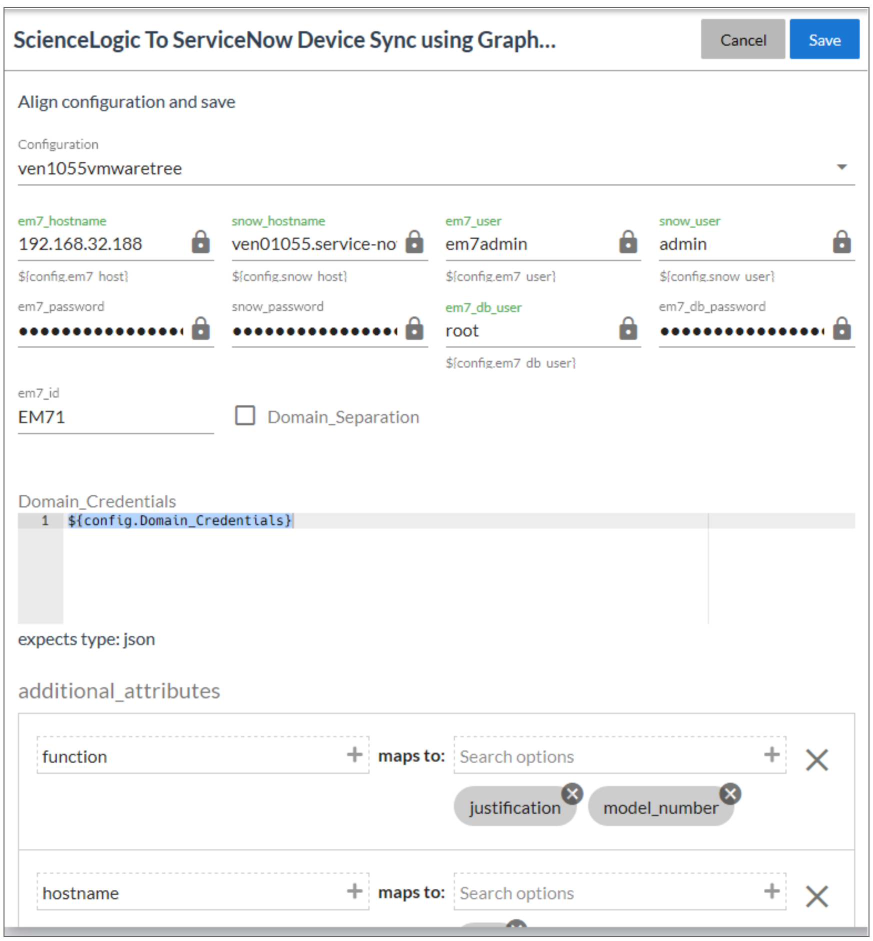
**TIP:** If the main pane has too many steps to see without scrolling, you can zoom in or out by clicking and holding the wheel on your mouse. You can also use the pane in the bottom right-hand corner to click on a part of the integration application that you want to see, and it will move the screen to focus on that part.

# Configuring an Integration Application

Before you can run an integration application, you must align the application with a configuration from the **[Configurations]** tab. A ***configuration*** defines global variables, such as endpoints and credentials, that can be used by multiple steps and integration applications. Where relevant, you can also edit the sections for the **additional_attributes** and **mappings** parameters to update or add new mappings between SL1 and another application.

To configure an integration application:

1. On the **[Integrations]** tab, select the integration application you want to configure.
2. On the **Integration Application Editor** page, click the **[Configure]** button. The **Configuration** pane opens on the right side of the window:




3. Select a configuration from the **Configuration** drop-down list to "align" to this integration application. This step is required.

4. To update the device attribute mappings for this integration application, scroll down to the section for the **additional_attributes** parameter and click the **[Add Mapping]** button to add a custom attribute, or edit an existing attribute that you want to map between SL1 and another application.

> **TIP**: Use the **[Tab]** button to move down through the list of options in a **Mapping** dropdown list, press **[Shift]**+ **[Tab]** to move up, and press **[Enter]** to select a highlighted option.

5. To update the device class and asset mappings, scroll down to the section for the **mappings** parameter and click the **[Add Mapping]** button to add a custom class or asset, or edit an existing class or asset that you want to map between SL1 and another application.
6. As needed, edit the other configuration values for the application.

> **NOTE:** To prevent potential issues with security and configuration, the fields related to configuration and any fields that are encrypted on the **Configuration** pane for an integration application cannot be edited.

7. When you are finished, click the **[Save]** button.

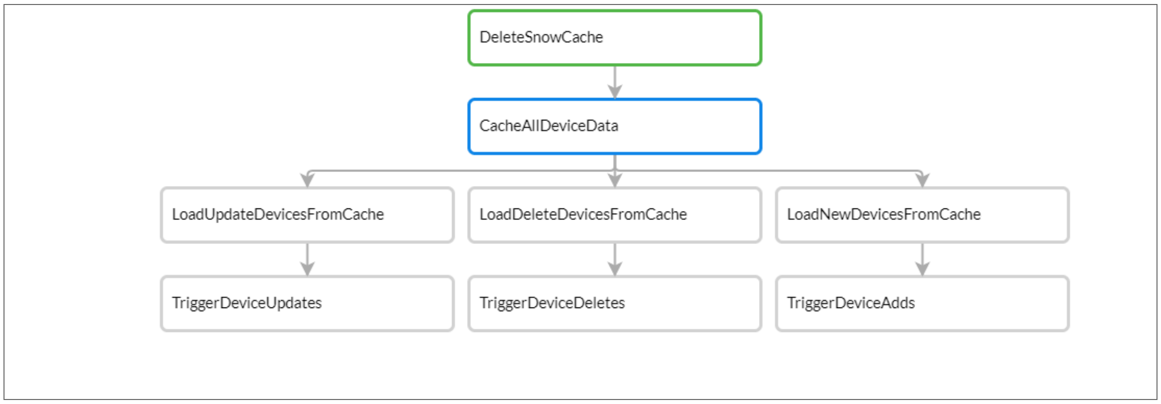
> **NOTE:** If you have created a new configuration in your own text editor, you must upload it to your Integration Service instance using the command line tool. After you upload it to the instance, it appears in the **Configuration** drop-down field on the **Configuration** pane. For more information, see the ***Integration Service for Developers*** manual.

4

## Running or Stopping an Integration Application

To run an integration application:

1. Click the **[Run Now]** button from the **Integrations** window or the individual application's window.
2. As the application runs, the color of the border around each step represents whether it is running, successful, or has failed:

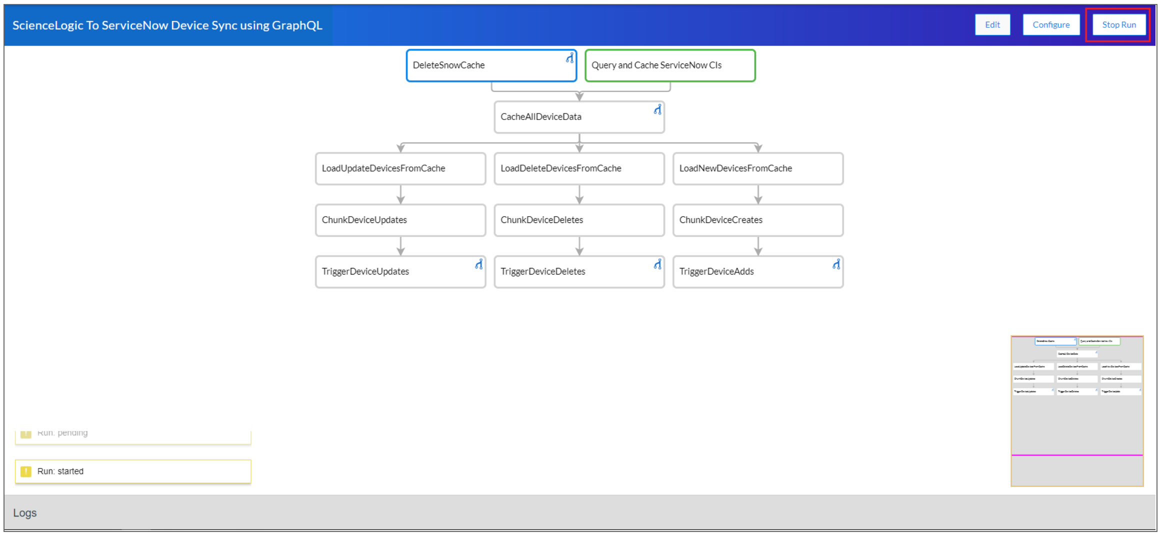


| Step Color | State |
|---|---|
| Blue | Running |
| Green | Successful |
| Red | Failed |

**NOTE:** Pop-up status messages also appear in the bottom left-hand corner of the **Integration Application Editor** page to update you on the progress of the application status and alert you of any errors.


3. If a step triggers a child application, a branch icon () appears in the upper right-hand corner of the step:

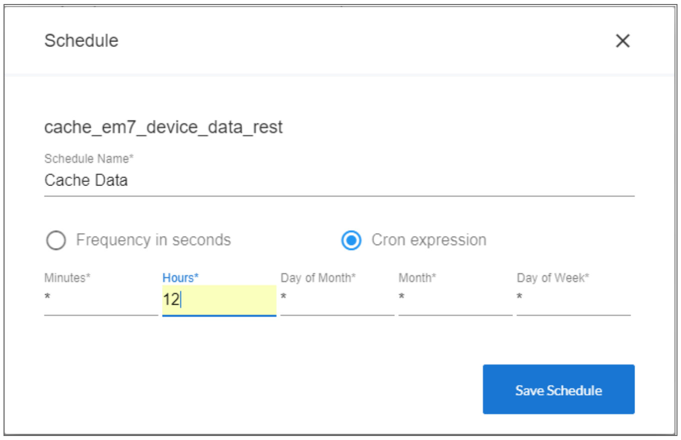
4. Double-click the branch icon to open the child application. Click the branch icon once to display the triggered application's run ID as a link in a pop-up window. If no run ID is present, the branch icon displays "NONE".
5. To stop the integration while it is running, click the **[Stop Run]** button. The **[Stop Run]** button is the same as the **[Run Now]** button, but toggles to **[Stop Run]** when you run the integration.




# Scheduling an Integration Application

To schedule an integration application:

1. On the **[Integrations]** tab of the Integration Service user interface, click the **[Schedule]** button for the integration application you want to schedule. The **Schedule** window appears:



2. In the **Schedule** window, complete the following fields:

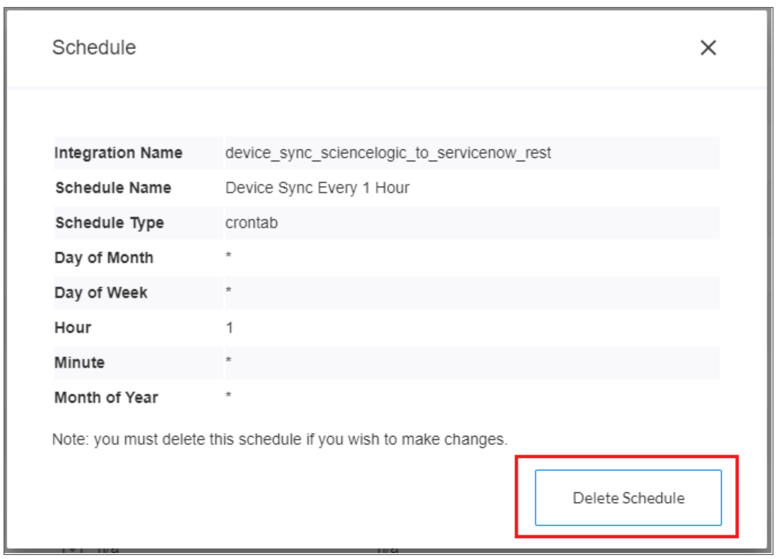
   - ***Schedule Name***. Type a name for your schedule.
   - ***Frequency in seconds***. Schedule the integration to run in an interval of seconds.
   - ***Cron expression***. Schedule the integration using a cron expression.

3. Click **[Save Schedule]**. The word "Scheduled" displays in the **Scheduled** column for this integration application.

After you create a schedule, it continues to run until you delete it.

To delete a schedule:

1. Click the **[Schedule]** button for the integration application whose schedule you wish to delete. A **Schedule** window appears with information about the application's current schedule.

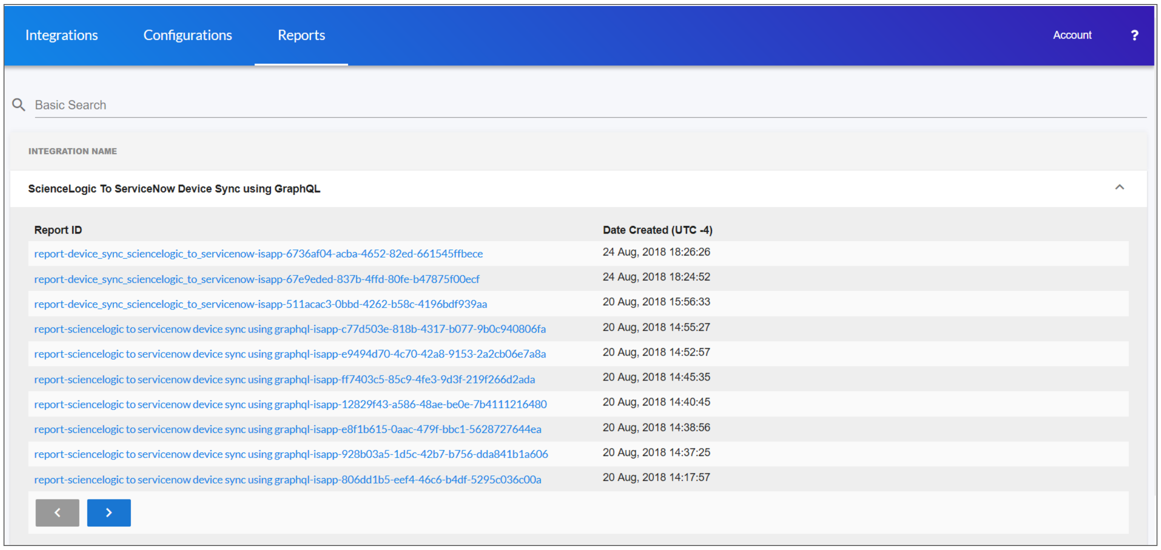



2. Click the **[Delete Schedule]** button. The schedule is removed.

# Viewing a Report for an Integration Application

In the Integration Service user interface, the **[Reports]** tab contains a list of reports associated with integration applications. If an integration application has the reporting feature enabled and the application supports reports, then the Integration Service will generate a report each time you run the integration application.

An individual report displays data only from the most recent run of the integration application; a report is not an aggregation of all previous runs.




**NOTE**: Not all integration applications generate reports. Currently, only the "ScienceLogic To ServiceNow Device Sync using GraphQL" integration application supports the generation of reports.

An integration application report includes the following fields:

- ***Device Name***
- ***IP Address***
- ***SL1 Device ID***
- ***ScienceLogic URL***
- ***ServiceNow Sys ID***
- ***Status***. The current state of the synced item, which can include *New*, *Removed*, *Updated*, or *Unchanged*.

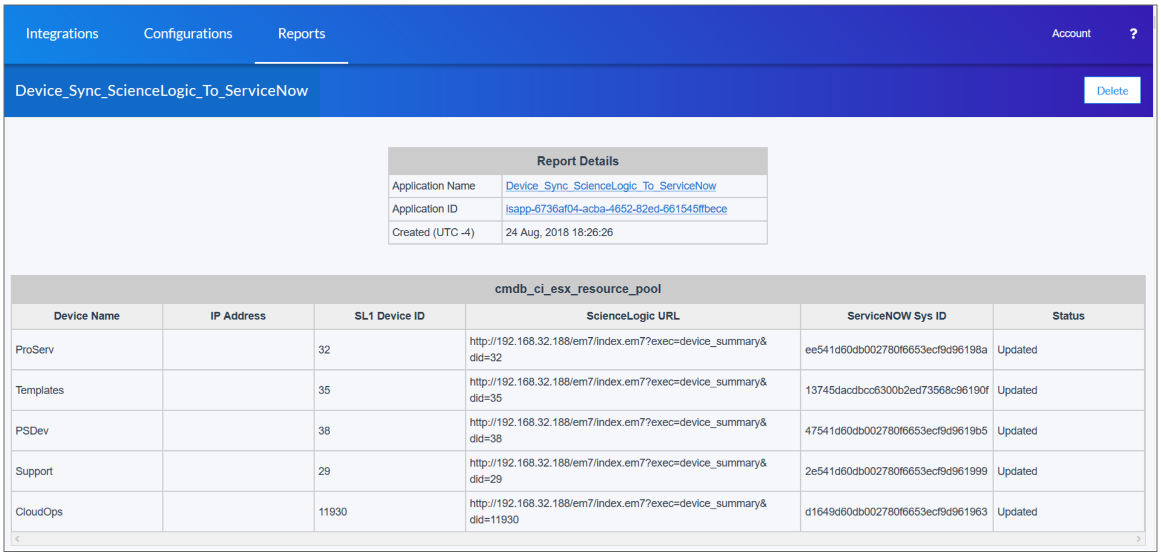
4

To view details for an integration application report:

1. On the **[Reports]** tab, click the name of the integration application to expand the list of reports for that application.

> **TIP**: Click the arrow buttons to scroll forward and back through the list of reports.

2. Click a report name in the **Report ID** column. The **Report Details** page appears:




| Report Details | |
|---|---|
| Application Name | Device_Sync_ScienceLogic_To_ServiceNow |
| Application ID | isapp-6736af04-acba-4652-82ed-661545ffbece |
| Created (UTC -4) | 24 Aug, 2018 18:26:26 |

cmdb_ci_esx_resource_pool

| Device Name | IP Address | SL1 Device ID | ScienceLogic URL | ServiceNOW Sys ID | Status |
|---|---|---|---|---|---|
| ProServ | | 32 | http://192.168.32.188/em7/index.em7?exec=device_summary&did=32 | ee541d60db002780f6653ecf9d96198a | Updated |
| Templates | | 35 | http://192.168.32.188/em7/index.em7?exec=device_summary&did=35 | 13745dacdbcc6300b2ed73568c96190f | Updated |
| PSDev | | 38 | http://192.168.32.188/em7/index.em7?exec=device_summary&did=38 | 47541d60db002780f6653ecf9d9619b5 | Updated |
| Support | | 29 | http://192.168.32.188/em7/index.em7?exec=device_summary&did=29 | 2e541d60db002780f6653ecf9d961999 | Updated |
| CloudOps | | 11930 | http://192.168.32.188/em7/index.em7?exec=device_summary&did=11930 | d1649d60db002780f6653ecf9d961963 | Updated |

3. To view the detail page for the integration application on the **[Integrations]** tab, click the **Application Name** link.

> **TIP**: From the detail page for the integration application, click the **[Reports]** button to return to the **[Reports]** tab.

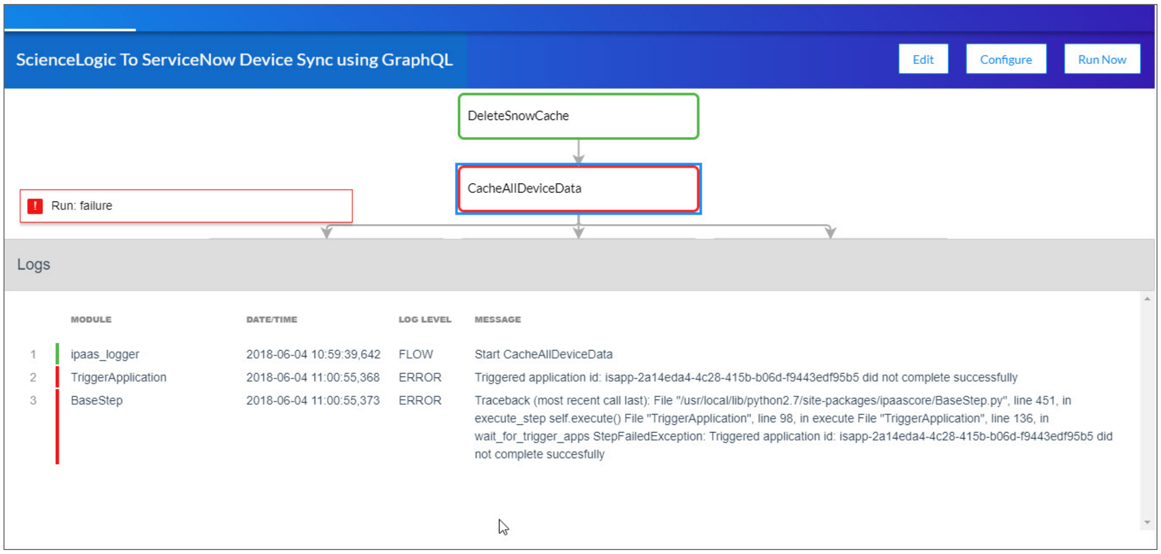
4. To view the specific run-time instance for the integration application that generated the report, click the **Application ID** link.
5. To delete a report, click the **[Delete]** button. Click **[OK]** to delete the report.

# Viewing Logs for an Integration Application

You can view logs for each step in an integration application on the **Integration Application Editor** page in the Integration Service user interface.

To view logs for the steps of an integration application:

1. From the **[Integrations]** tab, select an integration application. The **Integration Application Editor** page appears.
2. Select a step in the integration application.
3. Click **Logs** in the bottom left-hand corner of the screen. The **Logs** pane appears at the bottom of the page, and it displays status messages for the selected step:




> **TIP:** Click the gray area of the **Logs** pane to close the pane.

## Creating and Saving Integration Service Components

Instead of using the Integration Service user interface, you can create steps, integration applications, and configurations in your own editor and then upload them using the API or the command line interface (CLI).

For more information, see the ***Integration Service for Developers*** manual.

# Chapter 5

# Managing Configurations

## Overview

On the **[Configurations]** tab of the Integration Service user interface, you can create a configuration to define global variables that all steps and integration applications can use.

This chapter covers the following topics:



5

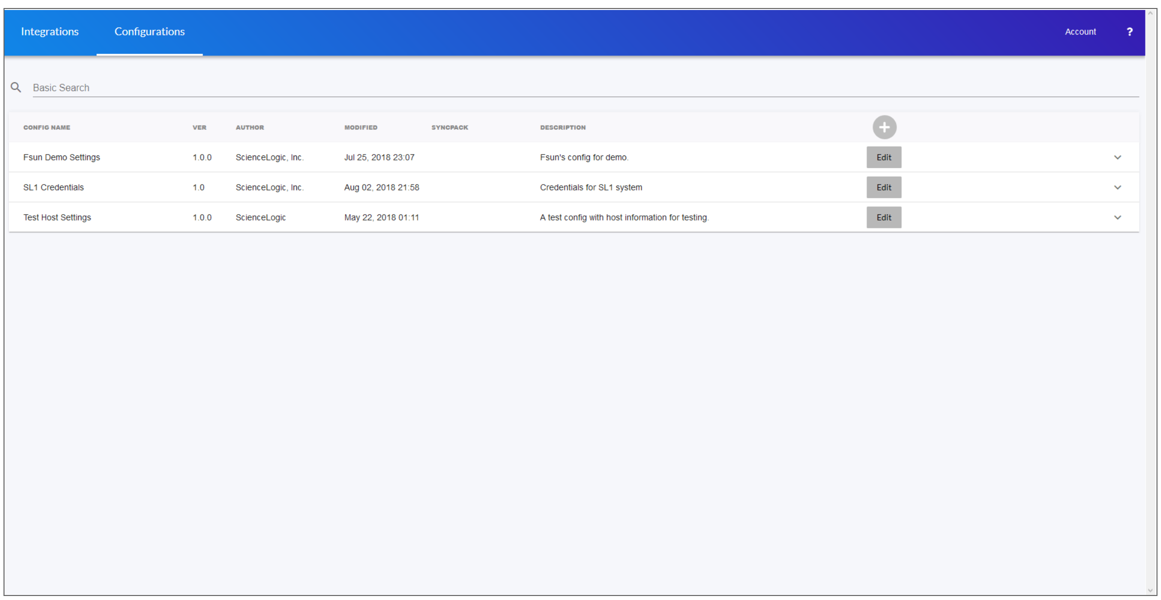
# What is a Configuration?

A ***configuration*** is a stand-alone JSON file that lives on the Integration Service system. A configuration defines global variables that can be used by all steps and integration applications.

After you create the configuration, it appears in the **Configuration** drop-down field on the **Configuration** pane of the **[Integrations]** tab. Before you can run an integration application, you must select a configuration and "align" that configuration with the integration application.

You can include the **config.** prefix with a variable to tell the Integration Service to use a configuration file to resolve the variable. If you want to re-use the same settings between applications, such as hostname and credentials, define configuration variables.

The **Configuration Registry** page displays a list of available configurations. From this page you can create and edit configurations:



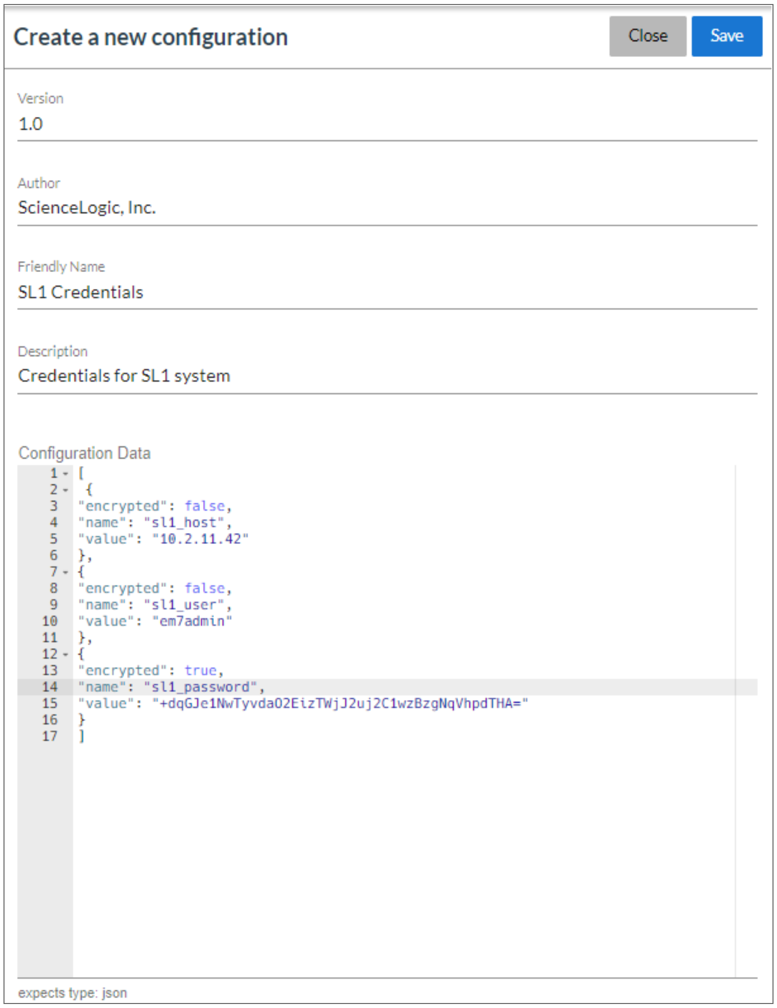

For each configuration, the page displays:

- ***Config Name***. Name of the configuration.
- ***Ver***. Version of the configuration.
- ***Author***. User or organization that created the configuration.
- ***Modified***. The date and time the configuration was created or last edited.
- ***SyncPack***. The SyncPack associated with the configuration.
- ***Description***. A brief description of the configuration.

# Creating a Configuration

To create a new configuration:

1. Navigate to the **Configuration Registry** page (the **[Configurations]** tab).
2. Click the plus icon ( ). The **Create a new configuration** pane appears:




3. Complete the following fields:

- ***Version***. Version of the configuration.
- ***Author***. User or organization that created the configuration.
- ***Friendly Name***. Name of the configuration.
- ***Description***. A brief description of the configuration.

4. In the ***Configuration Data*** field, specify the variable definitions:

   For example, you could add the following JSON code to the ***Configuration Data*** field:

```
[
  {
    "encrypted": false,
    "name": "em7_host",
    "value": "10.2.11.42"
  },
  {
    "encrypted": false,
    "name": "em7_user",
    "value": "em7admin"
  },
  {
    "encrypted": true,
    "name": "em7_password",
    "value": "+dqGJe1NwTyvdaO2EizTWjJ2uj2C1wzBzgNqVhpdTHA="
  }
]
```

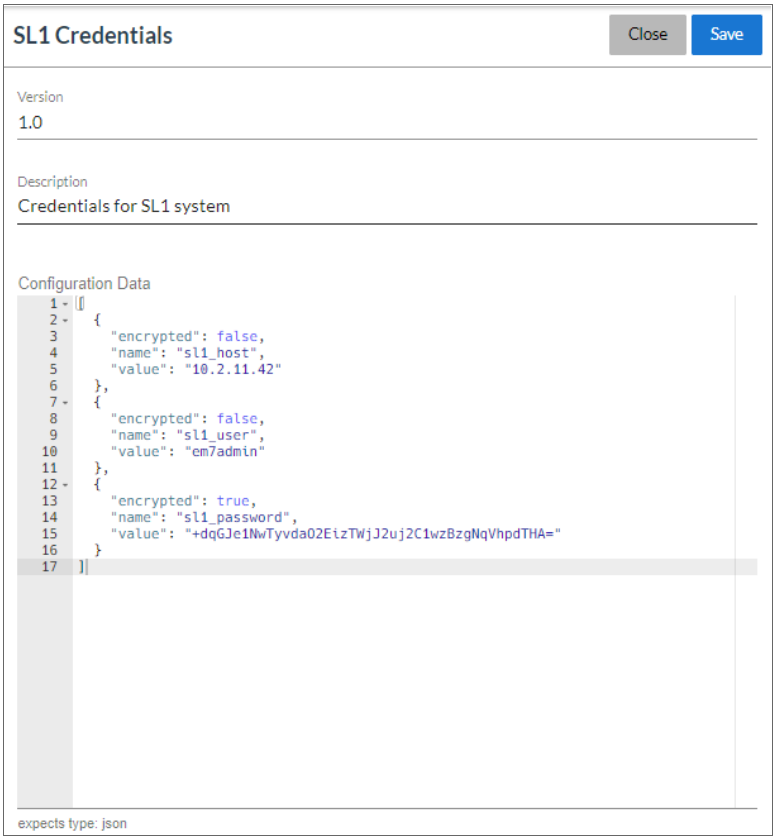
5. When creating a configuration variable, note the syntax:
   - The configuration file is surrounded by square brackets.
   - Each variable definition is surrounded by curly braces.
   - Each key name is surrounded by double-quotes and followed by a colon, while each value is surrounded by double-quotes and followed by a comma.
   - Each key:value pair in the definition is separated with a comma after the closing curly brace. The last key:value pair should not include a comma.
6. To create a configuration variable, define the following keys:
   - ***encrypted***. Specifies whether the value will appear in plain text or encrypted in this JSON file. If you set this to "true", when the value is uploaded, the Integration Service encrypts the value of the variable. The plain text value cannot be retrieved again by an end user. The encryption key is unique to each Integration Service system. The value is followed by a comma.
   - ***name***. Specifies the name of the configuration file, without the JSON suffix. This value appears in the user interface. The value is surrounded by double-quotes and followed by a comma.
   - ***value***. Specifies the value to assign to the variable. The value is surrounded by double-quotes and followed by a comma.
7. Repeat steps 5 and 6 for each configuration variable.
8. Click the **[Save ]**button.

**NOTE**: In a step, you can include the **config.** prefix with a variable to tell the Integration Service system to look in a configuration file to resolve the variable.

# Editing a Configuration

To edit an existing configuration:

1. Navigate to the **Configuration Registry** page (the **[Configurations]** tab).
2. Click the **[Edit]**button for the configuration you want to edit. The **Configuration Editor** pane appears:




3. On the **Configuration Editor** pane, edit the values in the following fields as needed:
   - ***Version***. Version of the configuration.
   - ***Description***. A brief description of the configuration.
   - ***Configuration Data***. Definition of each global variable. You can edit an existing definition and add definitions.
4. Click **[Save]** to save your changes.

Chapter

6

# Viewing Logs in the Integration Service

## Overview

This chapter describes the different types of logging available in the Integration Service.

This chapter covers the following topics:



6

# Logging Data in the Integration Service

The Integration Service allows you to view log data locally, remotely, or through Docker.

## Local Logging

The Integration Service writes logs to files on a host system directory. Each of the main components, such as the process manager or Celery workers, and each application that is run on the platform, generates a log file. The application log files use the application name for easy consumption and location.

In a clustered environment, the logs must be written to the same volume or disk being used for persistent storage. This ensures that all logs are gathered from all hosts in the cluster onto a single disk, and that each application log can contain information from separately located, disparate workers.

You can also implement log features such as rolling, standard out, level, and location setting, and you can configure these features with their corresponding environment variable or setting in a configuration file.

## Remote Logging

If you use your own existing logging server, such as Syslog, Splunk, or Logstash, the Integration Service can route its logs to a customer-specified location. To do so, attach your service, such as logspout, to the Microservice stack and configure your service to route all logs to the server of your choice.

> **CAUTION:** Although the Integration Service supports logging to these remote systems, ScienceLogic does not officially own or support the configuration of the remote logging locations. Use the logging to a remote system feature at your own discretion.

### Viewing Logs in Docker

You can use the Docker command line to view the logs of any current running service in the Integration Service cluster. To view the logs of any service, run the following command:

```
docker service logs -f iservices_service_name
```

Some common examples include the following:

```
docker service logs -f iservices_couchbase
```

```
docker service logs -f iservices_steprunner
```

```
docker service logs -f iservices_contentapi
```

## Logging Configuration

The following table describes the variables and configuration settings related to logging in the Integration Service:

| Environment Variable/Config Setting | Description | Default Setting |
|---|---|---|
| logdir | The director to which logs will be written. | /var/log/iservices |
| stdoutlog | Whether logs should be written to standard output (stdout). | True |
| loglevel | Log level setting for Integration Service application modules. | debug/info (varies between development and product environments) |
| celery_log_level | The log level for Celery-related components and modules. | debug/info (varies between development and product environments) |
| log_rollover_size | Size of the Integration Service logs to keep before rolling over. | 10 MB |
| log_rollover_max_files | Max number of log files to keep when rolling over. | 5 |

## Viewing Logs in the User Interface

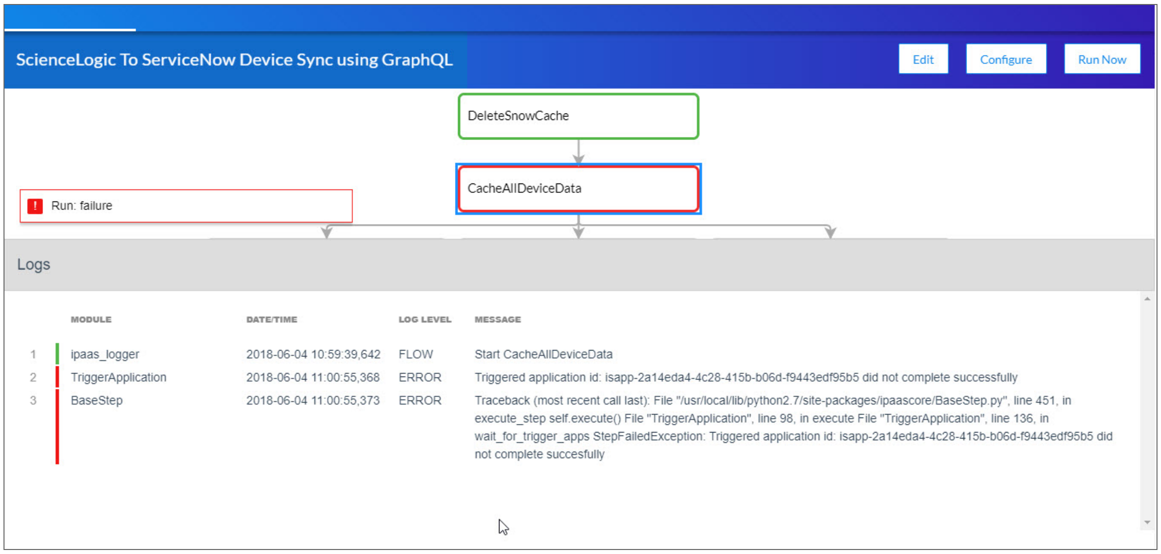
You can view logs for each step in an integration application on the **Integration Application Editor** page in the Integration Service user interface.

To view logs for the steps of an integration application:

1. From the **[Integrations]** tab, select an integration application. The **Integration Application Editor** page appears.
2. Select a step in the integration application.

6

3. Click **Logs** in the bottom left-hand corner of the screen. The **Logs** pane appears at the bottom of the page, and it displays status messages for the selected step:




**TIP:** Click the gray area of the **Logs** pane to close the pane.

# Chapter 7

# API Endpoints in the Integration Service

## Overview

The Integration Service includes an API that is available after you install the Integration Service system.

This chapter covers the following topics:



7

# Interacting with the API

To view the full documentation for the IS API:

1. From the Integration Service system, copy the file **/opt/iservices/scripts/swagger.yml** to your local computer.
2. Open a browser session and go to editor.swagger.io
3. In the Swagger Editor, open the **File** menu, select **Import File**, and import the file swagger.yml. The right pane in the Swagger Editor displays the IS API documentation.

# Available Endpoints

## POST

**/applications**. Add a new application or overwrite an existing application.

**/applications/run**. Run a single application by name.

**/configurations**. Add a new configuration or overwrite an existing configuration.

**/steps**. Add a new step or overwrite an existing step.

**/steps/run**. Run a single step by name.

**/schedule**. Add a new schedule entry.

## GET

**/applications**. Retrieve a list of all available applications.

**/applications/{appName}**. Retrieve a specific application.

**/applications/{appName}/logs**. Retrieve the logs for the specified application.

**/cache/{cache_ID}**. Retrieve a specific cache.

**/configurations**. Retrieve a list of all configurations available.

**/configurations/{configName}**. Retrieve a specific configuration.

**/reports**. Retrieve a list of all available reports.

**/reports/{appName}**. Retrieve a specific report by name.

**/reports/{reportId}**. Retrieve a specific report by ID.

**/steps**. Retrieve a list of all steps.

**/steps/{stepName}**. Retrieve a specific step.

**/schedule**. Retrieve a list of all schedule entries.

**/tasks/{taskId}**. Retrieve a specific task.

## REST

**/tasks**. Terminate all running tasks.

**/tasks/{taskId}**. Terminate a specific running task.

## DELETE

**/schedule/{scheduleName}**. Delete a schedule entry by ID.

**/reports/{appName}**. Delete a specific report by name.

**/reports/{reportId}**. Delete a specific report by its ID.

800-SCI-LOGIC (1-800-724-5644)

International: +1-703-354-1010