Integration Service Platform

Version 1.4.0

Table of Contents

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 is a platform that 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 Integration Service includes:

- ***Steps***. A step is A generic Python class that performs one 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 All-In-One Appliance includes a list of steps and metadata for those steps. Each step will execute a single action and pipe 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 SL1 and another remote system; only the arguments, such as 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 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:

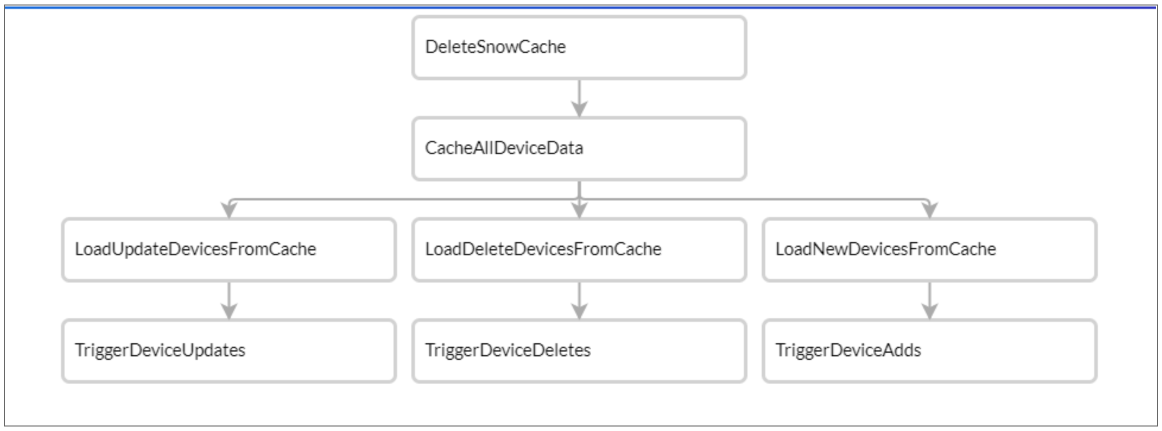
- ***Standard***. Standard steps do not require any previously collected data to perform. Standard steps are generally used to generate data to perform a transformation or insert. These steps can be run independently and concurrently.
- ***Aggregated***. Aggregated steps require data that was generated by a previously run standard 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 with varying parameters or different circumstances. 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

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:

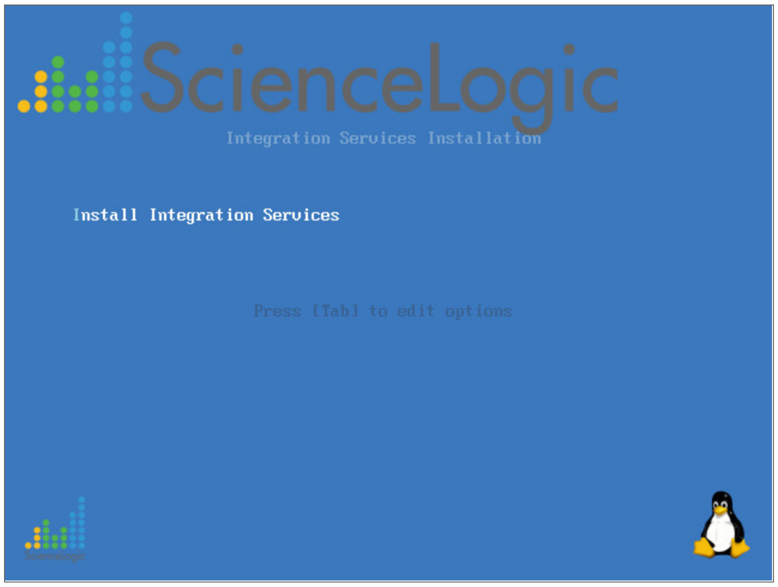
- 8 CPUs
- 12 GB total RAM
- 100 GB total storage

Installing the Integration Service

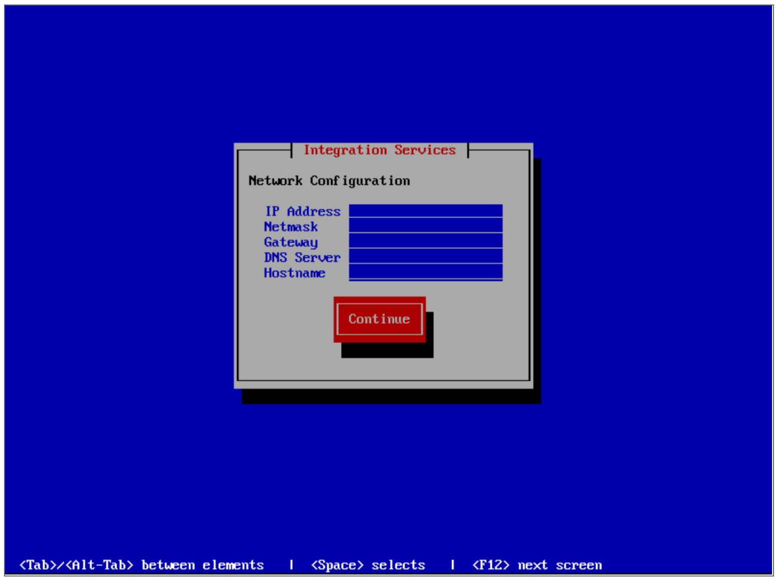
To install the Integration Service:

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

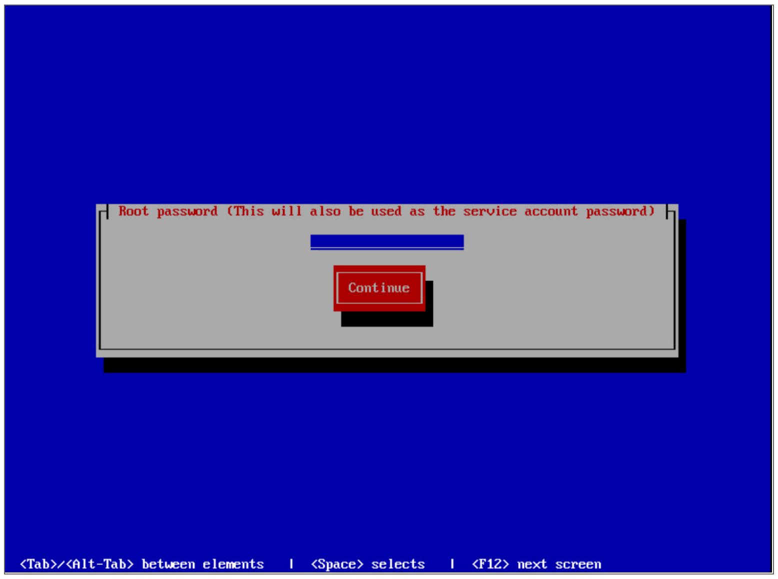
2. Using your hypervisor or bare-metal (single-tenant) server of choice, mount and boot from the Integration Service ISO. See the *System Requirements* section for prerequisites for your server. The Integration Service Installation window appears:


ScienceLogic
Integration Services Installation
Install Integration Services
Press [Tab] to edit options

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:



Root password (This will also be used as the service account password)

Continue
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen

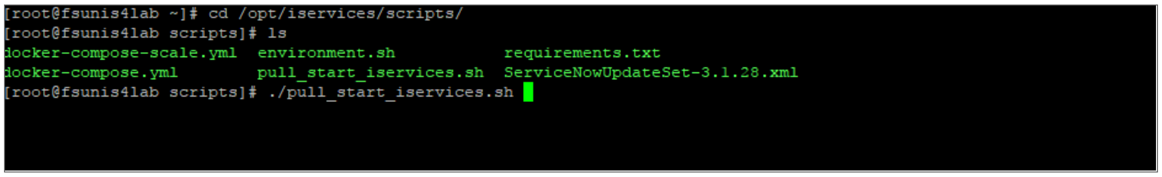
6. Type the password you want to set for the root user on the Integration Service host and press the "Enter" key.

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
```



```
[root@fsunis4lab ~]# cd /opt/iservices/scripts/
[root@fsunis4lab scripts]# ls
docker-compose-scale.yml  environment.sh           requirements.txt
docker-compose.yml        pull_start_iservices.sh  ServiceNowUpdateSet-3.1.28.xml
[root@fsunis4lab scripts]# ./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:

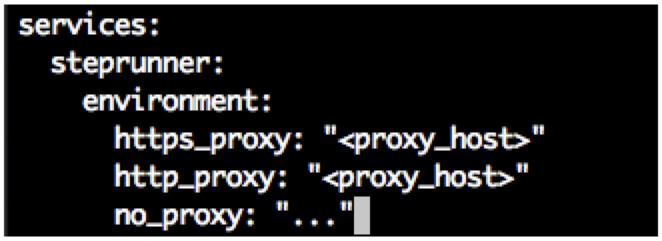
```
/opt/iservices/scripts/ispasswd
```

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.

If desired, a user can change the HTTPS certificate used by the API and user interface by mounting a new certificate to: /etc/iservices/is_cert.pem, and the key to: /etc/iservices/is_key.pem in the API and user interface containers.

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.

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

Overview

This chapter describes how to create backups for the Integration Service, including setting up Disaster Recovery and High Availability configurations to protect the data in the Integration Service.

This chapter covers the following topics:

Backing Up Integration Service Data

You can manually create a backup of your Integration Service data from one Integration Service system to another Integration Service system.

To back up your Integration Service data:

1. Manually copy the following directories from one Integration Service system to another Integration Service system:
   - /var/data/couchbase
   - /etc/iservices
2. Clean Couchbase data from both Integration Service systems by running the following command:

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

3. Copy the /etc/iservices/encryption_key and /etc/iservices/is_pass from one Integration Service system to another.

To test your backup:

1. Deploy iservices on one of the Integration Service systems.
2. Using the Integration Server user interface for that system, try one of more of the following:
   - Add one or more configurations and align those configurations to one or more integration applications.
   - Run one or more integration applications and update configuration details.
   - Set both types of schedules for one or more integration applications.
   - Update one or more integration applications and steps using iscli.
3. Copy the /var/data/couchbase directory to another Integration Service system.
4. Deploy iservices on theIntegration Service system where you pasted the Couchbase data.
5. Log in to iservices and verify that the items you changed in step 2 were recovered.

Configuring Disaster Recovery for the Integration Service

You can set up Disaster Recovery to recover and protect your Integration Service data.

Setting up Disaster Recovery

After running this setup, you should keep the Disaster Recovery node up-to-date and make sure it matches the Integration Service version on the primary node.

NOTE: If you are using the root account, the SSH key is /root/.ssh/id_rsa.

To set up Disaster Recovery:

1. Deploy an Integration Service Disaster Recovery node with the latest Integration Service ISO. The password should be the same as the primary Integration Service node.
2. Make sure that the Couchbase hostname in the /opt/iservices/scripts/docker-override.yml file is the same as the primary Integration Service node.
3. Run the compose-override.sh script on the Disaster Recovery node:

   ```
   /opt/iservices/scripts/compose-override.sh
   ```

4. Load the following Docker images on Disaster Recovery 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
   ```

5. On the primary node run ssh-keygen to generate an SSH key:

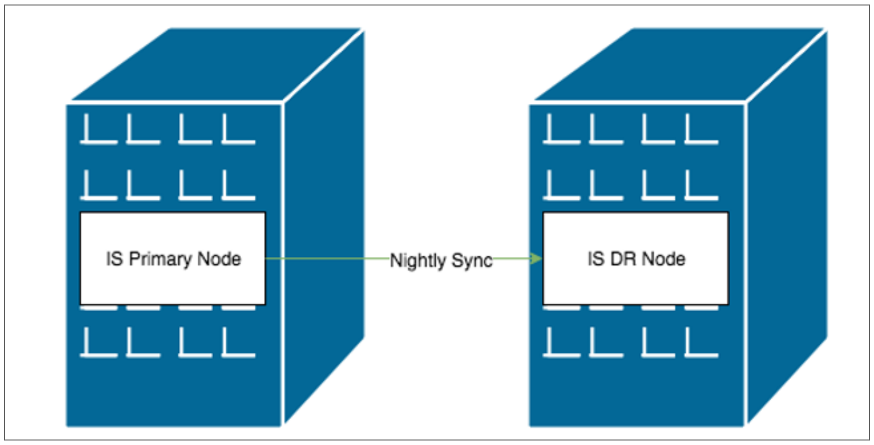
   ```
   ssh-keygen
   ```

6. Copy the SSH key to the disaster recovery node:

   ```
   ssh-copy-id <user>@<ip of DR node>
   ```

7. Create a nightly cron job on the primary Integration Service nodes to copy the Couchbase volume to the Disaster Recovery Integration Service node. The volume data is located in /var/data/couchbase/.

   ```
   0 23 * * * <user> /usr/bin/scp -i /home/<user>/.ssh/id_rsa -r /var/data/couchbase
   <user>@<host>:/var/data/couchbase
   ```



NOTE: The Disaster Recovery node for the Integration Service should not be running during the sync.

Recovering Integration Service Data

In the event of a disaster, run the following steps:

1. Run a docker stack deploy on the disaster recovery node:

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

2. Point the load balancer to the IP address of the Disaster Recovery nodes.

NOTE: If the primary Integration node is in a High Availability configuration, you need to follow the failover steps for the Couchbase database covered in its High Availability documentation when you bring up Integration Service on the Disaster Recovery node before Integration Service will be available.

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 manual intervention will be required.

Docker Swarm Requirements for High Availability

Implementing Docker Swarm High Availability ensures that 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:

- At least three nodes running as managers exist in the cluster. Three nodes are required to ensure that there can always 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?

Only using two nodes goes against the Docker 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 odes 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

Implementing 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 a cluster to be run on a minimum of *three* nodes.

Each node will have an independent and persistent storage volume, which is constantly replicated throughout the cluster. Alternatively, shared storage can be used instead of independent persistent volumes. This 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 in 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 before the database is operational again

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

In this situation, no data will be lost, because the data is replicated everywhere. If multiple Couchbase data nodes go down at the same time, automatic failover may not happen, not even nodes for quorum to fail over.

You will then need to perform manual failover steps. After you perform these manual actions, the Integration Service will be back up and running.

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

NOTE: With less than three nodes, you need 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 backed up 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***.

To ensure that all nodes have joined, you can expose the management user interface port (15672) of the Rabbit services when configuring clusters. The user interface should only be enabled temporary as it might pose a security risk. The default user interface login is *guest/guest*.

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

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

Potential Risks and Mitigations

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

You can prevent this situation by using your choice of network shared storage, so that if the queue fails on one node, the queu can be brought back up on another node and continue to have the same persisted queue data.

Requirements/Setup (Enabled by Default)

- You must statically set the RabbitMQ hostname 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

Using this configuration lets multiple nodes join a RabbitMQ cluster. Clustering multiple nodes means that 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.

Due to the disk-persisted queues, even if all nodes in the Rabbit cluster fail, or if the service is removed entirely, then no data loss should occur. Restarting these nodes should resume with the same cluster setup with the previously saved data.

When this setup is configured, the Integration Service automation automatically clusters multiple RabbitMQ services with the HA policy of all-node replication, and with retroactive queue synchronization disabled. For more information, refer to the RabbitMQ documentation.

Potential Risks and Mitigations

If a docker swarm cluster is created with only two nodes, the cluster might stop functioning if a single node fails. To prevent this situation, whenever you set up a cluster configuration, you should use at least *three* nodes.

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.
- A new RabbitMQ service must be added into your 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 your docker-compose.yml file is updated, the nodes should 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

At times you might need to check on the status of your RabbitMQ cluster. This section contains commands and additional resources to help you when you are administering your clusters. You can run all the commands by running `docker exec` on any RabbitMQ container.

Helpful Commands

Use the following commands to check the status of your clustered RabbitMQ environment:

- `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.

More Information about RabbitMQ

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

Configuring Clustering and High Availability

This topic 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, but the use of the Integration Service ISO is not required.

System Requirements for High Availability

On each Docker Swarm node, 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 and High Availability:

1. If you do not already have Integration Service running in your environment, follow the steps to bring up Integration Service on a single node. This creates a single-node Docker Swarm manager.
2. Run the following command on the Docker Swarm manager to get the join token that is required 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 warm 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
```

Configuring the Couchbase Database

NOTE: If you are adding a couchbase-worker to an existing Integration Service system, complete the following steps. If you are *not* adding a couchbase-worker, you can skip the entire "Configuring the Couchbase Database" section.

To configure the Couchbase Database for High Availability:

1. Add a new alias to the couchbase node in the docker-override.yml file:

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

2. Run the following Docker commands:

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

3. Log in to the Couchbase Docker container and run the following command to set the hostname on the container:

```
curl -v -X POST -u isadmin:<password> http://127.0.0.1:8091/node/controller/rename
-d hostname=couchbase.isnet
```

To add the 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'
```

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

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

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
```

Code Example: docker-compose-override.yml

The following section includes a complete example of a docker-compose-override.yml file:

```
version: '3.2'
services:
  steprunner:
    environment:
      db_host: couchbase,couchbase-worker

  scheduler:
    environment:
      db_host: couchbase,couchbase-worker

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

  couchbase-worker:
    image: sciencelogic/is-couchbase:1.2.0
    container_name: couchbase-worker
    volumes:
      - "/var/data/couchbase:/opt/couchbase/var"
    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
    ulimits:
      nofile: 80000
      core: 100000000
      memlock: 100000000
    environment:
      TYPE: 'WORKER'
      AUTO_REBALANCE: 'true'
      db_host: 'couchbase'
    depends_on:
      - couchbase

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

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.

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>
   ```

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.

Known Issues

The Docker Network Alias is incorrect

If the Docker Network Alias IP is incorrect, you might experience issues with the iservices_contentapi container.

To address this issue, run the following Docker commands:

```
docker service scale iservices_contentapi=0
docker service scale iservices_contentapi=1 (or the number want)
```

The Couchbase service is not starting, and it is unable to continue after nc -z localhost

To resolve this issue, stop the container where this is happening and remove its persistent volume:

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

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.

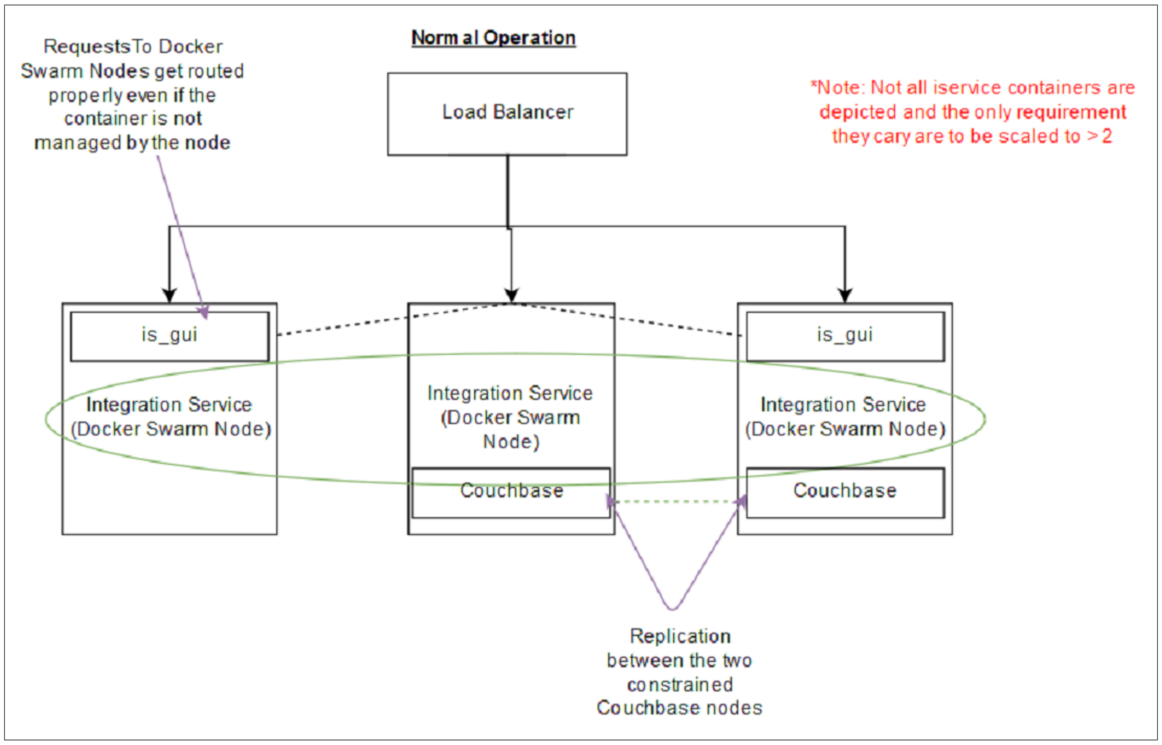
The is_gui fails to start after a manual failover of the swarm node:

To address this issue, run the following Docker commands:

```
docker stack rm iservices
systemctl restart docker
docker stack deploy -c docker-compose.yml iservices
```

Additional Configuration Information

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


Normal Operation
Requests To Docker Swarm Nodes get routed properly even if the container is not managed by the node
Load Balancer
*Note: Not all iservice containers are depicted and the only requirement they cary are to be scaled to > 2
is_gui
Integration Service (Docker Swarm Node)
Integration Service (Docker Swarm Node)
Couchbase
is_gui
Integration Service (Docker Swarm Node)
Couchbase
Replication between the two constrained Couchbase nodes

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
```

Chapter

4

Managing Integration Applications

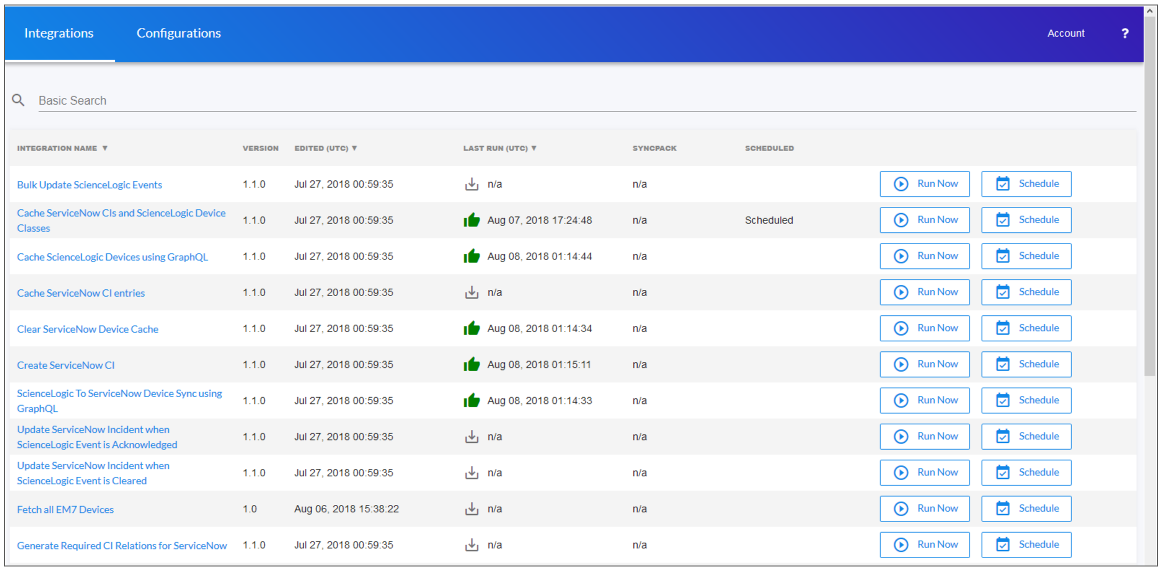
Overview

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:



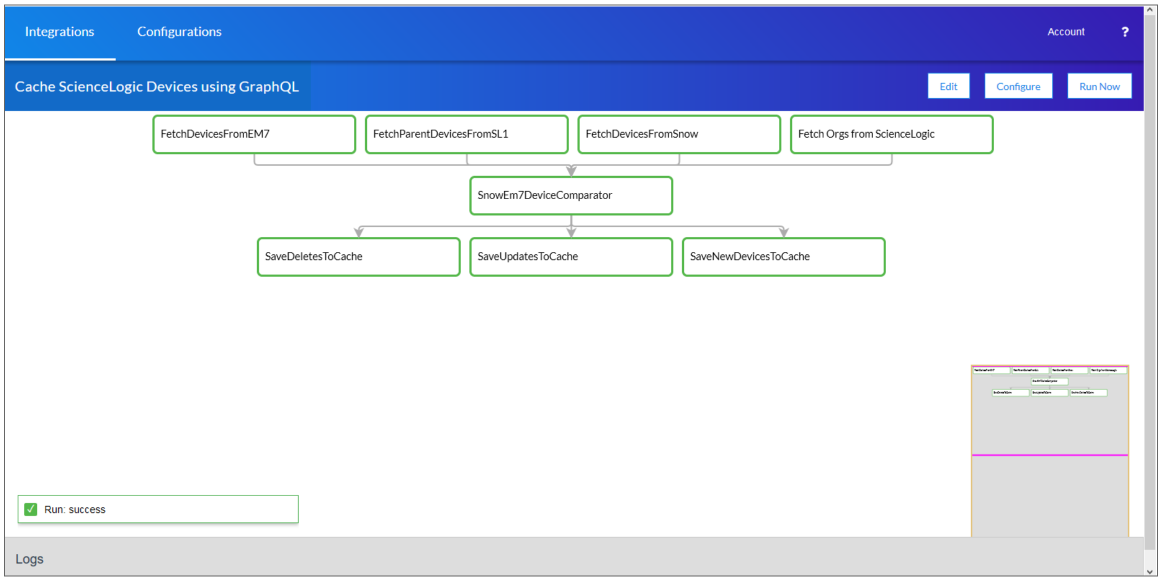
INTEGRATION NAME ▼	VERSION	EDITED (UTC) ▼	LAST RUN (UTC) ▼	SYNCPACK	SCHEDULED		
Bulk Update ScienceLogic Events	1.1.0	Jul 27, 2018 00:59:35	n/a	n/a		Run Now	Schedule
Cache ServiceNow CIs and ScienceLogic Device Classes	1.1.0	Jul 27, 2018 00:59:35	Aug 07, 2018 17:24:48	n/a	Scheduled	Run Now	Schedule
Cache ScienceLogic Devices using GraphQL	1.1.0	Jul 27, 2018 00:59:35	Aug 08, 2018 01:14:44	n/a		Run Now	Schedule
Cache ServiceNow CI entries	1.1.0	Jul 27, 2018 00:59:35	n/a	n/a		Run Now	Schedule
Clear ServiceNow Device Cache	1.1.0	Jul 27, 2018 00:59:35	Aug 08, 2018 01:14:34	n/a		Run Now	Schedule
Create ServiceNow CI	1.1.0	Jul 27, 2018 00:59:35	Aug 08, 2018 01:15:11	n/a		Run Now	Schedule
ScienceLogic To ServiceNow Device Sync using GraphQL	1.1.0	Jul 27, 2018 00:59:35	Aug 08, 2018 01:14:33	n/a		Run Now	Schedule
Update ServiceNow Incident when ScienceLogic Event is Acknowledged	1.1.0	Jul 27, 2018 00:59:35	n/a	n/a		Run Now	Schedule
Update ServiceNow Incident when ScienceLogic Event is Cleared	1.1.0	Jul 27, 2018 00:59:35	n/a	n/a		Run Now	Schedule
Fetch all EM7 Devices	1.0	Aug 06, 2018 15:38:22	n/a	n/a		Run Now	Schedule
Generate Required CI Relations for ServiceNow	1.1.0	Jul 27, 2018 00:59:35	n/a	n/a		Run Now	Schedule

For each integration application, the page displays:

- ***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 (UTC)***. The last time the integration application was edited.
- ***Last Run (UTC)***. The last time the integration application was run.
- ***Next Run (UTC)***. The next time the integration application is scheduled to 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.

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 they will run when the application is executed.

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: success". Below the status is the **Logs** pane, which displays any 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.

Using the buttons on the **Integration Application Editor** page, you can edit, configure, and run the integration application.

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`*

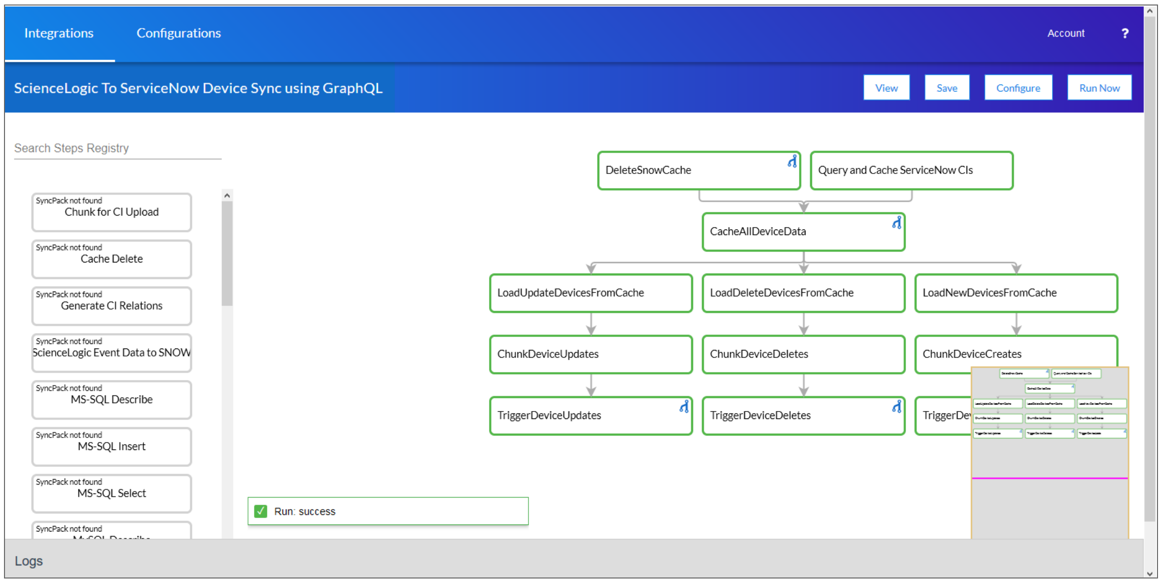
   where:

   - *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.

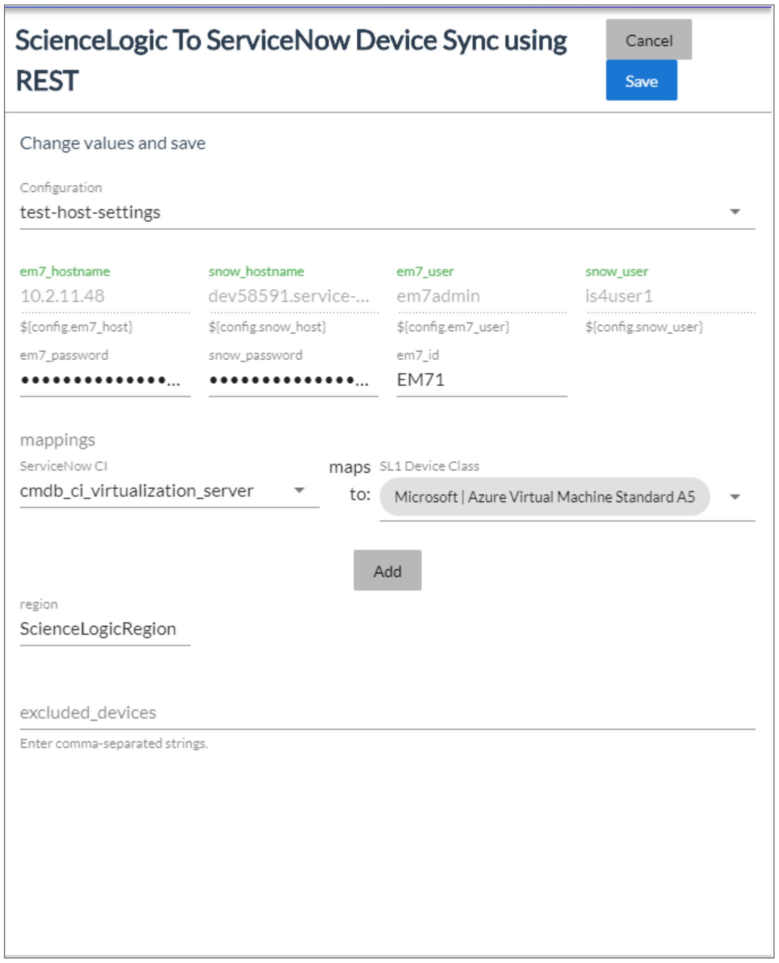
3. Click the step you want to add, drag it to the main pane of the an **Integration Application Editor** page, and drop it into the integration application .
4. To adjust the position of any step in the integration application, click the step you want to move and drag it to its new location.
5. To redirect 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

To configure an integration application:

1. From the **Integration Application Editor** page, click the **[Configure]** button. The **Configuration** pane opens on the right side of the window, displaying credential information.


ScienceLogic To ServiceNow Device Sync using REST

Change values and save

Configuration: test-host-settings

em7_hostname: 10.2.11.48 — ${config.em7_host}
snow_hostname: dev58591.service-... — ${config.snow_host}
em7_user: em7admin — ${config.em7_user}
snow_user: is4user1 — ${config.snow_user}
em7_password: ••••••••••••••...
snow_password: ••••••••••••••...
em7_id: EM71

mappings
ServiceNow CI: cmdb_ci_virtualization_server maps to: SL1 Device Class: Microsoft | Azure Virtual Machine Standard A5

region: ScienceLogicRegion

excluded_devices
Enter comma-separated strings.

2. In the **Configuration** pane, edit the values in the application. The fields are different for each 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.

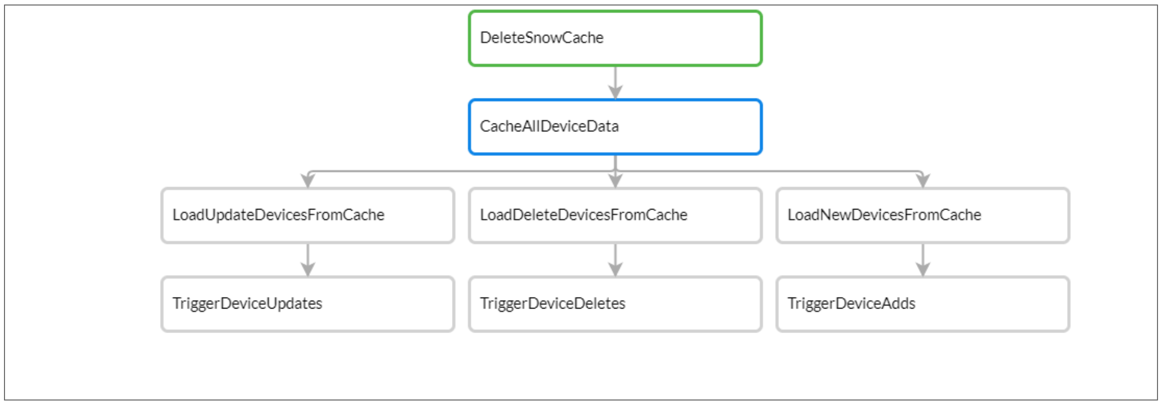
3. When you are finished, click **[Save]**.

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.

Running or Stopping an Integration Application

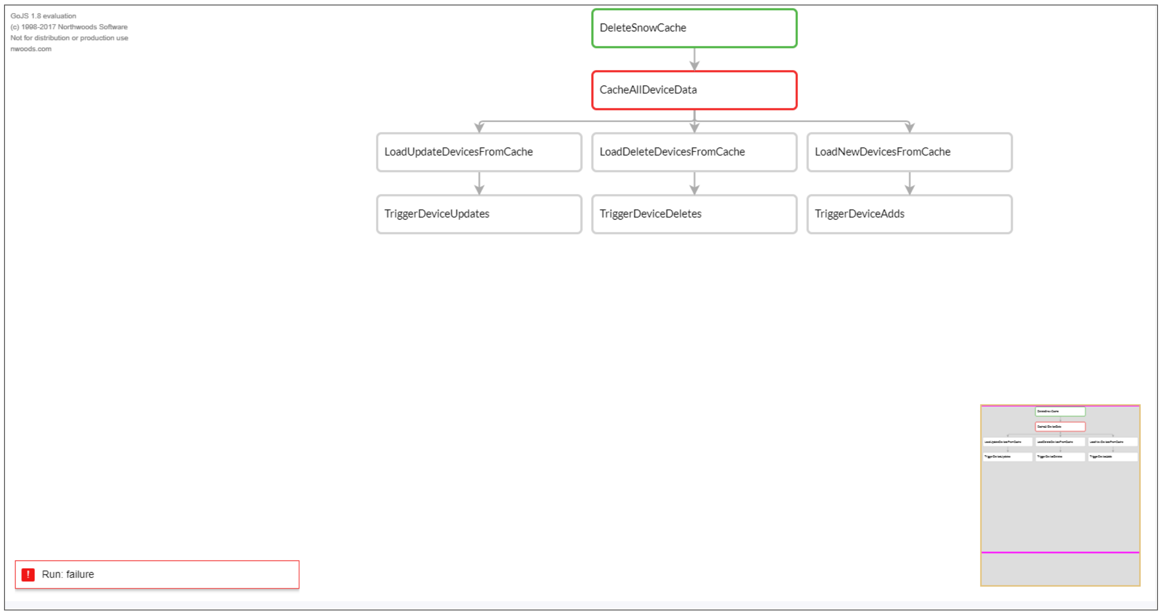
To run an integration application:

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:



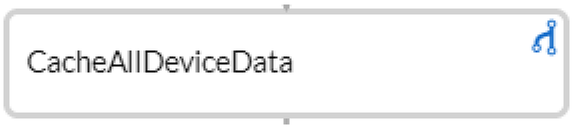
Step Color	State
Blue	Running
Green	Successful
Red	Failed

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.

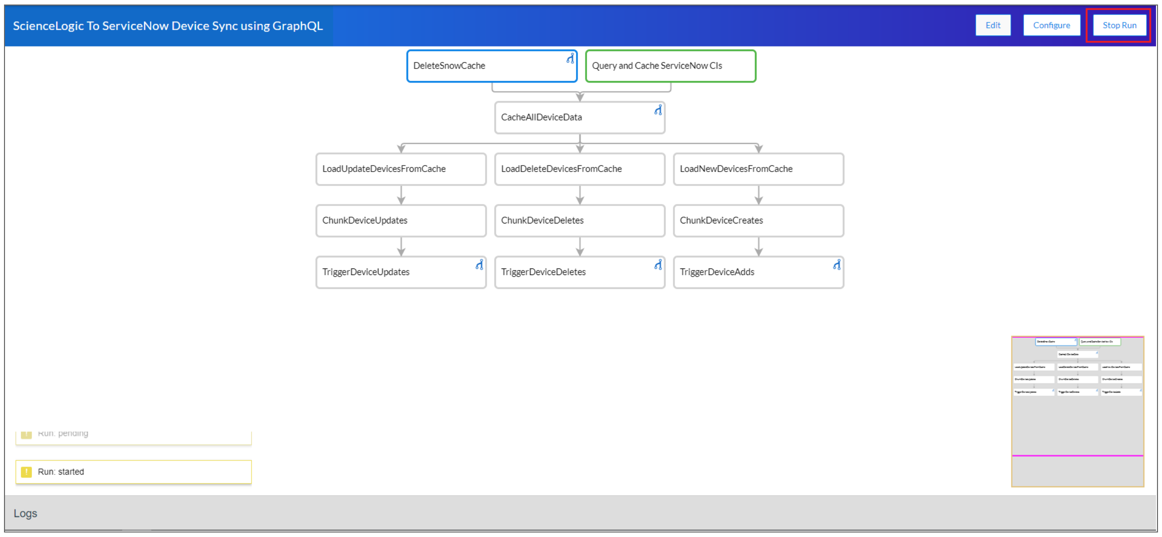


Run: failure

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.

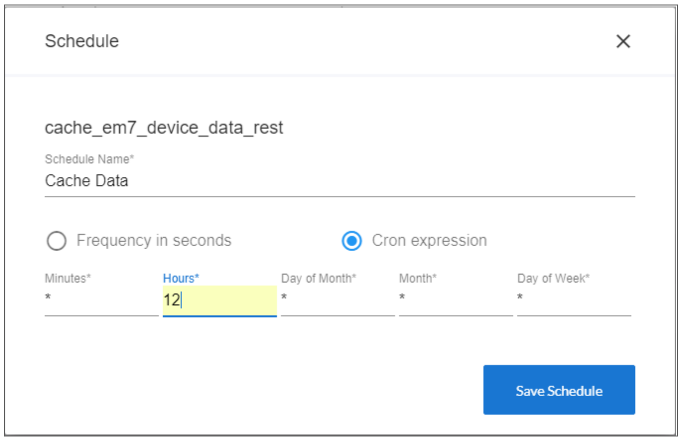


4

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:



cache_em7_device_data_rest

Schedule Name*
Cache Data

Frequency in seconds [✓ Cron expression]

Minutes*	Hours*	Day of Month*	Month*	Day of Week*
*	12	*	*	*

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

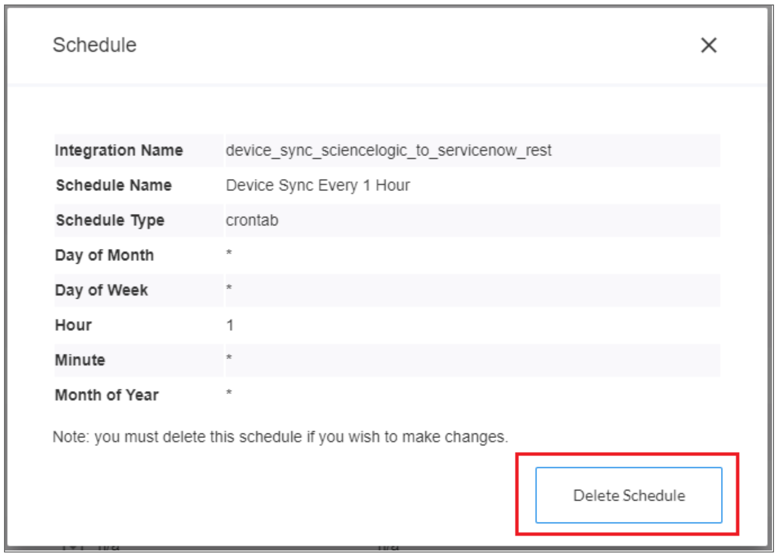
   - ***Schedule Name***. Type a name for your schedule.
   - ***Frequency in seconds***. Schedule the integration to run in an interval of seconds.
   - ***Cron expression***. Schedule the integration using a cron expression.

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.



Schedule

Integration Name	device_sync_sciencelogic_to_servicenow_rest
Schedule Name	Device Sync Every 1 Hour
Schedule Type	crontab
Day of Month	*
Day of Week	*
Hour	1
Minute	*
Month of Year	*

Note: you must delete this schedule if you wish to make changes.

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

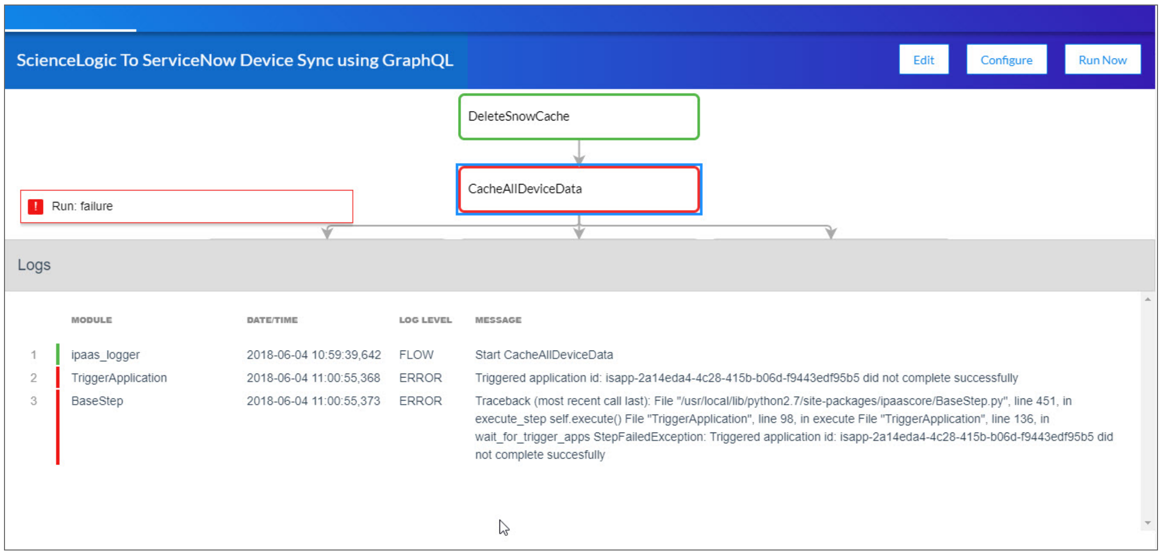
Viewing Logs for an Integration Application

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:



	MODULE	DATE/TIME	LOG LEVEL	MESSAGE
1	ipaas_logger	2018-06-04 10:59:39,642	FLOW	Start CacheAllDeviceData
2	TriggerApplication	2018-06-04 11:00:55,368	ERROR	Triggered application id: isapp-2a14eda4-4c28-415b-b06d-f9443edf95b5 did not complete successfully
3	BaseStep	2018-06-04 11:00:55,373	ERROR	Traceback (most recent call last): File "/usr/local/lib/python2.7/site-packages/ipaascore/BaseStep.py", line 451, in execute_step self.execute() File "TriggerApplication", line 98, in execute File "TriggerApplication", line 136, in wait_for_trigger_apps StepFailedException: Triggered application id: isapp-2a14eda4-4c28-415b-b06d-f9443edf95b5 did not complete succesfully

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.

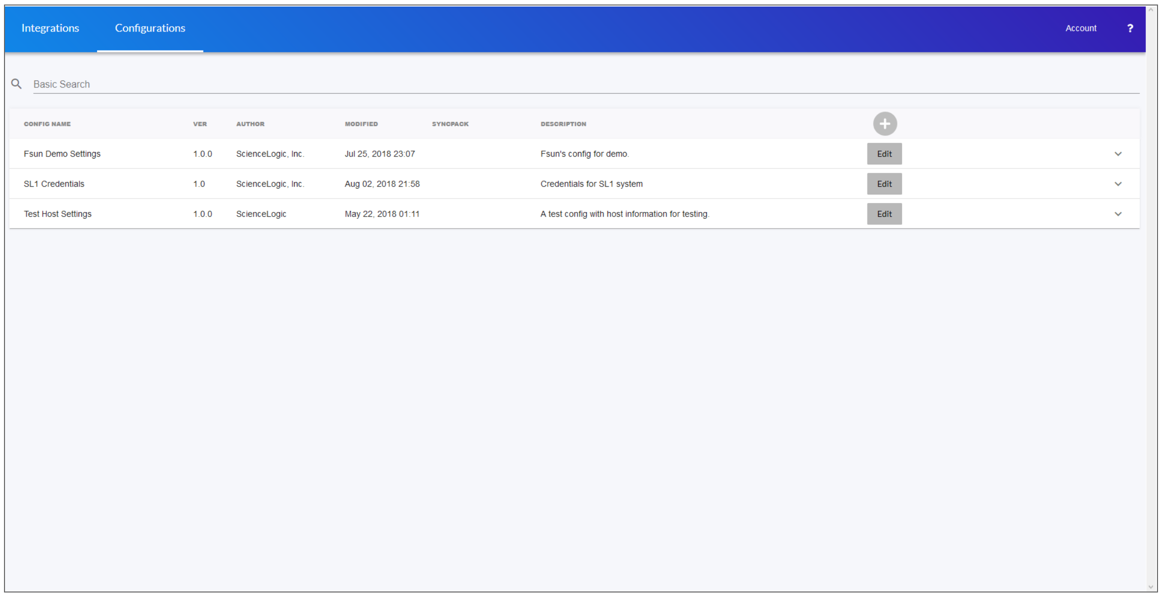
This chapter covers the following topics:

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.

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:



CONFIG NAME	VER	AUTHOR	MODIFIED	SYNCPACK	DESCRIPTION	
Fsun Demo Settings	1.0.0	ScienceLogic, Inc.	Jul 25, 2018 23:07		Fsun's config for demo.	Edit
SL1 Credentials	1.0	ScienceLogic, Inc.	Aug 02, 2018 21:58		Credentials for SL1 system	Edit
Test Host Settings	1.0.0	ScienceLogic	May 22, 2018 01:11		A test config with host information for testing.	Edit

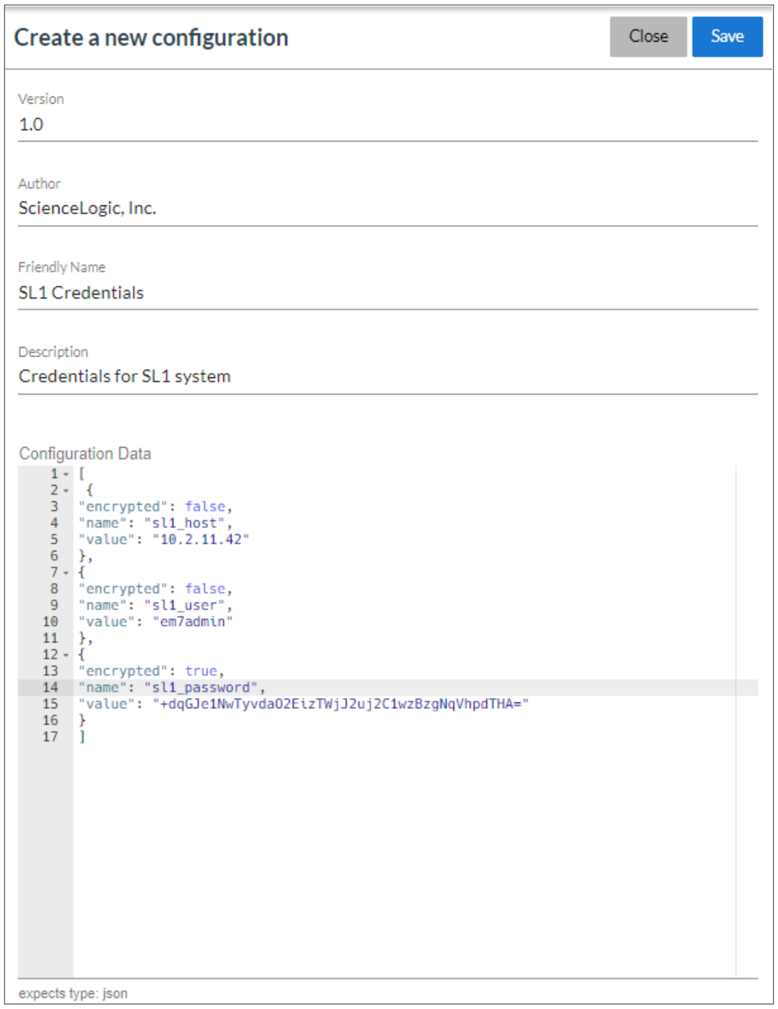
For each configuration, the page displays:

- ***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:



Version
1.0

Author
ScienceLogic, Inc.

Friendly Name
SL1 Credentials

Description
Credentials for SL1 system

Configuration Data

```
[
 {
"encrypted": false,
"name": "sl1_host",
"value": "10.2.11.42"
},
{
"encrypted": false,
"name": "sl1_user",
"value": "em7admin"
},
{
"encrypted": true,
"name": "sl1_password",
"value": "+dqGJe1NwTyvdaO2EizTWjJ2uj2C1wzBzgNqVhpdTHA="
}
]
```

expects type: json

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.

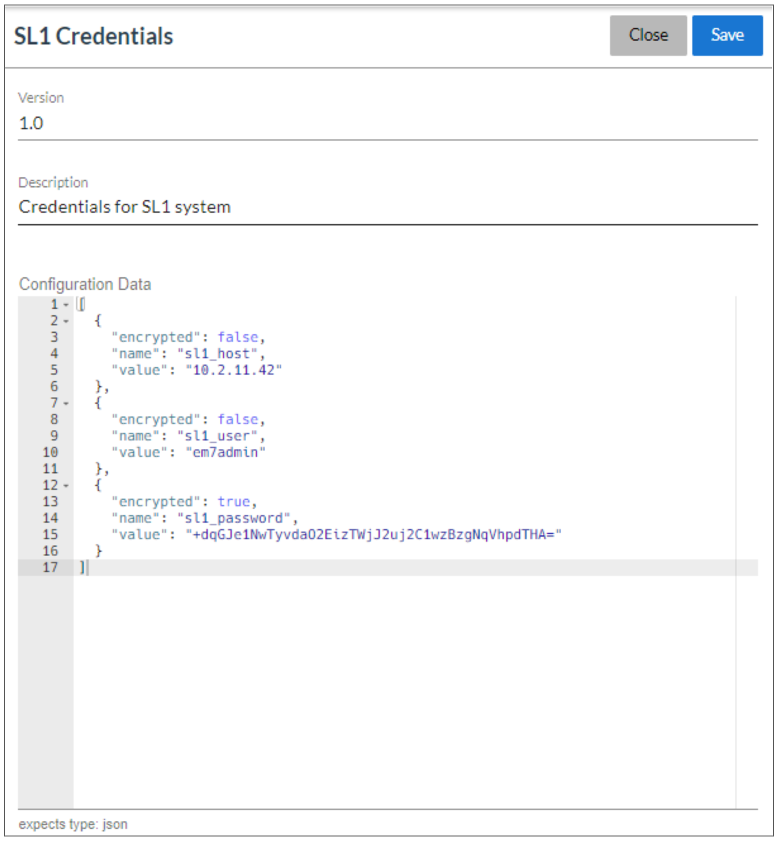
7. Repeat steps 5 and 6 for each configuration variable.
8. Click the **[Save]**button.

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:



SL1 Credentials

Version
1.0

Description
Credentials for SL1 system

Configuration Data

```
[
  {
    "encrypted": false,
    "name": "sl1_host",
    "value": "10.2.11.42"
  },
  {
    "encrypted": false,
    "name": "sl1_user",
    "value": "em7admin"
  },
  {
    "encrypted": true,
    "name": "sl1_password",
    "value": "+dqGJe1NwTyvdaO2EizTWjJ2uj2C1wzBzgNqVhpdTHA="
  }
]
```

expects type: json

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:

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

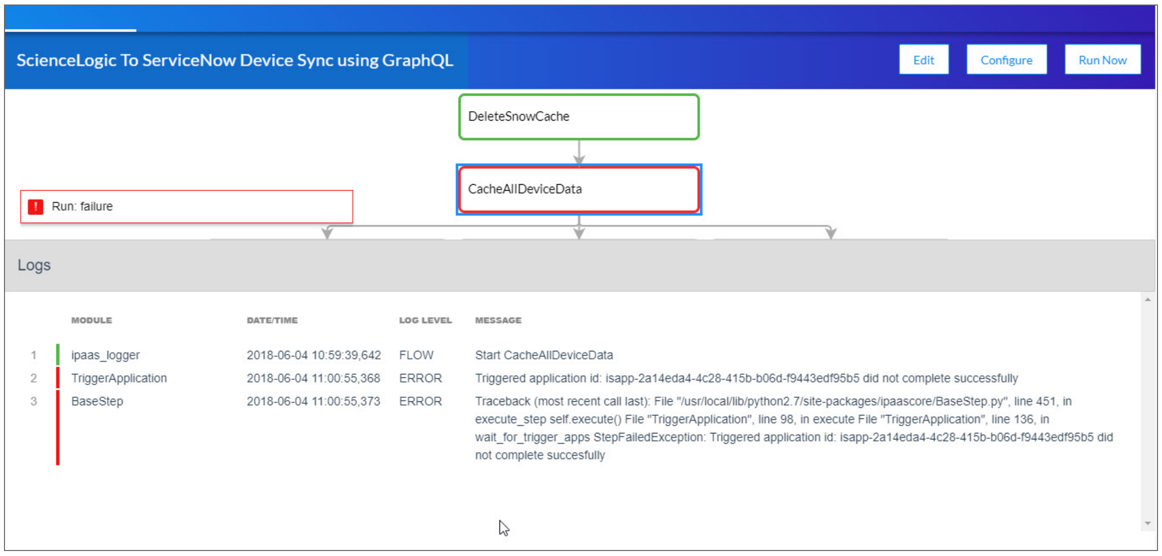
Viewing Logs in the User Interface

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.

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:



	MODULE	DATE/TIME	LOG LEVEL	MESSAGE
1	ipaas_logger	2018-06-04 10:59:39,642	FLOW	Start CacheAllDeviceData
2	TriggerApplication	2018-06-04 11:00:55,368	ERROR	Triggered application id: isapp-2a14eda4-4c28-415b-b06d-f9443edf95b5 did not complete successfully
3	BaseStep	2018-06-04 11:00:55,373	ERROR	Traceback (most recent call last): File "/usr/local/lib/python2.7/site-packages/ipaascore/BaseStep.py", line 451, in execute_step self.execute() File "TriggerApplication", line 98, in execute File "TriggerApplication", line 136, in wait_for_trigger_apps StepFailedException: Triggered application id: isapp-2a14eda4-4c28-415b-b06d-f9443edf95b5 did not complete succesfully

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:

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.

ScienceLogic

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

International: +1-703-354-1010