



Integration Service Platform

Version 1.8.3

Table of Contents

Introduction to the Integration Service	7
What is the Integration Service?	8
What is a Step?	9
What is an Integration Application?	10
What is a Configuration?	11
Creating and Saving Integration Service Components	11
Installing and Configuring the Integration Service	12
Integration Service Architecture	14
Prerequisites for the Integration Service	15
System Requirements	15
Installing the Integration Service	16
Installing the Integration Service via ISO	16
Installing the Integration Service via RPM	19
Configuring the Integration Service	21
Setting a Hard Memory Limit in Docker	21
Setting a Soft Memory Limit in the Worker Environment	21
Upgrading the Integration Service	22
Changing the Integration Service Password	23
Configuring a Proxy Server	23
Configuring Security Settings	24
Changing the HTTPS Certificate	24
Using Password and Encryption Key Security	24
Monitoring the Integration Service	25
PowerPacks for Monitoring Elements of the Integration Service	25
Monitoring the Integration Service with SL1	26
Setting Up Discovery	26
Monitoring the RabbitMQ Service	29
Monitoring the Couchbase Service	32
Integration Service System Diagnostics	36
Integration Service Endpoints	37
Flower API	37
Couchbase API	38
Docker Statistics	38
Integration Service Log Files	39
Accessing Docker Log Files	39
Accessing Local File System Logs	39
Understanding the Contents of Log Files	40
Configuring the Integration Service for High Availability	41
System Requirements for High Availability	43
Configuring a High Availability Environment for the Integration Service	43
Docker Swarm Requirements for High Availability	44
Docker Swarm Frequently Asked Questions for High Availability	44
Couchbase Database Requirements for High Availability	45
Couchbase Database Frequently Asked Questions for High Availability	45
RabbitMQ Clustering and Persistence for High Availability	46
RabbitMQ Option 1: Persisting Queue to Disk on a Single Node (Default Configuration)	46
RabbitMQ Option 2: Clustering Nodes with Persistent Queues on Each Node	47
Checking the Status of a RabbitMQ Cluster	48
Configuring Clustering and High Availability	49
Configuring Docker Swarm	49

Configuring the Couchbase Database	49
Troubleshooting Issues with Couchbase Indexes	52
Code Example: docker-compose-override.yml	53
Scale iservices-contentapi	55
Manual Failover	55
Additional Configuration Information	58
Exposing Additional Couchbase Cluster Node Management Interfaces over TLS	58
HAProxy Configuration (Optional)	59
Known Issues	60
Docker Network Alias is incorrect	60
Docker container on last swarm node cannot communicate with other swarm nodes	60
Couchbase service does not start, remains hung at nc -z localhost	60
Couchbase-worker fails to connect to master	60
The Integration Service user interface fails to start after a manual failover of the swarm node	61
The Integration Service user interface returns 504 errors	61
NTP should be used, and all node times should be in sync	61
Example Logs from Flower	61
Managing Integration Applications	62
Viewing the List of Integration Applications	63
Editing an Integration Application	64
Default Steps	65
Working with an Integration Application	66
Configuring an Integration Application	67
Running or Stopping an Integration Application	68
Viewing Previous Runs of an Integration Application	69
Scheduling an Integration Application	72
Viewing a Report for an Integration Application	76
Viewing System Diagnostics	78
Backing up Data for Disaster Recovery	79
Creating a Backup	80
Restoring a Backup	83
Managing Configurations	85
What is a Configuration?	86
Creating a Configuration	88
Editing a Configuration	90
Viewing Logs in the Integration Service	91
Logging Data in the Integration Service	92
Local Logging	92
Remote Logging	92
Viewing Logs in Docker	92
Logging Configuration	93
Viewing the Step Logs for an Integration Application	93
Removing Logs on a Regular Schedule	94
API Endpoints in the Integration Service	96
Interacting with the API	96
Available Endpoints	97
POST	97
Querying for the State of an Integration Application	97
GET	97
REST	98
DELETE	98
Troubleshooting the Integration Service	99

Resources for Troubleshooting	101
Useful Integration Service Ports	101
Helpful Docker Commands	101
Viewing Container Versions and Status	101
Restarting a Service	101
Stopping all Integration Service Services	102
Restarting Docker	102
Viewing Logs for a Specific Service	102
Clearing RabbitMQ Volume	102
Viewing the Process Status of All Services	103
Deploying Services from a Defined Docker Compose File	103
Dynamically Scaling for More Workers	103
Completely Removing Services from Running	103
Helpful Couchbase Commands	103
Checking the Couchbase Cache to Ensure an SL1 Device ID is Linked to a ServiceNow Sys ID	103
Clearing the Internal Integration Service Cache	104
Accessing Couchbase with the Command-line Interface	105
Useful API Commands	105
Getting Integrations from the Integration Service API	105
Creating and Retrieving Schedules with the Integration Service API	105
Diagnosis Tools	106
Identifying Why a Service or Container Failed	107
Step 1: Obtain the ID of the failed container for the service	107
Step 2: Check for any error messages or logs indicating an error	107
Step 3: Check for out of memory events	108
Troubleshooting a Cloud Deployment of the Integration Service	108
Identifying Why an Integration Application Failed	109
Determining Where an Integration Application Failed	109
Retrieving Additional Debug Information (Debug Mode)	109
General Troubleshooting Steps	110
Integration Service	111
ServiceNow	111
Frequently Asked Questions	111
What is the first thing I should do when I have an issue with my Integration Service?	111
Why do I get a "Connection refused" error when trying to communicate with Couchbase?	111
How do I remove a schedule that does not have a name?	112
Why are there client-side timeouts when communicating with Couchbase?	113
What causes a Task Soft Timeout?	113
Why are incident numbers not populated in SL1 on Incident creation in ServiceNow?	113
Why am I not getting any Incidents after disabling the firewall?	114
Why are Incidents not getting created in ServiceNow?	114
How can I point the "latest" container to my latest available images for the Integration Service?	114
How do I manually create a backup of my Integration Service system, and how do I manually restore?	114
What do I do if I get a Code 500 error when I try to access the Integration Service user interface?	115
What are some common examples of using the iscli tool?	116
How do I view a specific run of an application on IS?	116
Using SL1 to Monitor the Integration Service	117
What Does the Integration ServicePowerPack Monitor?	118
Monitoring Integration Applications in the Integration Service	118
Installing the ScienceLogic: Integration ServicePowerPack	120
Creating a SOAP/XML Credential for the Integration Service	121
Creating a Virtual Device for the Integration Service PowerPack	122

Aligning the Integration Service PowerPack Dynamic Applications	123
Integration Service for Multi-tenant Hosted Environments	124
Quick Start Checklist for Deployment	125
Deployment	125
Core Service Nodes	125
Requirements	126
Configuring Core Service Nodes	126
Critical Elements to Monitor on Core Nodes	126
Worker Service Nodes	126
Requirements	127
Event Sync Throughput Node Sizing	127
Test Environment and Scenario	127
Configuring the Worker Node	127
Initial Worker Node Deployment Settings	128
Worker Failover Considerations and Additional Sizing	128
Knowing When More Resources are Necessary for a Worker	128
Keeping a Worker Node on Standby for Excess Load Distribution	128
Critical Elements to Monitor in a Steprunner	129
Onboarding a Customer	129
Create the Configuration	129
Label the Worker Node Specific to the Customer	129
Creating a Node Label	130
Placing a Service on a Labeled Node	130
Dedicating Queues Per Integration or Customer	130
Add Workers for the New Queues	130
Create Integration Schedules and Automation Settings to Utilize Separate Queues	132
Scheduling an Integration with a Specific Queue and Configuration	133
Configuring Automations to Utilize a Specific Queue and Configuration	133
Failure Scenarios	133
Worker Containers	133
API	134
Couchbase	135
RabbitMQ	136
Integration Service User Interface	136
Redis	137
Known Issue for Groups of Containers	137
Examples and Reference	138
Example of an Integration Service Configuration Object	138
Example of a Schedule Configuration	140
Test Cases	144
Load Throughput Test Cases	144
Failure Test Cases	144
Backup Considerations	145
What to Back Up	145
Fall Back and Restore to a Disaster Recovery (Passive) System	145
Resiliency Considerations	146
The RabbitMQ Split-brain Handling Strategy (SL1 Default Set to Autoheal)	146
ScienceLogic Policy Recommendation	147
Changing the RabbitMQ Default Split-brain Handling Policy	147
Using Drained Managers to Maintain Swarm Health	147
Updating the Integration Service Cluster with Little to No Downtime	148
Updating Offline (No Connection to a Docker Registry)	148

Updating Online (All Nodes Have a Connection to a Docker Registry)	148
Additional Sizing Considerations	148
Sizing for Couchbase Services	148
Sizing for RabbitMQ Services	148
Sizing for Redis Services	149
Sizing for contentapi Services	149
Sizing for the GUI Service	149
Sizing for Workers: Scheduler, Steprunner, Flower	149
Node Placement Considerations	149
Preventing a Known Issue: Place contentapi and Redis services in the Same Physical Location	149
Common Problems, Symptoms, and Solutions	150
Common Resolution Explanations	158
Elect a New Swarm Leader	158
Recreate RabbitMQ Queues and Exchanges	158
Resynchronize RabbitMQ Queues	159
Identify the Cause of a Service not Deploying	159
Repair Couchbase Indexes	160
Add a Broken Couchbase Node Back into the Cluster	161
Restore Couchbase Manually	161
Integration Service Multi-tenant Upgrade Process	162
Perform Environment Checks Before Upgrading	162
Prepare the Systems	162
Perform the Upgrade	164
Upgrade Core Services (Rabbit and Couchbase)	165
Update the GUI	168
Update Workers and contentapi	168

Introduction to the Integration Service

Overview

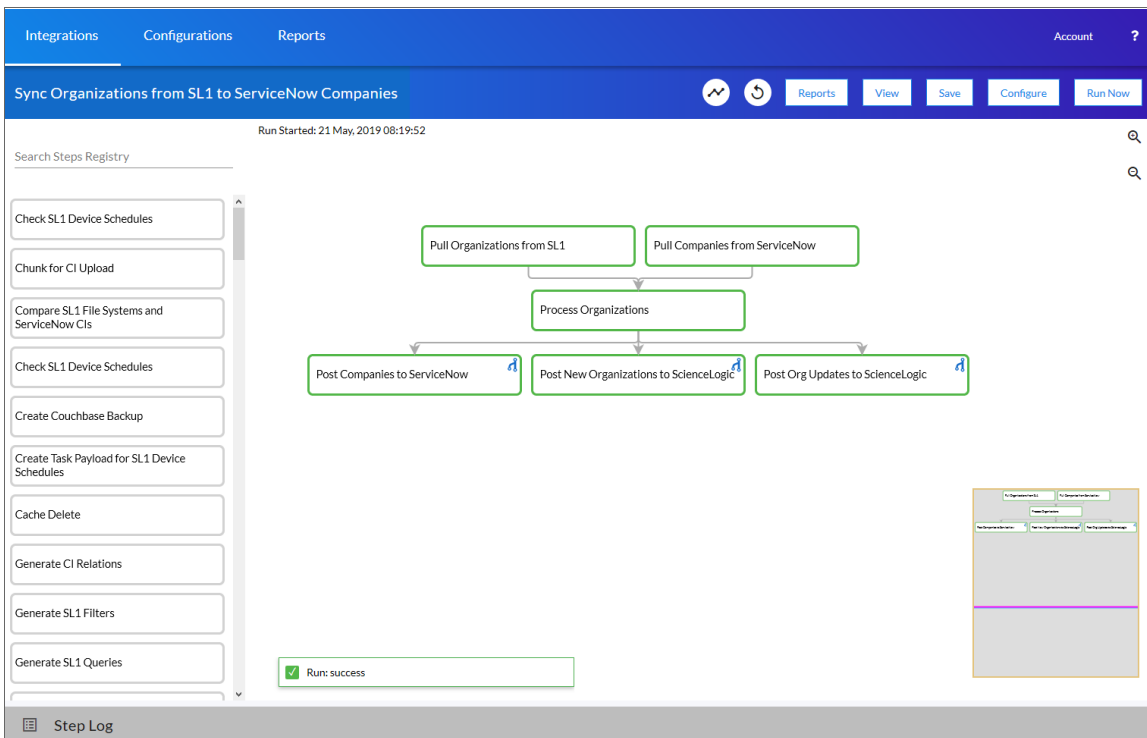
The Integration Service provides a generic platform for integrations between SL1 and third-party platforms.

This chapter covers the following topics:

<i>What is the Integration Service?</i>	8
<i>What is a Step?</i>	9
<i>What is an Integration Application?</i>	10
<i>What is a Configuration?</i>	11
<i>Creating and Saving Integration Service Components</i>	11

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.

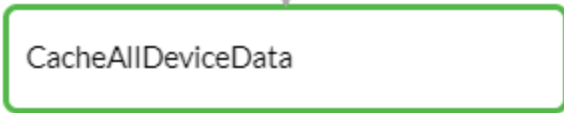


The key elements of the Integration Service user interface include the following:

- **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 object 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. Click the **[Configure]** button from an integration application in the Integration Service user interface to access the configuration object for that integration application.

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 you if any required parameters are missing before the Integration Service runs the step.

Steps are grouped into the following types:

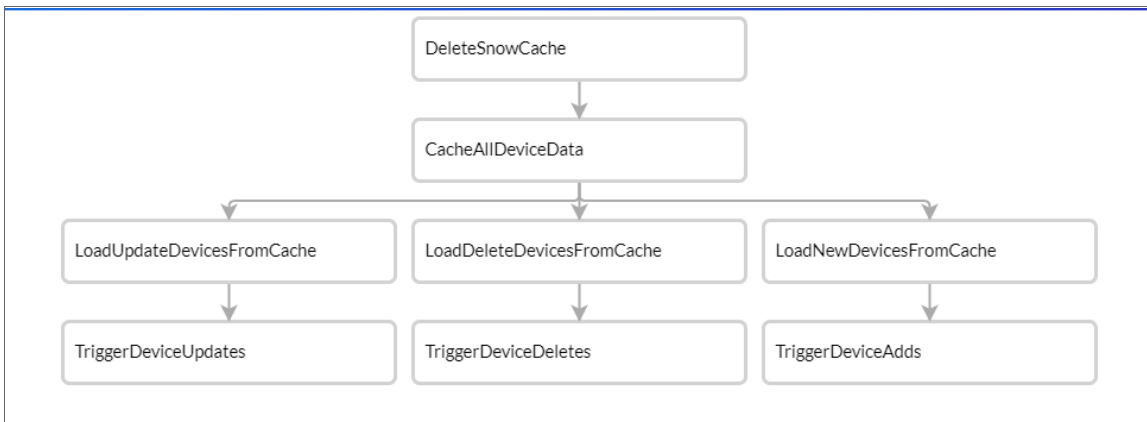
- **Standard**. Standard steps do not require any previously collected data to perform. Standard steps are generally used to generate data to perform a transformation or a database insert. These steps can be run independently and concurrently.
- **Aggregated**. Aggregated steps require data that was 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 steps by sending a GET request using the [API /steps 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:

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

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.

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.

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.

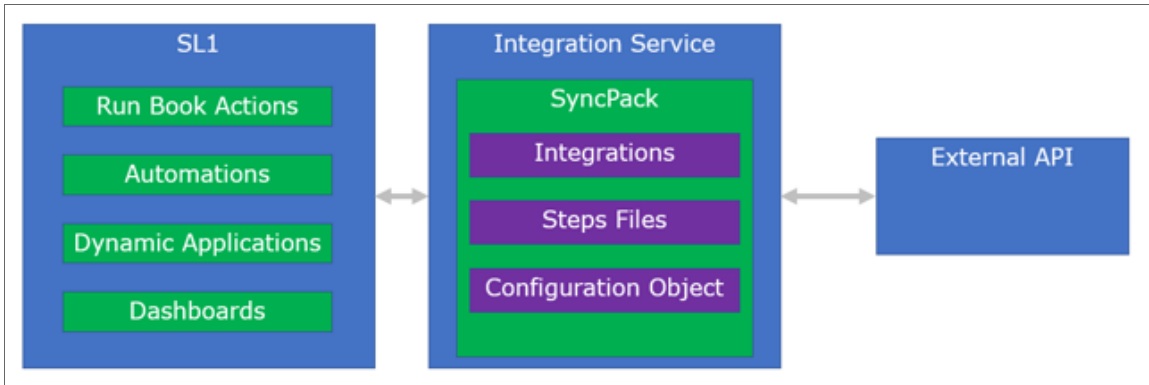
This chapter covers the following topics:

<i>Integration Service Architecture</i>	14
<i>Prerequisites for the Integration Service</i>	15
<i>System Requirements</i>	15
<i>Installing the Integration Service</i>	16
<i>Installing the Integration Service via ISO</i>	16
<i>Installing the Integration Service via RPM</i>	19
<i>Configuring the Integration Service</i>	21
<i>Setting a Hard Memory Limit in Docker</i>	21
<i>Setting a Soft Memory Limit in the Worker Environment</i>	21
<i>Upgrading the Integration Service</i>	22
<i>Changing the Integration Service Password</i>	23
<i>Configuring a Proxy Server</i>	23
<i>Configuring Security Settings</i>	24
<i>Changing the HTTPS Certificate</i>	24
<i>Using Password and Encryption Key Security</i>	24

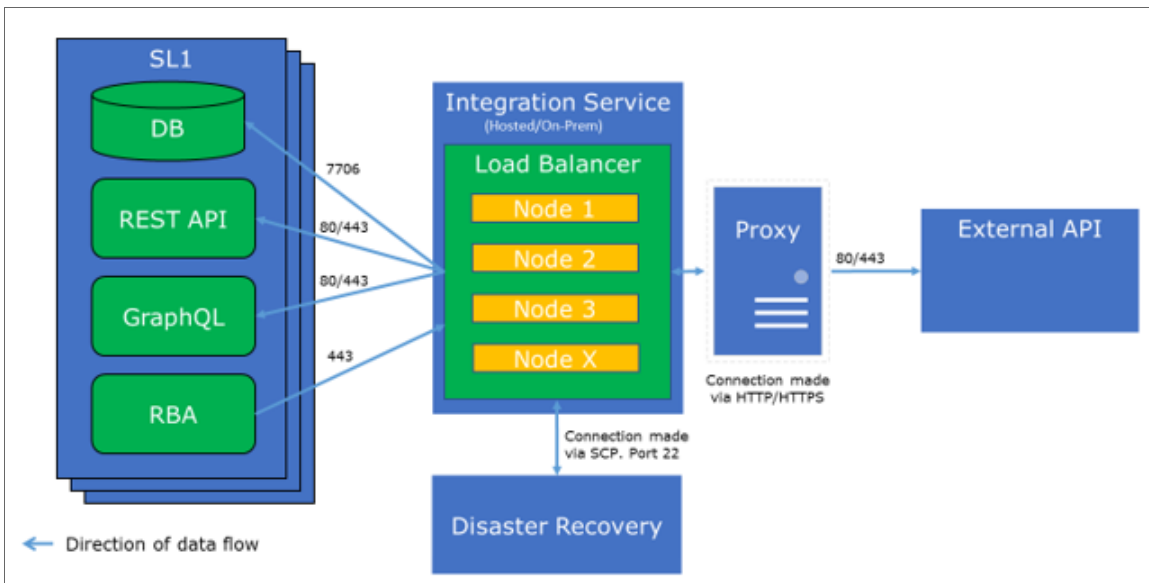
Monitoring the Integration Service	25
<i>PowerPacks for Monitoring Elements of the Integration Service</i>	25
<i>Monitoring the Integration Service with SLI</i>	26
<i>Monitoring the RabbitMQ Service</i>	29
<i>Monitoring the Couchbase Service</i>	32
<i>Integration Service System Diagnostics</i>	36
<i>Integration Service Endpoints</i>	37
<i>Integration Service Log Files</i>	39

Integration Service Architecture

The following diagram details the various elements that are contained in SL1 and the Integration Service, and how the Integration Service sits between the core SL1 platform and an external data platform:



The following diagram provides an example of the high-level architecture of an Integration Service system with High Availability, Disaster Recovery, and a proxy configured:



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.
- Python.
- Postman for interacting with the Integration Service API.
- Couchbase. For more information about using Couchbase, see [Helpful Couchbase Commands](#).
- Docker. For more information about using Docker, see [Helpful Docker Commands](#) and <https://docs.docker.com/engine/reference/commandline/cli/>.

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. The Integration Service requires the **docker-ce 18.06** version of Docker.

System Requirements

The Integration Service has the following minimum system requirements. Please note that these system requirements ultimately depend on the amount of workload you plan on running on your Integration Service:

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

NOTE: The Integration Service needs its own dedicated memory.

The following table offers a conservative starting point for sizing based on a typical environment (any object being processed by the Integration Service is considered a synced object):

Minimum at 1,000 Synced Objects			Minimum at 10,000 Synced Objects			Minimum at 50,000 Synced Objects		
RAM (GB)	Cores	Disk (GB)	RAM (GB)	Cores	Disk (GB)	RAM (GB)	Cores	Disk (GB)
24	8	100	36	8	100	48	8	200

All workloads are different. Storage requirements will vary based upon monitoring depth, frequency of integrations, and length of retention. Sizing recommendations may differ based on multi-SL 1 stack support.

The Integration Service also requires the following versions:

- SL1 version 8.10.0 or later.
- **ap2** version of 5.64.1 or later of the new user interface for SL1. For more information, see the *Introduction to SL1* manual.

ScienceLogic recommends that you use the following ports to access the tools used by the Integration Service platform:

- Port 8091: Couchbase Dashboard
- Port 8081: Docker Visualizer
- Port 15672: RabbitMQ Dashboard (use *guest/guest* as the username/password)

The following table lists the port access required by the Integration Service:

Source IP	Integration Service Destination	Integration Service Source Port	Destination Port	Requirement
Integration Service	SL1 Database	Any	TCP 7706	SL1 Database Access
Integration Service	SL1 API	Any	TCP 443	SL1 API Access
SL1 Run Book Action	Integration Service	Any	TCP 443	Send SL1 data to Integration Service

ScienceLogic highly recommends that you disable all firewall session-limiting policies. Firewalls will drop HTTPS requests, which results in data loss.

Installing the Integration Service

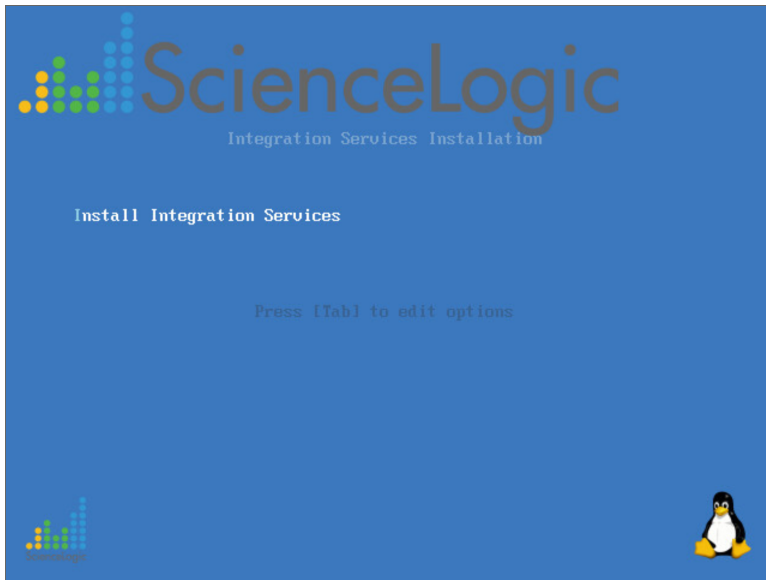
You can install the Integration Service via ISO to a server on your network, or via RPM to a cloud-based server.

Installing the Integration Service via ISO

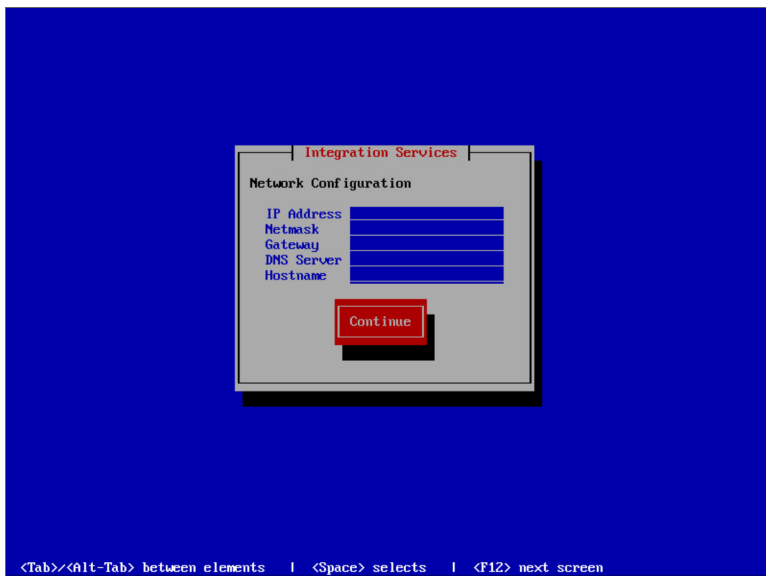
To install the Integration Service via ISO file:

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

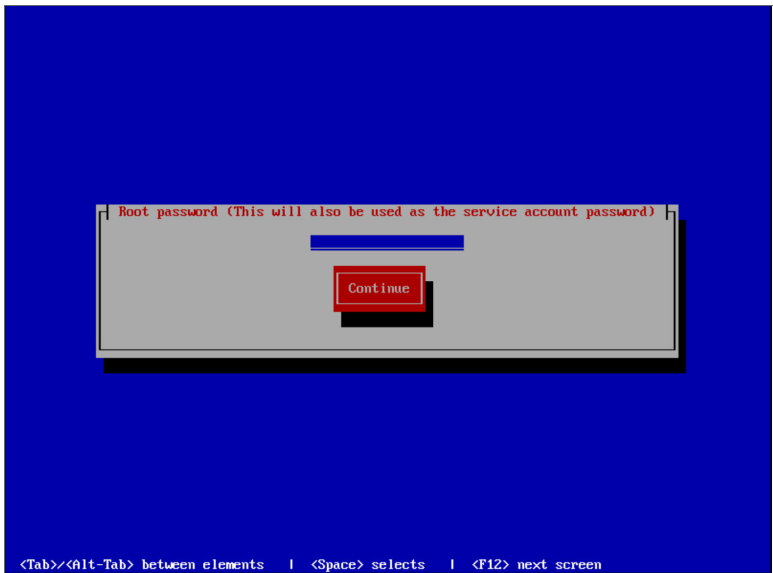
- Using your hypervisor or bare-metal (single-tenant) server of choice, mount and boot from the Integration Service ISO. The Integration Service Installation window appears:



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

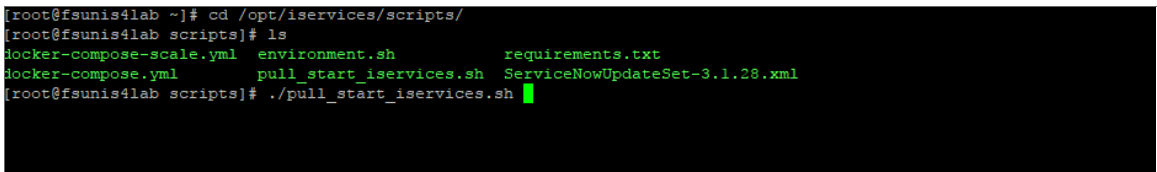


4. Complete the following fields:
 - **IP Address.** Type the primary IP address of the Integration Service server.
 - **Netmask.** Type the netmask for the primary IP address of the Integration Service server.
 - **Gateway.** Type the IP address for the network gateway.
 - **DNS Server.** Type the IP 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. The password must be at least six characters and no more than 24 characters, and all special characters are supported.
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
```

Installing the Integration Service via RPM

To install a single-node Integration Service via RPM to a cloud-based environment:

1. Deploy a new Oracle Linux 7.6 virtual machine in Amazon Web Service (AWS) EC2:
 - Search for owner `131827586825`. This is Oracle's owner ID.
 - Select a virtual machine running Oracle Linux 7.6 or greater for installation. The following image shows AMI `ami-0d401c561deebd182`.



- Size the virtual machine:
 - *Single node deployments.* The minimum is `t2.xlarge` (four CPUs with 16 GB memory), and ScienceLogic recommends `t2.2xlarge` (8 CPUs with 32 GB memory).
 - *Cluster deployments:* Cluster deployments depend on the type of node you are deploying. Refer to the separate hosted environment guide for more sizing information. ScienceLogic recommends that you allocate at least 50 GB or more for storage.
2. Define the security group:
 - Only inbound port 443 needs to be exposed to any of the systems that you intend to integrate.
 - For Integration Service version 1.8.2 and later, port 8091 is exposed through `https`. ScienceLogic recommends that you make port 8091 available externally to help with troubleshooting.

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	Description ⓘ
Custom UDP Rule	UDP	8091	72.165.86.42/32	IS DB Admin Interf...
SSH	TCP	22	0.0.0.0/0	IS SSH access
Custom TCP Rule	TCP	8091	72.165.86.42/32	IS DB Admin interf...
HTTPS	TCP	443	0.0.0.0/0	IS HTTPS access

3. Ensure that the latest required packages are installed by running the following commands:

```
sudo yum install wget
sudo pip install docker-compose
wget https://download.docker.com/linux/centos/7/x86_
64/stable/Packages/docker-ce-18.06.1.ce-3.el7.x86_64.rpm && sudo yum install
docker-ce-18.06.1.ce-3.el7.x86_64.rpm
```

4. Create the Docker group:

```
sudo groupadd docker
```

5. Add your user to the Docker group:

```
sudo usermod -aG docker $USER
```

6. Log out and log back in to ensure that your group membership is re-evaluated.

7. Configuration updates:

```
setenforce 0
vim /etc/sysconfig/selinux
SELINUX=permissive
sudo systemctl enable docker
sudo systemctl start docker
sudo yum install yum-utils
sudo yum-config-manager --enable ol7_addons ol7_optional_latest ol7_latest
sudo yum-config-manager --disable ol7_ociyum_config
wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
rpm -Uvh epel-release-7*.rpm
sudo yum update
sudo pip install docker-compose
sudo yum install firewalld
systemctl enable firewalld
systemctl start firewalld
systemctl disable iptables
```

8. Firewall commands:

```
firewall-cmd --add-port=2376/tcp --permanent
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 --add-protocol=esp --permanent
firewall-cmd --reload
```

9. Copy the Integration Service RPM to the AWS instance of installation:

```
sudo yum install s11-integration-services
systemctl restart docker
```

10. Create a password:

```
sudo printf 'isadmin' > /etc/iservices/is_pass
```

11. Pull and start services:

```
/opt/iservices/scripts/pull_start_iservices.sh
```

NOTE: ScienceLogic recommends that you switch to an Amazon EC2 user soon instead of running all the commands on root.

Configuring the Integration Service

If you have multiple workers running on the same Integration Service system, you might want to limit the amount of memory allocated for each worker. This helps prevent memory leaks, and also prevents one worker using too many resources and starving other workers. You can apply these limits in two ways:

- Set a hard memory limit in Docker (this is the default)
- Set a soft memory limit in the worker environment

Setting a Hard Memory Limit in Docker

Setting a memory limit for the worker containers in your `docker-compose.yml` file sets a hard limit. If you set a memory limit for the workers in the `docker-compose` file and a worker exceeds the limit, the container is terminated via `SIGKILL`. If the currently running task caused memory usage to go above the limit, that task might not be completed, and the worker container is terminated in favor of a new worker. This setting helps to prevent a worker from endlessly running and consuming all memory on the Integration Service system.

You can configure the hard memory limit in the `steprunner` service of the `docker-compose.yml` file:

```
deploy:
  resources:
    limits:
      memory: 2G
```

Setting a Soft Memory Limit in the Worker Environment

You can set the memory limit for a worker application, and not at the Docker level. Setting the memory limit at the application level differs from the hard memory limit in Docker in that if a worker exceeds the specified memory limit, that worker is not immediately terminated via `SIGKILL`. Instead, if a worker exceeds the soft memory limit, the worker waits until the currently running task is completed to recycle itself and start a new process. As a result, tasks will complete if a worker crosses the memory limit, but if a task is running infinitely with a memory leak, that task might consume all memory on the host.

NOTE: The soft memory limit is less safe from memory leaks than the hard memory limit.

You can configure the soft memory limit with the worker environment variables. The value is in KiB (1024 bytes). Also, each worker instance contains three processes for running tasks. The memory limit applies to each individual instance, and not the container as a whole. For example, a 2 GB memory limit for the container would translate to 2 GB divided by three, or about 700 MB for each worker:

```
steprunner:
  image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1
  environment:
    additional_worker_args: ' --max-memory-per-child 700000'
```

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. You must be root to upgrade the RPM file.
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, such as **sl1-integration-services-1.x.0-1.x86_64**.

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

```
systemctl restart docker
```

6. After the RPM is installed, run the following Docker command:

```
docker stack rm iservices
```

7. Re-deploy the Docker stack to update the containers:

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

8. 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 .

9. 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 at least six characters and no more than 24 characters, and all special characters are supported. The password cannot be the same as the old password.

NOTE: If an Integration Service user makes multiple incorrect login attempts, the Integration Service locks out the user. To unlock the user, run the following command: `unlock_user -u <username>`

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.

TIP: The Integration Service supports all special characters in passwords.

Monitoring the Integration Service

You can use a number of ScienceLogic PowerPacks to help you monitor the health of your Integration Service system. This section describes those PowerPacks and additional resources and procedures you can use to monitor the components of the Integration Service.

PowerPacks for Monitoring Elements of the Integration Service

You can download the following PowerPacks from the ScienceLogic [Customer Portal](#) to help you monitor your Integration Service system:

- [ScienceLogic: Integration Service PowerPack](#). This PowerPack monitors the state of tasks running on the Integration Service. This PowerPack alerts users on SL1 if an application on the Integration Service fails. This PowerPack lets you monitor the status of your integration applications, and based on the events generated by this PowerPack, you can diagnose why applications failed on the Integration Service.
- [Docker PowerPack](#): This PowerPack monitors the Docker containers, services, and Swarm that manages the Integration Service containers. This PowerPack also monitors the Integration Service when it is configured for High Availability. Use version 103 or later of the Docker PowerPack to monitor Integration Service services in SL1.
- [RabbitMQ PowerPack](#). This PowerPack monitors RabbitMQ configuration data and performance metrics using Dynamic Applications. You can use this PowerPack to monitor the RabbitMQ service used by the Integration Service.
- [Couchbase PowerPack](#): This PowerPack monitors the Couchbase database that the Integration Service uses for storing the cache and various configuration and application data. This data provides insight into the health of the databases and the Couchbase servers.

- **Linux Base PowerPack:** This PowerPack monitors Linux Servers with SSH. The Integration Service ISO is built on top of an Oracle Linux Operating System. This PowerPack provides key performance indicators to provide insight into how your Integration Service server is performing.

For more information about installing and using the *ScienceLogic: Integration Service PowerPack*, see [Using SL1 to Monitor the Integration Service](#).

Monitoring the Integration Service with SL1

You can use each of the PowerPacks listed above to monitor different aspects of the Integration Service. Because of the various technologies being used by the Integration Service, you should complete the following steps in the specified order to set up monitoring of the Integration Service within SL1.

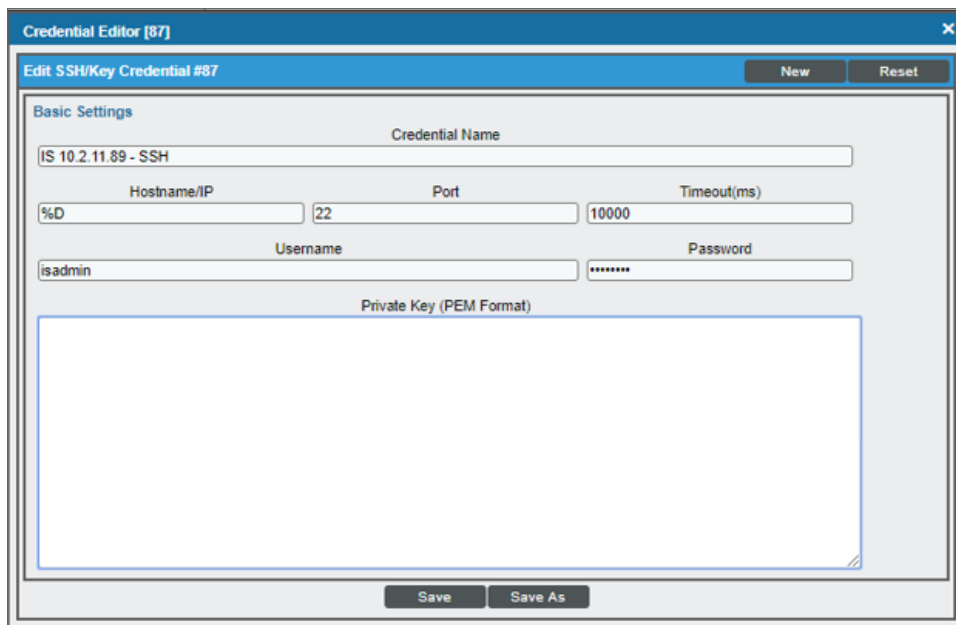
First, make sure you have installed all of the PowerPacks listed above on your SL1 system. After installing the PowerPacks, follow the steps below.

Setting Up Discovery

To discover the Integration Service in SL1, use the Docker PowerPack to discover the Integration Service using the SSH credentials.

To discover the Integration Service:

1. Ensure that you have installed the Linux Base PowerPack, the Integration Service PowerPack, and the Docker PowerPack on your SL1 system.
2. In SL1, navigate to the **Credential Management** page (System > Manage > Credentials) and click the wrench icon (🔧) for the **Docker SSH** credential:



3. Update the **Credential Name** field with a new descriptive name.
4. Update the **Username** and **Password** fields with the username and password for your Integration Service instance.
5. Click **[Save As]** and close the credential modal page.
6. Navigate to the Discovery Control Panel (System > Manage > Discovery) and click the **[Create]** button to create a new discovery session:

The screenshot shows the 'Discovery Session Editor | Editing Session [2]' window. It is divided into four main sections:


- Identification Information:** Includes fields for 'Name' (containing 'IS 10.2.11.89') and 'Description'.
- IP and Credentials:** Contains an 'IP Address/Hostname Discovery List' with '10.2.11.89', an 'Upload File' section with a 'Browse...' button, and two lists of credentials: 'SNMP Credentials' (with 'EM7 Default V3' selected) and 'Other Credentials' (with 'Cisco Dial Peer - Example' selected).
- Detection and Scanning:** Features dropdown menus for 'Initial Scan Level', 'Scan Throttle', and 'Port Scan All IPs', all set to '[System Default (recommended)]'. It also has a 'Port Scan Timeout' dropdown. Below these is a 'Detection Method & Port' list with options like 'UDP: 161 SNMP', 'TCP: 1 - tcpmux', etc. At the bottom of this section are input fields for 'Interface Inventory Timeout (ms)' (600000) and 'Maximum Allowed Interfaces' (10000), and a 'Bypass Interface Inventory' checkbox.
- Basic Settings:** Includes checkboxes for 'Discover Non-SNMP' (checked), 'Model Devices' (checked), and 'DHCP' (unchecked). It also has input fields for 'Device Model Cache TTL (h)' (2) and 'Collection Server PID: 1' (fsunem7aio42), and a dropdown for 'Organization' (System). There is a section for 'Add Devices to Device Group(s)' with a list containing 'None Servers'. At the bottom is an 'Apply Device Template' dropdown set to '[Choose a Template]'.

At the bottom of the window are 'Save' and 'Save As' buttons, and a 'Log All' checkbox.

7. Complete the following fields:

- **Name.** Type a name for the discovery session. This name is displayed in the list of discovery sessions in the **Discovery Control Panel** page.
- **IP Address.** Type the IP addresses or hostname for your Integration Service. For a clustered Integration Service environment, list all Docker hosts.
- **Other Credentials.** Select the Docker SSH credential you updated in steps 2-4.
- **Discover Non-SNMP.** Specifies whether or not SL1 should discover devices that don't respond to SNMP requests.

8. Click **[Save]** and run your discovery session. After the detailed discovery is complete, various Docker and Linux Dynamic Applications should automatically align to your discovered Docker Host for your Integration Service.
9. On the **Device Manager** page (Registry > Devices > Device Manager), click the wrench icon (🔧) for your new device and verify on the **[Collections]** tab that the following Dynamic Applications were automatically aligned:

Device Name: 10.2.11.89	Managed Type: Physical Device	
IP Address / ID: 10.2.11.89 30	Category: Servers	
Class: Linux	Sub-Class: Oracle Linux Server 7.3	
Organization: System	Uptime: 0 days, 00:00:00	
Collection Mode: Active	Collection Time: 2019-02-14 03:53:00	
Description:	Group / Collector: CUG fsunem7aio42	
Device Hostname:		

Dynamic Application™ Collections						Expand	Actions	Reset	Guide
Dynamic Application	ID	Pol Frequency	Type	Credential					
+ Docker: Containers Performance	1459	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Docker: Host Performance	1472	15 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Docker: Image Performance	1474	15 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Linux: CPU Cores	1494	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Linux: CPU Stats	1487	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Linux: Disk IOPS Stats	1486	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Linux: File System Stats	1493	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Linux: ICMP Stats	1484	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Linux: Interface Stats	1483	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Linux: Memory Stats	1482	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Linux: System Load	1479	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Linux: TCP Stats	1477	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Linux: UDP Stats	1475	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Linux: Zombie Process	1489	5 mins	Snippet Performance	IS 10.2.11.89 - SSH					
+ Docker: Container Discovery	1456	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Docker: Host Configuration	1460	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Docker: Host Reclassification	1473	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Docker: Image Configuration	1454	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Docker: Network Configuration	1470	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Docker: Swarm Cluster Discovery	1467	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Linux: Configuration Cache	1491	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Linux: CPU Config	1488	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Linux: Hardware Config	1485	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Linux: Network Config	1481	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Linux: Performance Cache	1492	5 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Linux: Route Table	1490	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Linux: System Config	1480	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Linux: TCP Services	1478	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					
+ Linux: UDP Services	1476	15 mins	Snippet Configuration	IS 10.2.11.89 - SSH					

10. To view your newly discovered device, navigate to the **Device Components** page (Registry > Devices > Device Components). If you do not see your newly discovered Docker Host, wait for the dynamic applications on the Docker host to finish modeling out its component devices. A Docker Swarm virtual root device will also be discovered. After discovery finishes, you should see the following devices representing your Integration Service on the **Device Components** page:

Device Components Devices Found [2]			
Device Name	IP Address	Device Category	Device Class Sub-class
1. + 10.2.11.89	10.2.11.89	Servers	Linux Oracle Linux Server 7.3
2. - Docker Swarm qc80ytui3fp8afs0cnbcgmpng	--	EM7	ScienceLogic Integration Service
Device Name	IP Address	Device Category	Device Class Sub-class
1. - iservices	--	Service	Stack Docker Stack
Device Name	IP Address	Device Category	Device Class Sub-class
1. iservices_contentapi	--	Service	Service Docker Service
2. iservices_couchbase	--	Service	Service Docker Service
3. iservices_flower	--	Service	Service Docker Service
4. iservices_gui	--	Service	Service Docker Service
5. iservices_rabbitmq	--	Service	Service Docker Service
6. iservices_redis	--	Service	Service Docker Service
7. iservices_scheduler	--	Service	Service Docker Service
8. iservices_stepperunner	--	Service	Service Docker Service
9. iservices_visual	--	Service	Service Docker Service

11. If the root device is modeled with a different device class, click the **[Actions]** button on the **Device Properties** window and select *Device Class*, and select *ScienceLogic | Integration Service* as the Device Class for your Docker Swarm.

NOTE: There is a known issue where the Docker Host will be re-classified to an Oracle Linux Server. This will be addressed in an upcoming release of the Docker and Linux PowerPacks.

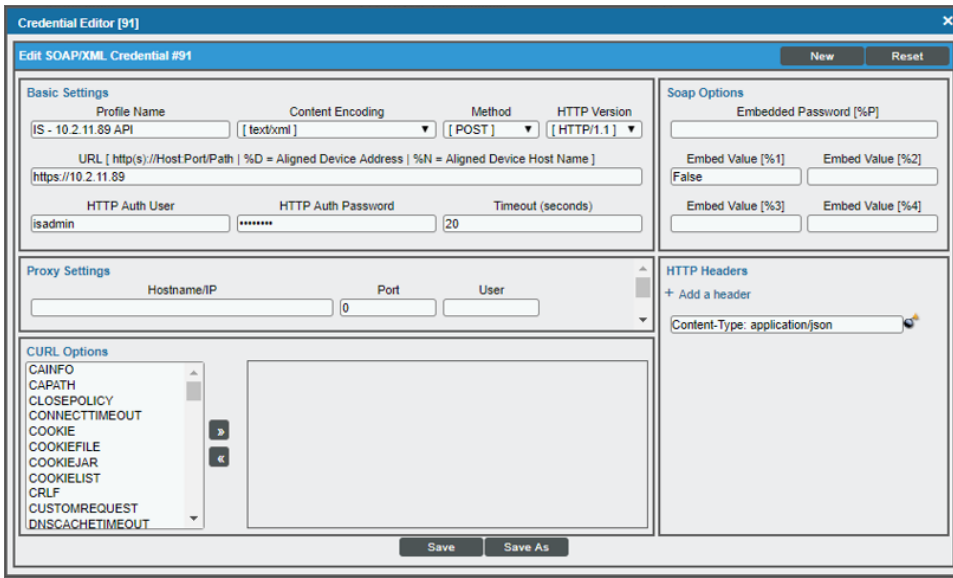
Monitoring the RabbitMQ Service

After you discover your Integration Service, you can monitor the queue service. The *ScienceLogic: Integration Service* PowerPack creates a Major event in SL1 for any integration applications on your Integration Service that are in a *Failed* state. You can also monitor the RabbitMQ service with the *RabbitMQ* PowerPack.

You can access the RabbitMQ Dashboard by using port 15672 with your Integration Service, such as https://<integration_service>:15672 (use *guest/guest* as the username/password).

To monitor the Rabbit MQ service:

1. Ensure that you have installed the ScienceLogic: Integration Service PowerPack on your SL1 system.
2. In SL1, navigate to the **Credential Management** page (System > Manage > Credentials) and click the wrench icon (🔧) for the **IS - Example** credential:

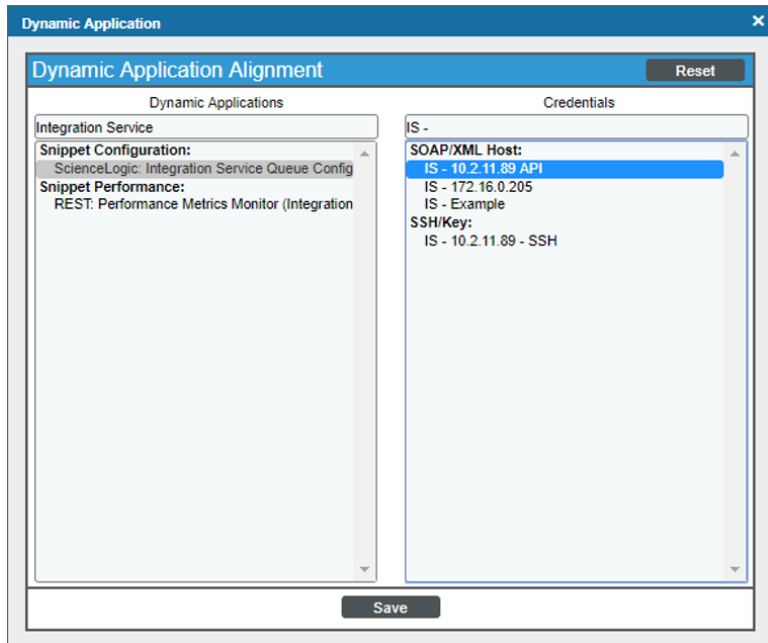


3. Complete the following fields:
 - **Profile Name**. Type a new descriptive name for the credential.
 - **URL**. Type the full URL for your Integration Service.
 - **HTTP Auth User**. Type the username for your Integration Service instance.
 - **HTTP Auth Password**. Type the password for your Integration Service instance.

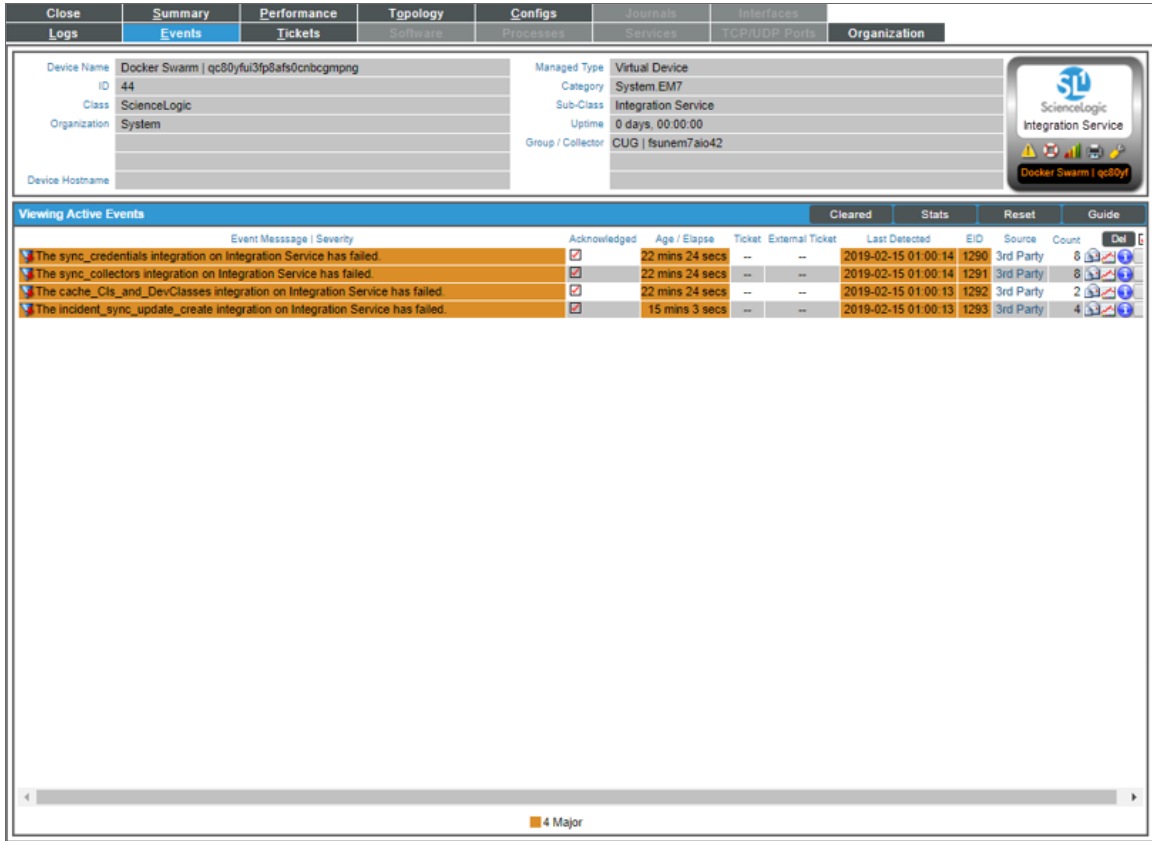
NOTE: For a clustered Integration Service environment, point the credentials at the Integration Service's load balancer. The example above is for a single node deployment.

4. Click **[Save As]**.
5. Go to the **Device Components** page (Registry > Devices > Device Components) and click the wrench icon (🔧) for the "ScienceLogic | Integration Service" root device.
6. Click the **[Collections]** tab on the **Device Properties** window.

- Align the "ScienceLogic: Integration Service Queue Configuration" Dynamic Application by clicking the **[Actions]** and selecting *Add Dynamic Application*. The **Dynamic Application Alignment** modal appears:



- Select the "ScienceLogic: Integration Service Queue Configuration" Dynamic Application and select the IS credential that you created in steps 2-3. Click **[Save]**. This Dynamic Application queries the Integration Service every 15 minutes by default to retrieve information about any failed integrations, which generates a Major event in SL1 (the events auto-expire after 90 minutes):



TIP: The events generated by this Dynamic Application include the **Integration ID**, which you can use to find the relevant integration application on your Integration Service instance. Copy the name in the event message and navigate to `https://<integration_service_host>/integrations/<integration_ID>`.

- To view more information about your failed integration applications, navigate to the **[Configurations]** tab for the device and click the report for the "ScienceLogic: Integration Service Queue Configuration" Dynamic Application. This configuration report shows you more information about the failed integrations on your Integration Service instance. For example, you can use the **Last Run ID** field to find the exact logs for a specific execution of the integration application. To do this, copy the *Integration ID* and the *Last Run ID* and navigate to `https://<integration_service_host>/integrations/<integration_ID>?runid=<last_run_id>`

Monitoring the Couchbase Service

Couchbase stores all cache and configuration data on the Integration Service. Monitoring the performance of your Integration Service is critical in ensuring the health your Integration Service instance.

To monitor the Couchbase service:

1. Ensure that you have installed the Couchbase PowerPack on your SL1 system.
2. In SL1, navigate to the **Credential Management** page (System > Manage > Credentials) and click the wrench icon (🔧) for the **Couchbase Sample** credential:

Credential Editor [110]

Edit SOAP/XML Credential #110 New Reset

Basic Settings

Profile Name: Couchbase Sample Credential | Content Encoding: [text/xml] | Method: [POST] | HTTP Version: [HTTP/1.1]

URL [http(s)://Host:Port/Path | %D = Aligned Device Address | %N = Aligned Device Host Name]
https://10.2.11.22:8091

HTTP Auth User: isadmin | HTTP Auth Password: [masked] | Timeout (seconds): 20

Soap Options

Embedded Password [%P]: []

Embed Value [%1]: False | Embed Value [%2]: []

Embed Value [%3]: [] | Embed Value [%4]: []

Proxy Settings

Hostname/IP: [] | Port: 0 | User: [] | Password: []

CURL Options

CAINFO, CAPATH, CLOSEPOLICY, CONNECTTIMEOUT, COOKIE, COOKIEFILE, COOKIEJAR, COOKIELIST, CRLF, CUSTOMREQUEST, DNSCACHETIMEOUT

HTTP Headers

+ Add a header

Content-Type: application/json

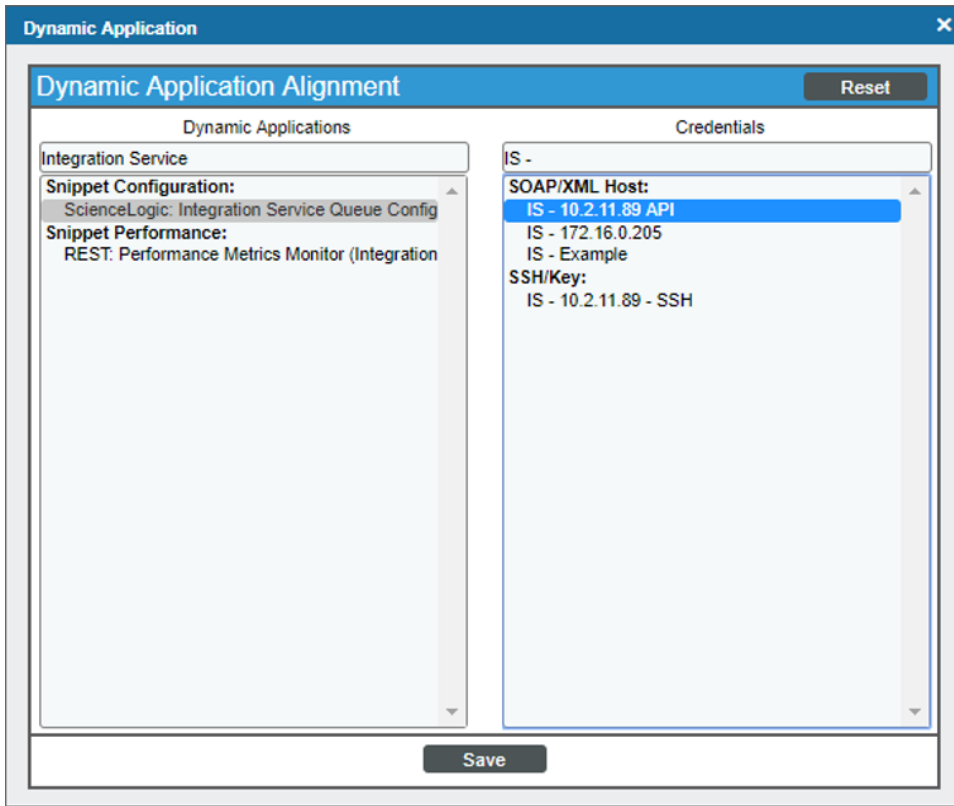
Save Save As

3. Complete the following fields:
 - **Profile Name**. Type a new descriptive name for the credential.
 - **URL**. Type the full URL for your Integration Service. Ensure that the port 8091 is appended to the hostname. For version 1.8.2 and later, use *https* for this URL.
 - **HTTP Auth User**. Type the username for your Integration Service instance.
 - **HTTP Auth Password**. Type the password for your Integration Service instance.

NOTE: For a clustered Integration Service environment, point the Couchbase credentials at the load balancer for the Integration Service. The example above is for a single node deployment.

4. Click **[Save As]**.
5. Go to the **Device Components** page (Registry > Devices > Device Components) and expand the “ScienceLogic | Integration Service” root device by clicking on the + icon.
6. Click the wrench icon (🔧) for the “Stack | Docker Stack” component device and click the **[Collections]** tab on the **Device Properties** window.

- Align the "Couchbase: Pool Discovery" Dynamic Application by clicking the **[Actions]** and selecting *Add Dynamic Application*. The **Dynamic Application Alignment** modal appears:



- Select the "Couchbase: Pool Discovery" Dynamic Application and select the IS credential that you created in steps 2-3. Click **[Save]**.
- Align the "Couchbase: Component Count" Dynamic Application by clicking the **[Actions]** and selecting *Add Dynamic Application*.
- Select the "Couchbase: Component Count" Dynamic Application and select the IS credential that you created in steps 2-3. Click **[Save]**. SL1 models out your Couchbase components and provides you with additional information about the usage of the Couchbase service.

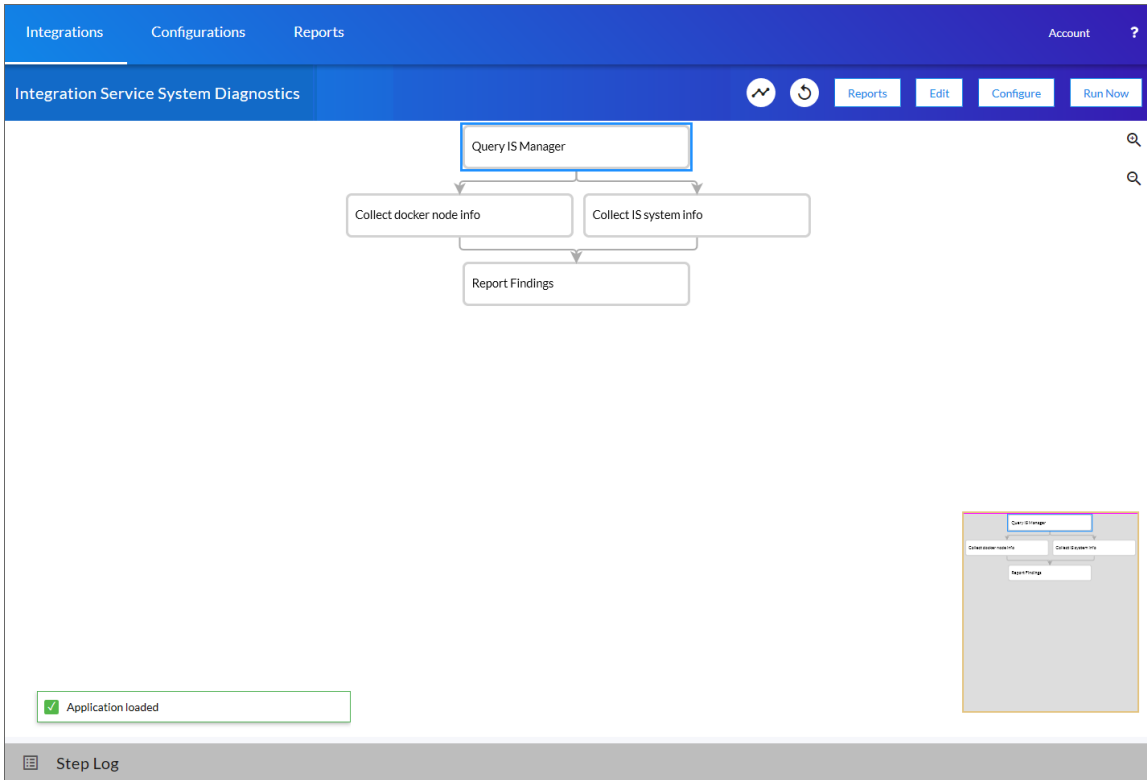
- Navigate to the **Device Components** page (Registry > Devices > Device Components) to see the components:

	Device Name *	IP Address	Device Category	Device Class Sub-class
1. +	10.2.11.89	10.2.11.89	Servers	Linux Oracle Linux Server 7.3
2. -	Docker Swarm qc80yfui3fp8afs0cnbcgmpng	--	EM7	ScienceLogic Integration Service
1. -				
	iservices	--	Service	Stack Docker Stack
1. -				
	default	--	Pool	Couchbase Pool
1. -				
1.	content	--	Volume	Couchbase Bucket
2.	couchbase.isnet:8091	--	Node	Couchbase Node
3.	logs	--	Volume	Couchbase Bucket
2.	iservices_contentapi	--	Service	Service Docker Service
3.	iservices_couchbase	--	Service	Service Docker Service
4.	iservices_flower	--	Service	Service Docker Service
5.	iservices_gui	--	Service	Service Docker Service
6.	iservices_rabbitmq	--	Service	Service Docker Service
7.	iservices_redis	--	Service	Service Docker Service
8.	iservices_scheduler	--	Service	Service Docker Service
9.	iservices_steprunner	--	Service	Service Docker Service
10.	iservices_visual	--	Service	Service Docker Service

Integration Service System Diagnostics

The "Integration Service System Diagnostics" integration application lets you view platform diagnostics for the Integration Service. You can use the information displayed in these diagnostics to help you troubleshoot issues with the different tools used by the Integration Service.

To generate the report, select the "Integration Service System Diagnostics" integration application from the **[Integrations]** tab and click the **[Run Now]** button:



TIP: The "IS - System Diagnostic Configuration Example" configuration file contains the structure needed for the "Integration Service System Diagnostics" integration application. You can use this configuration as a template when running this integration application.

Running the "Integration Service System Diagnostics" integration application generates a report that you can access on the **[Reports]** tab:

The screenshot shows the 'Reports' tab with a sub-tab for 'is_system_diagnostics'. A 'Delete' button is visible in the top right. Below the sub-tab is a 'Details' table:

Details	
Application Name	is_system_diagnostics
Application ID	IS_System_Diagnostics
Created (UTC -5)	14 Jan, 2019 16:32:59

Below the details is a table for 'IS Integrations' with columns for 'Device Mappings', 'Incident integration run count', and 'Schedule'. A single row is shown with a run count of 0 and a detailed schedule configuration.

At the bottom, there is a section for 'Node 10_2_11_22 System' with columns for 'cpu', 'docker version', 'is rpm version', and 'kernel version'. The 'cpu' column contains extensive hardware and system statistics. The 'docker version' column shows '18.06.1-ce, build e68fc7a'. The 'is rpm version' column lists installed packages and their details. The 'kernel version' column shows 'Linux is22 3.10.0-862.3.2.el7.x86_64 #1 SMP Mon May 21 17:56:51 PDT 2018 x86_64 x86_64 x86_64 GNU/Linux'.

This diagnostic report displays overall Integration Service settings, such as the Integration Service version, Docker version, kernel version, hostname, cluster settings, scheduled applications, CPU and memory statistics, installation date, and cache information.

TIP: If you are using a specific integration application that you want to monitor with the "Integration Service System Diagnostics" integration application, click the **[Configure]** button and type the name of that integration application in the *incident_create_app* field of the **Configuration** pane.

Integration Service Endpoints

This section provides additional technical details about monitoring the Integration Service. The following information is also available in the PowerPacks listed above.

Flower API

The following Flower API endpoints return data about the Flower tasks, queues, and workers. The **tasks** endpoint returns data about task status, runtime, exceptions, and application names. You can filter this endpoint to retrieve a subset of information, and you can combine filters to return a more specific data set.

/flower/api/tasks. Retrieve a list of all tasks.

`/flower/api/tasks?app_id={app_id}`. Retrieve a list of tasks filtered by `app_id`.

`/flower/api/tasks?app_name={app_name}`. Retrieve a list of tasks filtered by `app_name`.

`/flower/api/tasks?started_start=1539808543&started_end=1539808544`. Retrieve a list of all tasks received within a time range.

`/flower/api/tasks?state=FAILURE|SUCCESS`. Retrieve a list of tasks filtered by state.

`/flower/api/workers`. Retrieve a list of all queues and workers

To view this information in the Flower user interface navigate to `<hostname_of_integration_service_system>/flower`.

For more information, see the [Flower API Reference](#).

NOTE: If you use the *ScienceLogic: Integration Service PowerPack* to collect this task information, the PowerPack will create events in SL1 if a Flower task fails.

Couchbase API

The following Couchbase API endpoints return data about the Couchbase service. The **pools** endpoint represents the Couchbase cluster. In the case of the Integration Service, each **node** is a Docker service, and **buckets** represent the document-based data containers. These endpoints return configuration and statistical data about each of their corresponding Couchbase components.

`<hostname_of_integration_service_system>:8091/pools/default`. Retrieve a list of pools and nodes.

`<hostname_of_integration_service_system>:8091/pools/default/buckets`. Retrieve a list of buckets.

To view this information in the Couchbase Administrator user interface, navigate to `<hostname_of_integration_service_system>:8091`.

For more information, see the [Couchbase API Reference](#).

NOTE: You can also use the *Couchbase PowerPack* to collect this information.

Docker Statistics

You can collect Docker information by using SSH to connect to the Docker socket. You cannot currently retrieve Docker information by using the API.

To collect Docker statistics:

1. Use SSH to connect to the Integration Service instance.
2. Run the following command:

```
curl --unix-socket /var/run/docker.sock http://docker<PATH>
```

where <PATH> is one of the following values:

- /info
- /containers/json
- /images/json
- /swarm
- /nodes
- /tasks
- /services

NOTE: You can also use the *Docker PowerPack* to collect this information.

Integration Service Log Files

Use the following procedures to help you locate and understand the contents of the various log files related to the Integration Service.

Accessing Docker Log Files

The Docker log files contain information logged by all containers participating in the Integration Service. The information below is also available in the PowerPacks listed above.

To access Docker log files:

1. Use SSH to connect to the Integration Service instance.
2. Run the following Docker command:

```
docker service ls
```

3. Note the Docker service name, which you will use for the <service_name> value in the next step.
4. Run the following Docker command:

```
docker service logs -f <service_name>
```

Accessing Local File System Logs

The local file system logs display the same information as the Docker log files. These log files include debug information for all of the Integration Service integration applications and all of the Celery worker nodes.

To access local file system logs:

1. Use SSH to connect to the Integration Service instance.
2. Navigate to the `/var/log/iseservices` directory to view the log files.

Understanding the Contents of Log Files

The pattern of deciphering log messages applies to both Docker logs and local log files, as these logs display the same information.

The following is an example of a message in a Docker log or a local file system log:

```
"2018-11-05 19:02:28,312","FLOW","12","device_sync_sciencelogic_to_servicenow","ipaas_logger","142","stop Query and Cache ServiceNow CIs|41.4114570618"
```

You can parse this data in the following manner:

```
'YYYY-MM-DD' 'HH-MM-SS,ss' 'log-level' 'process_id' 'is_app_name' 'file' 'lineOfCode'
'message'
```

To extract the runtime for each individual task, use regex to match on a log line. For instance, in the above example, there is the following sub-string:

```
"stop Query and Cache ServiceNow CIs|41.4114570618"
```

Use regex to parse the string above:

```
"stop ..... | ..."
```

where:

- Everything after the `|` is the time taken for execution.
- The string between `"stop"` and `|` represents the step that was executed.

In the example message, the `"Query and Cache ServiceNow CIs"` step took around 41 seconds to run.

Configuring the Integration Service for High Availability

Overview

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

<i>System Requirements for High Availability</i>	43
<i>Configuring a High Availability Environment for the Integration Service</i>	43
<i>Docker Swarm Requirements for High Availability</i>	44
<i>Docker Swarm Frequently Asked Questions for High Availability</i>	44
<i>Couchbase Database Requirements for High Availability</i>	45
<i>Couchbase Database Frequently Asked Questions for High Availability</i>	45
<i>RabbitMQ Clustering and Persistence for High Availability</i>	46
<i>RabbitMQ Option 1: Persisting Queue to Disk on a Single Node (Default Configuration)</i>	46
<i>RabbitMQ Option 2: Clustering Nodes with Persistent Queues on Each Node</i>	47
<i>Checking the Status of a RabbitMQ Cluster</i>	48
<i>Configuring Clustering and High Availability</i>	49
<i>Configuring Docker Swarm</i>	49
<i>Configuring the Couchbase Database</i>	49
<i>Troubleshooting Issues with Couchbase Indexes</i>	52
<i>Manual Failover</i>	55
<i>Additional Configuration Information</i>	58
<i>Known Issues</i>	60

<i>Docker Network Alias is incorrect</i>	60
<i>Docker container on last swarm node cannot communicate with other swarm nodes</i>	60
<i>Couchbase service does not start, remains hung at nc -z localhost</i>	60
<i>Couchbase-worker fails to connect to master</i>	60
<i>The Integration Service user interface fails to start after a manual failover of the swarm node</i>	61
<i>The Integration Service user interface returns 504 errors</i>	61
<i>NTP should be used, and all node times should be in sync</i>	61
<i>Example Logs from Flower</i>	61

System Requirements for High Availability

The High Availability solution for the Integration Service has the following system requirements:

1. Ensure that your Integration Service system has been updated with `yum upgrade`.
2. Run the following commands to open up the proper firewall ports for Docker Swarm on each swarm node:

```
firewall-cmd --add-port=2376/tcp --permanent
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 --add-protocol=esp --permanent
firewall-cmd --reload
```

NOTE: If your system is fully yum-updated, you only have to run the following commands:

```
firewall-cmd --add-service docker-swarm --permanent
firewall-cmd --reload
```

3. Ensure that `/etc/iservices/is_pass` and `/etc/iservices/encryption_key` are identical on all clustered nodes.
4. Ensure that NTP is properly configured on all nodes:

- Run the following command:

```
edit /etc/chronyd.conf
```

- Add NTP servers. If you want to use the `pool.ntp.org` NTP servers, remove the `.ol.` from the domain names.
- Enable `chronyd` by running the following commands:

```
systemctl start chronyd
systemctl enable chronyd
timedatectl #ensure ntp is enabled is yes and ntp sync is yes
```

NOTE: The hardware sizing recommendations remain the same for a single-node or a clustered environment.

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 following best practices from Docker, Couchbase, and RabbitMQ. The topics in this section 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.

The only *network* requirement is that all three nodes must be able to communicate to each other. There are no additional network requirements. All nodes should be deployed in the same network, with as little latency as possible. ScienceLogic recommends less than 40 ms of latency for optimal performance.

When deploying as a cluster, you will need to have a copy of the certificate in each node of the cluster so that the GUI/API container can bounce around.

You will need an additional load balancer and VIP to ensure that you can access the Integration Service cluster from a single IP. You can access the Integration Service system from any of the three node IPs, but if you want to monitor the cluster in SL1, do not use the static IP from one of the nodes in case one of the nodes fail. With a load balancer, you will know the node in the cluster to which you are connecting, as the load balancer will always route to the running nodes if another node fails.

Finally, on the Integration Service cluster, all nodes are orchestrated under the same Docker Swarm and are active/active. Services and requests to the services will be evenly distributed among all of the nodes. If a node fails, the other nodes will also have a full replicated set of data, and will perform automatic failover. Any services on that failing node will automatically be migrated to the other functioning nodes to ensure availability.

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 in the 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:latest
  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.

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.
2. Ensure that NTP is configured on all swarm nodes. For more information, see [System Requirements for High Availability](#).
3. 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
```

4. 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>
```

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

```
docker node ls
```

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

```
for i in $(ls -l /opt/iservices/images/); do docker load -i
/opt/iservices/images/$i; done
```

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 and couchbase-worker2 section. deploy > replicas on the workers should be set to 0:

```
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 remaining db_host variables (workers, scheduler, api, services) in docker-compose should be in the following format:

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

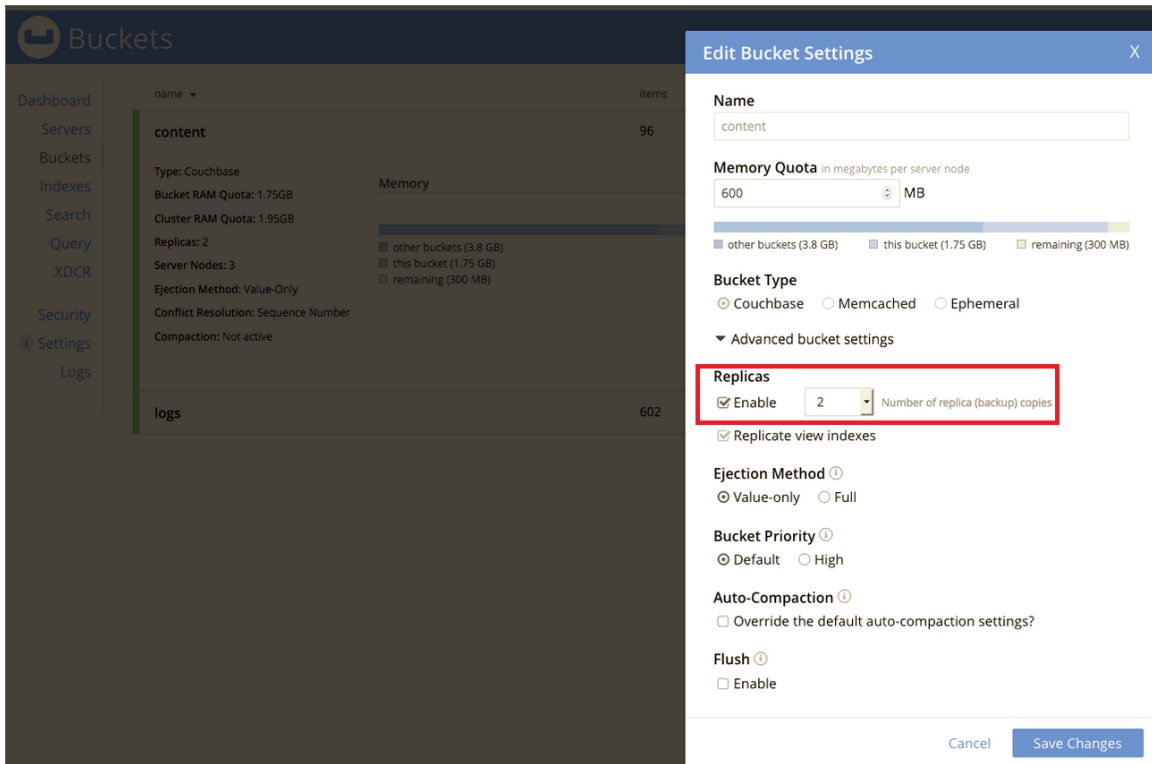
5. If you are using the override file, run the **compose-override.sh** script to generate the docker-compose.yml file.
6. Deploy the stack with only the Couchbase node by editing the replicas on couchbase-worker to 1 and running the following command:

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

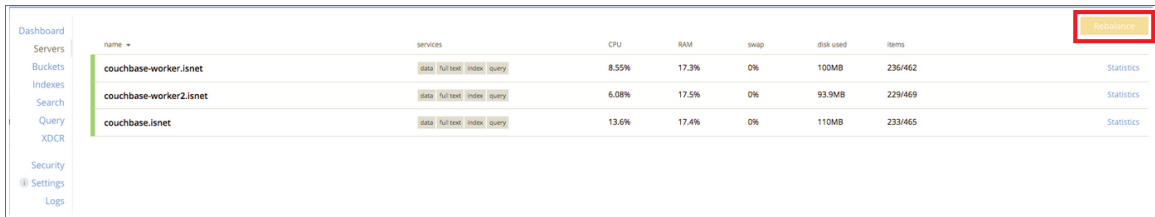
7. After the two-node Couchbase cluster has been successfully deployed and the secondary indexes are successfully added, edit the replicas on couchbase-worker2 to 1 and run the following command:

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

- Set the replicas in the `docker-compose-override.yml` file as well.
- 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]**:



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



Troubleshooting Issues with Couchbase Indexes

ScienceLogic recommends that every time you add a new node to the Couchbase cluster, you should check to make sure that the Couchbase indexes were successfully created on that node. If the first rebalance fails on the initial join of the node into the cluster, the indexes might not get created properly.

To ensure that the indexes were created and were correctly populated, complete one of the following procedures:

Option 1: Manually create the secondary index. For detailed steps, see [Recreating all indexes on a particular node](#).

Option 2: Remove the couchbase-worker service and add the node again:

1. Run the following command:

```
docker service rm iservices_couchbase-worker
```

2. Remove the node from the Couchbase cluster (failover from the user interface).
3. After failover, rebalance to completely remove the node.
4. Log in to the node system where you are trying to deploy the Couchbase worker service and run the following command (this command essentially turns the database into a fresh database that has never been started):

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

5. Run the following command to re-create the iservices_couchbase-worker:

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

6. Wait for the rebalance to complete. If the rebalance completes successfully, the indexes are created.

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.isnet,couchbase-worker2.isnet,couchbase-worker.isnet

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

  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:1.7.0
    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
environment:
  TYPE: 'WORKER'
  AUTO_REBALANCE: 'true'
  db_host: 'couchbase'
depends_on:
  - couchbase

couchbase-worker2:
  image: sciencelogic/is-couchbase:latest
  container_name: couchbase-worker2
  volumes:
    - "/var/data/couchbase:/opt/couchbase/var"
  deploy:
    replicas: 0
    placement:
      constraints:
        - node.hostname == <swarm node hostname>
networks:
  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:latest
  hostname: rabbit_node2.isnet
  volumes:
    - "rabbitdb2:/var/lib/rabbitmq2"
  networks:
    isnet:
      aliases:
        - rabbit
        - rabbit_node2.isnet
  deploy:
    placement:
      constraints:
        - node.hostname == <swarm node hostname>

rabbitmq3:
  image: sciencelogic/is-rabbit:latest
  hostname: rabbit_node3.isnet
  volumes:
    - "rabbitdb3:/var/lib/rabbitmq3"
  networks:
    isnet:
      aliases:
        - rabbit
        - rabbit_node3.isnet
  deploy:
    placement:
      constraints:
        - node.hostname == <swarm node hostname>

contentapi:
  environment:
    db_host: 'couchbase.isnet,couchbase-worker.isnet,couchbase-worker2.isnet'
volumes:
  rabbitdb:
  rabbitdb2:
  rabbitdb3:
  devpi:

```

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 (<https://<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>
```

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`

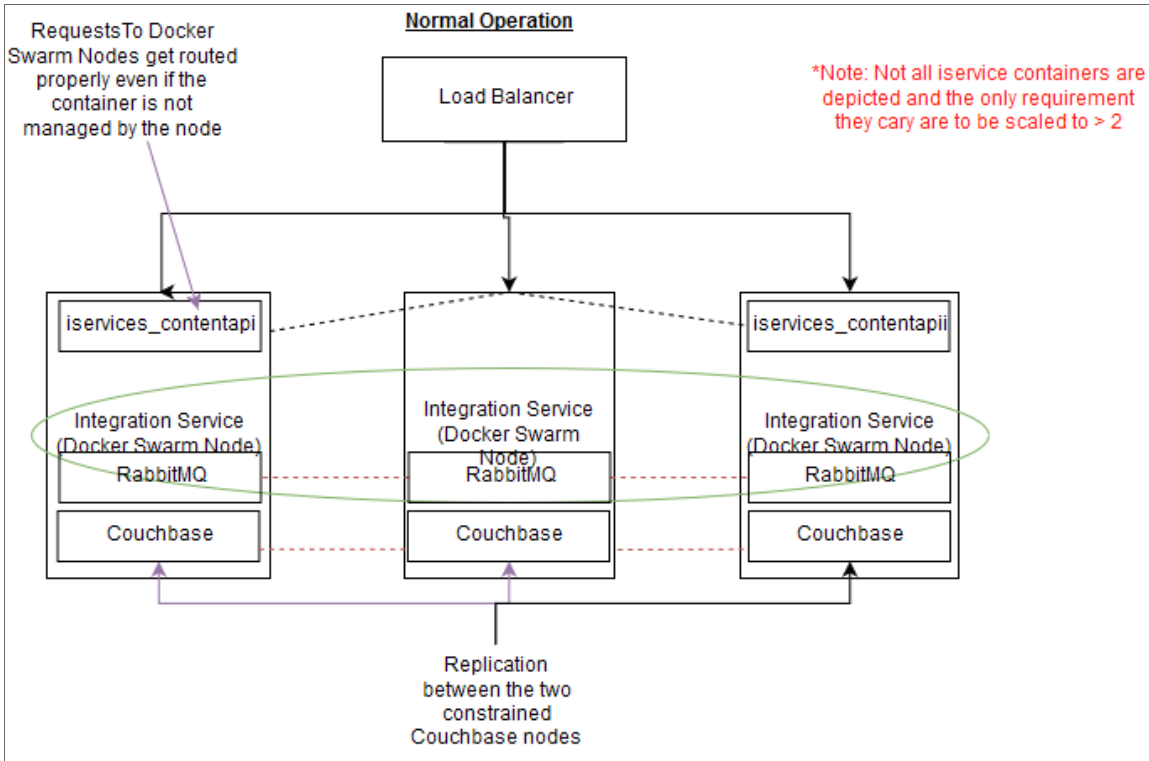
3. If the Docker Swarm node was restored and not rebuilt, remove files from the old container:

```
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:



Exposing Additional Couchbase Cluster Node Management Interfaces over TLS

The `is_gui` container acts as a reverse proxy to the internal services and their individual management interfaces. This container is configured in `/etc/nginx/conf.d/default.conf`.

To expose the management interfaces of additional Couchbase nodes within a cluster:

1. Copy the configuration from the gui container:

```
docker cp <container id>:/etc/nginx/conf.d/default.conf ./
```

2. Edit the configuration to include the desired services:

```
server {
    listen 8092 ssl;
    server_name couchbase-worker;

    location / {
        proxy_pass https://couchbase-worker.isnet:8091/;
        proxy_pass_header Server;
        proxy_pass_header Cache-Control;
        proxy_pass_header Content-Length;
    }
}
```

```

    proxy_pass_header Connection;
    proxy_pass_header Pragma;
    proxy_pass_header ns-server-ui;
    proxy_pass_header invalid-auth-response;
}
ssl_certificate /etc/iservices/is_cert.pem;
ssl_certificate_key /etc/iservices/is_key.pem;
ssl_protocols TLSv1.2;
ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-
SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-
SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256';
ssl_prefer_server_ciphers on;
ssl_session_cache shared:SSL:20m;
ssl_session_timeout 180m;
add_header Strict-Transport-Security "max-age=31536000" always;
}

```

3. Create the following Dockerfile:

```

FROM sciencelogic/is_gui
COPY ./default.conf /etc/nginx/conf.d/default.conf

```

4. Build the container with the new configurations:

```

docker build -t <customer>/is_gui:<is version>-1 -f Dockerfile .

```

5. Add the image name to the **is_gui** section in the **docker-compose-override.yml** file, and do a docker stack deploy to enable the new is_gui container.

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

```

Known Issues

The following section describes the known issues you might encounter with the High Availability solution and how to address those issues.

Docker Network Alias is incorrect

If you experience issues with the `iservices_contentapi` container, the Alias IP might be incorrect.

To address this issue, run the following commands on the relevant node:

```
docker service scale iservices_contentapi=0
docker service scale iservices_contentapi=1 (or another number as needed)
```

NOTE: This issue was addressed in the **docker-ce 18.06** version of Docker, which is required for version 1.8.0 of the Integration Service.

Docker container on last swarm node cannot communicate with other swarm nodes

This is an issue with the Encapsulating Security Payload (ESP) protocol not being enabled in `firewalld`. You can enable the ESP protocol with the `firewalld docker-swarm` script.

To address this issue, add the following firewall rule to each node:

```
firewall-cmd --add-protocol=esp --permanant
firewall-cmd --reload
```

Couchbase service does not start, remains hung at `nc -z localhost`

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

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

Couchbase-worker fails 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 Integration Service user interface fails to start after a manual failover of the swarm node

To address this issue, run the following commands on the relevant node:

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

The Integration Service user interface returns 504 errors

Ensure that your Integration Service systems have been updated with `yum upgrade`.

NTP should be used, and all node times should be in sync

If all nodes time are not in sync, you might experience issues with the `iservices_steprunners`.

The following is an example of a Docker Swarm error caused by the time not being in sync:

```
Error response from daemon: certificate (1 - 2v4umws4pxag6kbxae1wfl3vf) not valid
before Fri, 30 Nov 2018 13:47:00 UTC, and it is currently Fri, 30 Nov 2018 06:41:24
UTC: x509: certificate has expired or is not yet valid
```

For more information, see [System Requirements for High Availability](#).

Example Logs from Flower

```
iservices_flower.1.jg6glaf298d2@is-scale-05 | [W 181023 20:17:40 state:113]
Substantial drift from celery@1ee384863e37 may mean clocks are out of sync. Current
drift is iservices_flower.1.jg6glaf298d2@is-scale-05 | 18 seconds. [orig: 2018-10-23
20:17:40.090473 recv: 2018-10-23 20:17:58.486666]
```

Managing Integration Applications

Overview

This chapter describes how to use the Integration Service **[Integrations]** tab to run and schedule integration applications.

This chapter covers the following topics:

<i>Viewing the List of Integration Applications</i>	63
<i>Editing an Integration Application</i>	64
<i>Default Steps</i>	65
<i>Working with an Integration Application</i>	66
<i>Configuring an Integration Application</i>	67
<i>Running or Stopping an Integration Application</i>	68
<i>Viewing Previous Runs of an Integration Application</i>	69
<i>Scheduling an Integration Application</i>	72
<i>Viewing a Report for an Integration Application</i>	76
<i>Viewing System Diagnostics</i>	78
<i>Backing up Data for Disaster Recovery</i>	79
<i>Creating a Backup</i>	80
<i>Restoring a Backup</i>	83

Viewing the List of Integration Applications

The **[Integrations]** tab provides a list of the integration applications on your Integration Service system. From this page you can view, run, and schedule integration applications:

INTEGRATION NAME	VERSION	EDITED (UTC-4)	LAST RUN (UTC-4)	SCHEDULED
Cache ServiceNow CIs and SL1 Device Classes	2.0.1	21 Mar, 2019 15:06:01	21 Mar, 2019 15:13:45	Run Now Schedule
Cache ServiceNow CI entries	1.2.0	21 Mar, 2019 01:26:08	n/a	Run Now Schedule
Clear ServiceNow Interface Cache	1.2.0	21 Mar, 2019 15:06:06	n/a	Run Now Schedule
Sync Devices from SL1 to ServiceNow	2.1.0	21 Mar, 2019 15:05:59	21 Mar, 2019 15:13:34	Run Now Schedule
Update ServiceNow Incident when SL1 Event is Acknowledged	2.0.0	21 Mar, 2019 15:06:03	21 Mar, 2019 15:19:00	Run Now Schedule
Update ServiceNow Incident when SL1 Event is Cleared	2.0.0	21 Mar, 2019 15:05:59	21 Mar, 2019 15:17:19	Run Now Schedule
Generate Required CI Relations for ServiceNow	2.0.0	21 Mar, 2019 15:06:00	n/a	Run Now Schedule
Create or Update ServiceNow Incident from SL1 Event	2.0.0	21 Mar, 2019 15:06:02	21 Mar, 2019 15:18:40	Run Now Schedule
Template App	1.1.0	n/a	n/a	Run Now Schedule
Integration Service Backup	1.3.0	21 Mar, 2019 01:26:10	n/a	Run Now Schedule
Integration Service Restore	1.3.0	21 Mar, 2019 01:26:10	n/a	Run Now Schedule
Integration Service System Diagnostics	1.0.1	21 Mar, 2019 01:26:07	n/a	Run Now Schedule
Sync Discovery Session Status from SL1 to ServiceNow	2.0.0	21 Mar, 2019 15:06:01	n/a	Run Now Schedule
Sync Discovery Requirements	2.0.0	21 Mar, 2019 15:06:03	n/a	Run Now Schedule

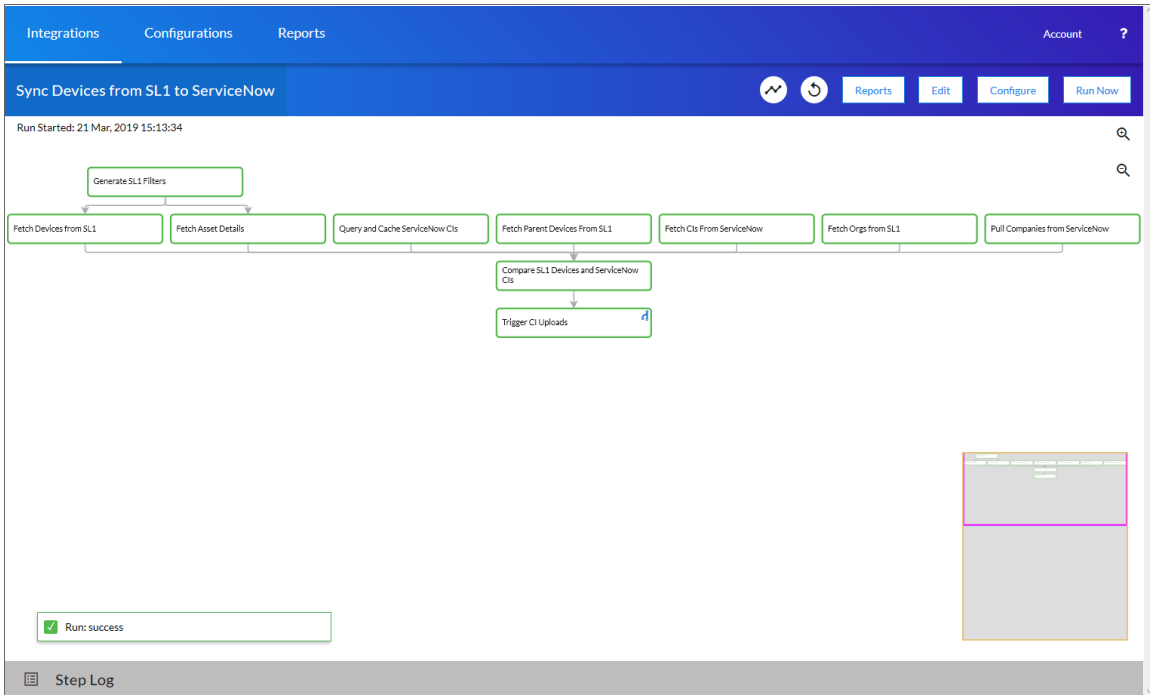
In the **Last Run** column, you can view the current status of an integration application:

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.


Some of the integration applications on the **[Integrations]** tab of the Integration Service user interface are *internal* applications that you should not run directly. Instead, other "parent" integration applications run these internal applications. To view the internal integration applications, click the Filter icon () at the top right of the **[Integrations]** tab and select *Show Hidden Integrations*. Internal integration applications are hidden by default.



Editing an Integration Application

When you click the name of an integration application on the [Integrations] tab, an **Integration Application** 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 past runs, reports for the integration application, and edit, configure, or run the integration application. Click the **Zoom** icons ( , ) to change the size of the flowchart of steps.

In the bottom left-hand corner of the page, you can view the status of the application. In the example above, the status is "Application loaded". Below the status is the **Step Log** pane, which displays the logs from a step that 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. Using an API tool like Postman or cURL, use the API `GET /steps` to download the steps:

URL_for_your_Integration_Service_system/api/v1/steps/step_name

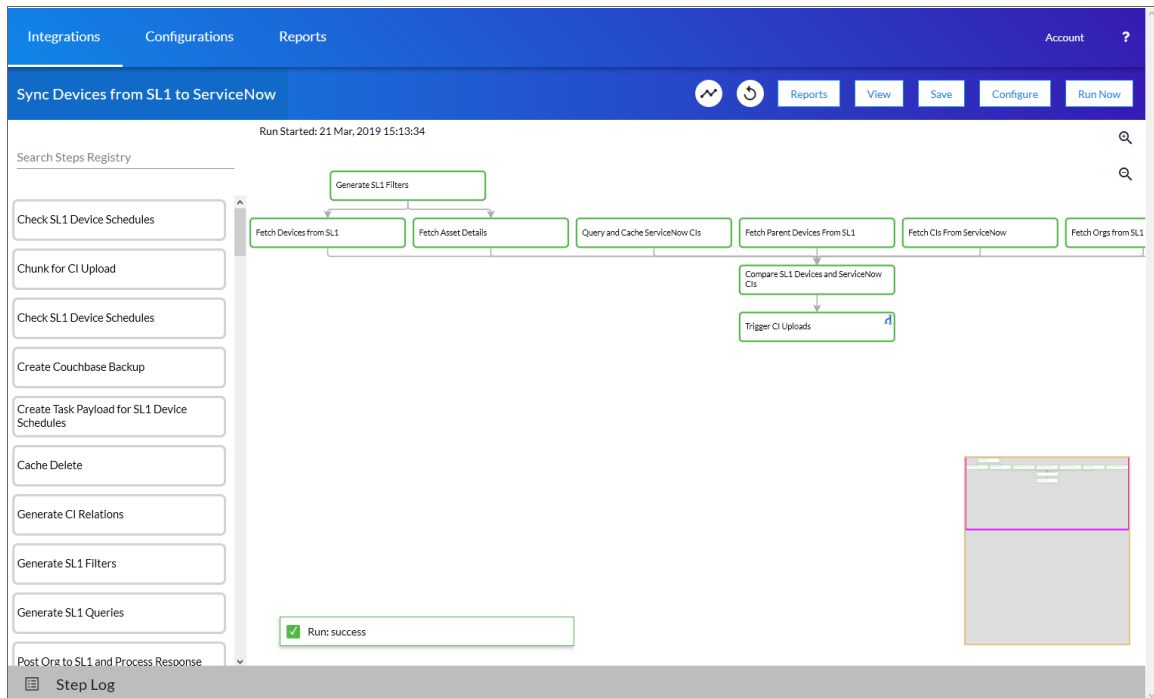
where:

- *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.

Working with an Integration Application

To work with an integration application:

1. From the **Integration Application** 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** 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.

You can "align" the configuration you want to use with an integration application from the **Configuration** pane for that application integration. 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:

Sync Devices from SL1 to ServiceNow [Cancel] [Save]

Align configuration and save

Configuration
scopedapp-prod-config

sl1_hostname 10.2.11.43 \${config.sl1 host}	snow_hostname ven01056.service-no \${config.snow host}	sl1_user em7admin \${config.sl1 user}	snow_user SL1User \${config.snow user}
sl1_password ●●●●●●●●●●	snow_password ●●●●●●●●●●	sl1_db_user root \${config.sl1 db user}	sl1_db_password ●●●●●●●●●●
region QARegion4368 \${config.region}			

mappings

cmdb_ci_linux_server maps to: Search options
Virtual Device | Windows Services

cmdb_ci_server_hardware maps to: Search options
Ping | ICMP

[Add Mapping]

Domain_Separation

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

- 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. Press **[Enter]** after editing each attribute mapping to make sure your changes are saved.

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

- 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. Press **[Enter]** after editing an item to make sure your changes are saved.
- As needed, edit the other configuration values for the application. Press **[Enter]** after editing an item to make sure your changes are saved.

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.

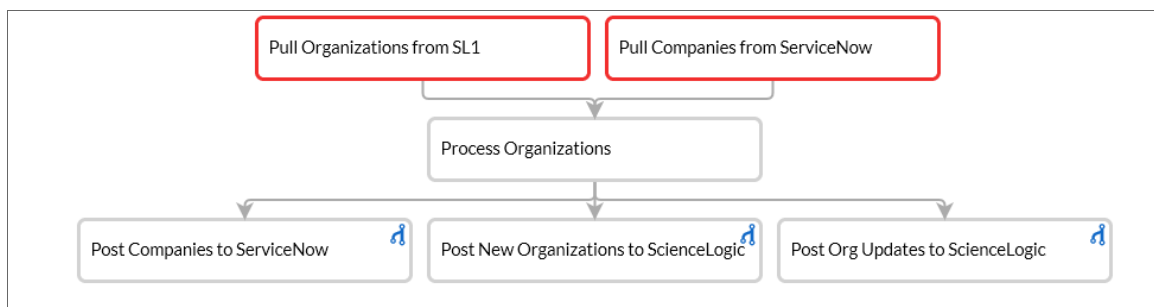
- When you are finished, click the **[Save]** button. If needed, click anywhere off of the **Configuration** pane to close the pane.

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.

Running or Stopping an Integration Application


To run an integration application:

- Click the **[Run Now]** button from the **Integrations** window or the individual application's window.
- 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.


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



- 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".
- 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.

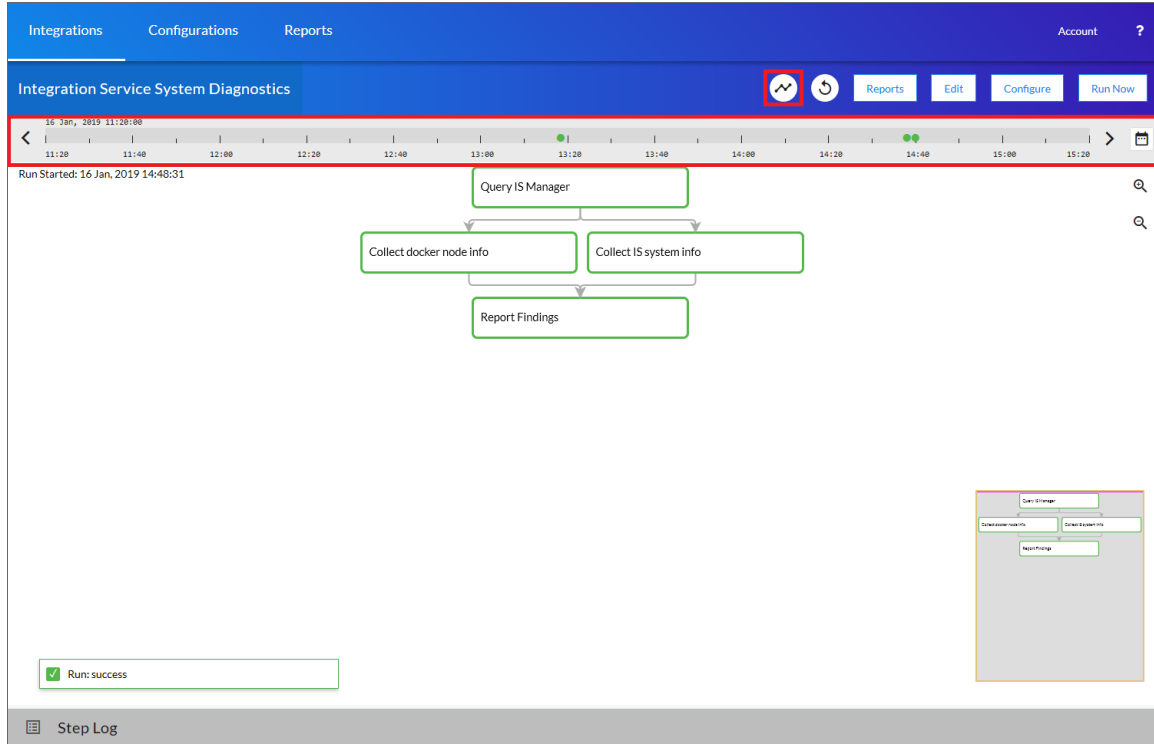
Viewing Previous Runs of an Integration Application

When you select an integration application from the Integrations tab, the **Integration Application Editor** page appears for that integration application. The **Integration Application Editor** page contains a Timeline feature that displays a history of previous runs of that integration application.

Also, you can click the **Replay** icon () to run the same integration application again, such as when an integration application failed.

To view and filter the Timeline:

1. From an **Integration Application** page, click the **Timeline** icon (📅). The Timeline displays above the steps for that integration application:

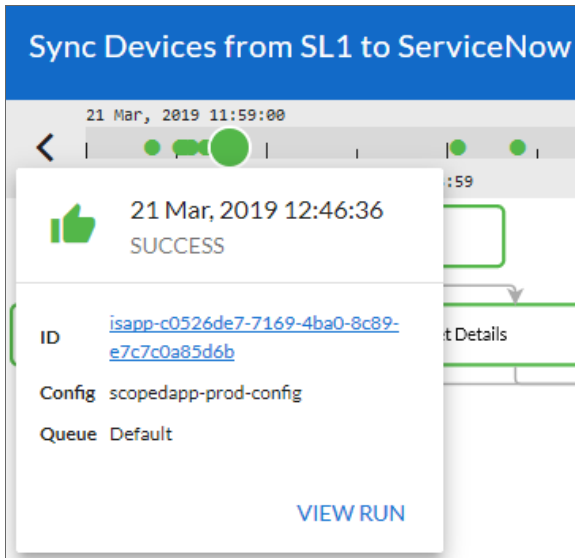


2. The default view for the Timeline shows the last two hours of runs for that integration application. The image above shows the last four hours of runs. Use the left arrow icon (⏪) and the right arrow icon (⏩) to move through the Timeline in 15-minute increments:



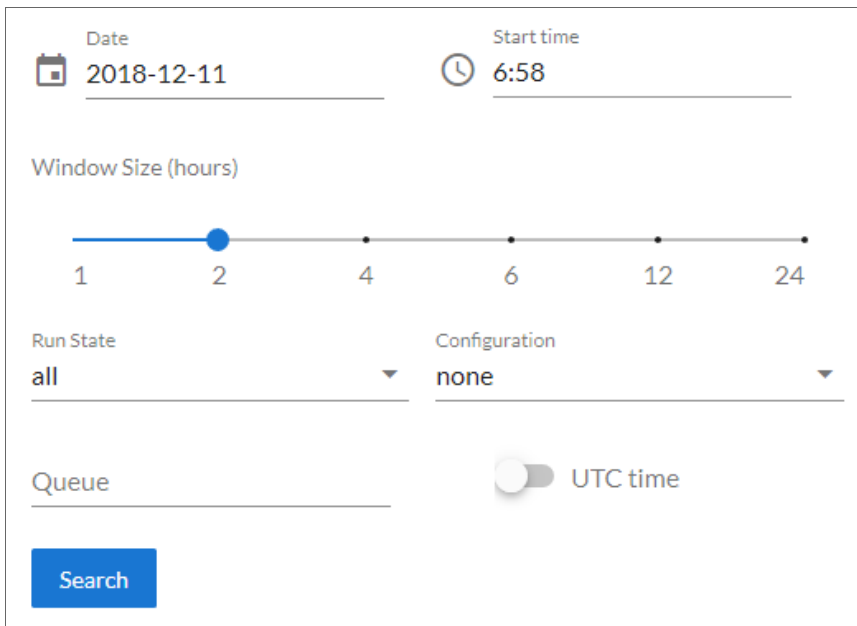
NOTE: The Timeline displays colored dots at a specific time that represent the last time this integration application was run. A green icon means a run was successful, and a red icon means that a run failed.

3. You can hover over or click an icon for a run on the Timeline to view a pop-up window that displays the run ID and the configuration and queue used for that run:



TIP: Click the link for the run ID or click **VIEW RUN** to open the **Integration Application Editor** page for that specific run of the integration application. On that page, you can select a step and open the **Step Log** to view any issues.


4. Click the Filter icon (📅) to filter or search the list of previous runs for this integration application. A **Filter** window appears:



5. Edit the following fields on the **Filter** window as needed:

- **Date.** The date for the history of previous runs you want to view. Click the field to open a pop-up calendar.
- **Start Time.** The starting time for the history, using local time instead of UTC time. Click the field to open a pop-up time selector.
- **Window Size.** The length of the history, in hours. The default history view for the Timeline is two hours.
- **Run State.** Select the type of previous runs you want to view. Your options include *all*, *success*, *failure*, and *pending*. The default is *all*.
- **Configuration.** Select a configuration file to filter for application runs using that configuration only.
- **Queue.** Type a queue name to filter for application runs that use that queue.
- **UTC Time.** Select UTC if you do not want to use local time. The Schedule feature uses UTC time.

6. Click the **[Search]** button to run the filter or search.

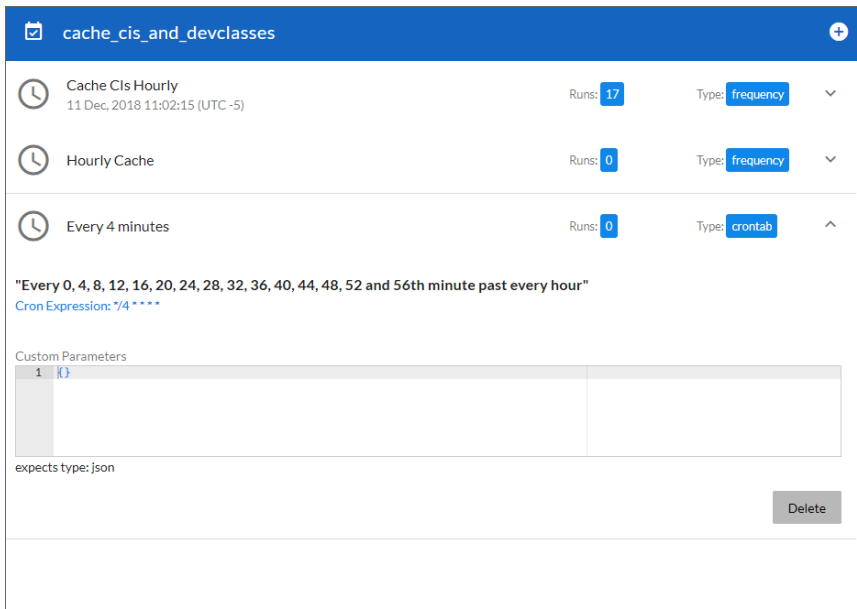
TIP: If the Timeline is open and you want to close it, click the Timeline icon ().

Scheduling an Integration Application

You can create one or more schedules for a single integration application in the Integration Service user interface. When creating the schedule, you can specify the queue and the configuration file for that 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 initial **Schedule** window appears, displaying any existing schedules for that application:



NOTE: If you set up a schedule using a cron expression, the details of that schedule display in a more readable format in this list. For example, if you set up a cron expression of `*/4 * * * *`, the schedule on this window includes the cron expression along with an explanation of that expression: "Every 0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, and 56th minute past every hour".

2. Click the + icon to create a schedule. A blank **Schedule** window appears:

cache_cis_and_devclasses

Schedule Name
Schedule Name

Switch to Cron Expression

Frequency secs

Custom Parameters

```
1 { }  
2 { }  
3 { }
```

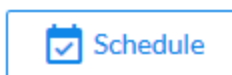
expects type: json

Save Schedule

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


- **Schedule Name.** Type a name for the schedule.
- **Cron expression.** Select this option to schedule the integration using a cron expression. If you select this option, you can create complicated schedules based on minutes, hours, the day of the month, the month, and the day of the week. As you update the cron expression, the **Schedule** window displays the results of the expression in more readable language, such as *Expression: "Every 0 and 30th minute past every hour on the 1 and 31st of every month"*, based on `*/30 * * /30 * *`.
- **Frequency in seconds.** Type the number of seconds per interval that you want to run the integration.
- **Custom Parameters.** Type any JSON parameters you want to use for this schedule, such as information about a configuration file or mappings.

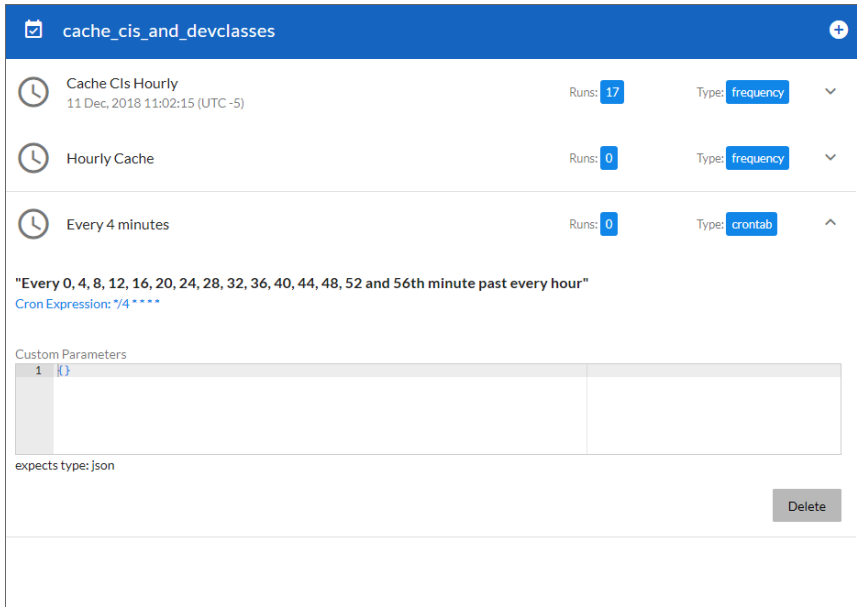
4. Click [**Save Schedule**]. The schedule is added to the list of schedules on the initial **Schedule** window. Also, on the [**Integrations**] tab, the word "Scheduled" appears in the **Scheduled** column for this integration application, and the [**Schedule**] button contains a check mark:



NOTE: After you create a schedule, it continues to run until you delete it. Also, you cannot edit an existing schedule, but you can delete it and create a similar schedule if needed.

To view or delete an existing schedule:

1. On the **[Integrations]** tab, click the **[Schedule]** button for the integration application that contains a schedule you want to delete. The initial **Schedule** window appears.
2. Click the down arrow icon () to view the details of an existing schedule:

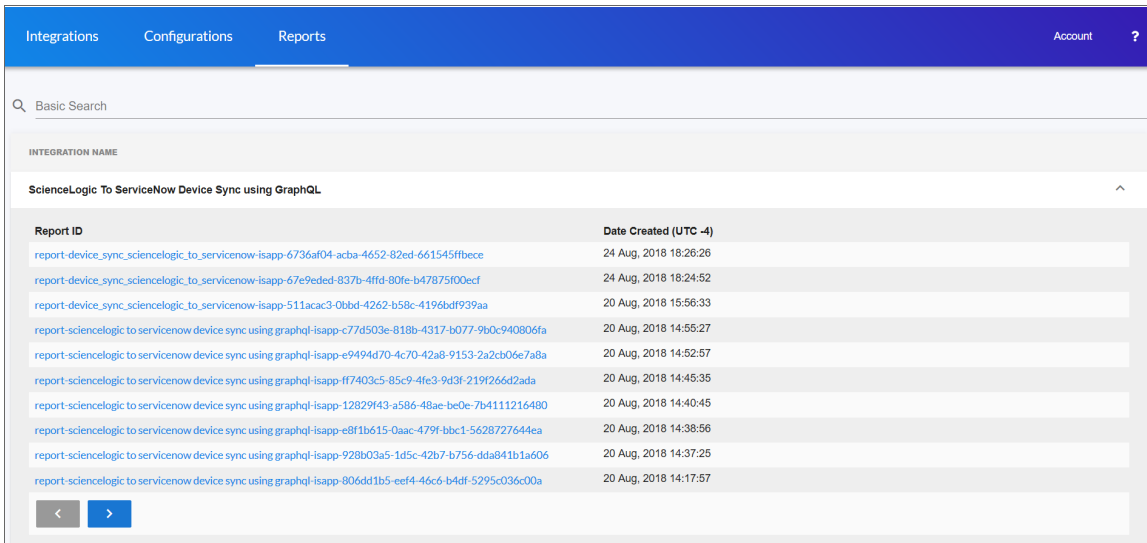


3. To delete the selected schedule, click the **[Delete]** button. The schedule is removed.

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.



The screenshot shows the 'Reports' tab in the Integration Service user interface. The navigation bar includes 'Integrations', 'Configurations', and 'Reports', with 'Reports' selected. A search bar is visible at the top. Below the search bar, the integration name 'ScienceLogic To ServiceNow Device Sync using GraphQL' is displayed. A table lists the following reports:

Report ID	Date Created (UTC -4)
report-device_sync_sciencelogic_to_servicenow-isapp-6736af04-acba-4652-82ed-661545ffbece	24 Aug, 2018 18:26:26
report-device_sync_sciencelogic_to_servicenow-isapp-67e9eded-837b-4ffd-90fe-b47875f00ecf	24 Aug, 2018 18:24:52
report-device_sync_sciencelogic_to_servicenow-isapp-511acac3-0bbd-4262-b58c-4196bdf939aa	20 Aug, 2018 15:56:33
report-sciencelogic to servicenow device sync using graphql-isapp-c77d503e-818b-4317-b077-9b0c940806fa	20 Aug, 2018 14:55:27
report-sciencelogic to servicenow device sync using graphql-isapp-e9494d70-4c70-42a8-9153-2a2cb06e7a8a	20 Aug, 2018 14:52:57
report-sciencelogic to servicenow device sync using graphql-isapp-ff7403c5-85c9-4fe3-9d3f-219f266d2ada	20 Aug, 2018 14:45:35
report-sciencelogic to servicenow device sync using graphql-isapp-12829f43-a586-48ae-bc0e-7b4111216480	20 Aug, 2018 14:40:45
report-sciencelogic to servicenow device sync using graphql-isapp-e8f1b615-0aac-479f-bbc1-5628727644ea	20 Aug, 2018 14:38:56
report-sciencelogic to servicenow device sync using graphql-isapp-928b03a5-1d5c-42b7-b756-dda841b1a606	20 Aug, 2018 14:37:25
report-sciencelogic to servicenow device sync using graphql-isapp-806dd1b5-eeef4-46c6-b4df-5295c036c00a	20 Aug, 2018 14:17:57

NOTE: Not all integration applications generate reports. Currently, only the "Sync Devices from SL1 to ServiceNow" and "Integration Service System Diagnostics" integration applications support 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*.

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-661545f1bece				
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	ee541d60db002780f6653ecd9d96198a	Updated
Templates		35	http://192.168.32.188/em7/index.em7?exec=device_summary&did=35	13745daacbcc6300b2ed73568c96190f	Updated
PSDev		38	http://192.168.32.188/em7/index.em7?exec=device_summary&did=38	47541d60db002780f6653ecd9d9619b5	Updated
Support		29	http://192.168.32.188/em7/index.em7?exec=device_summary&did=29	2e541d60db002780f6653ecd9d961999	Updated
CloudOps		11930	http://192.168.32.188/em7/index.em7?exec=device_summary&did=11930	d1649d60db002780f6653ecd9d961963	Updated

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

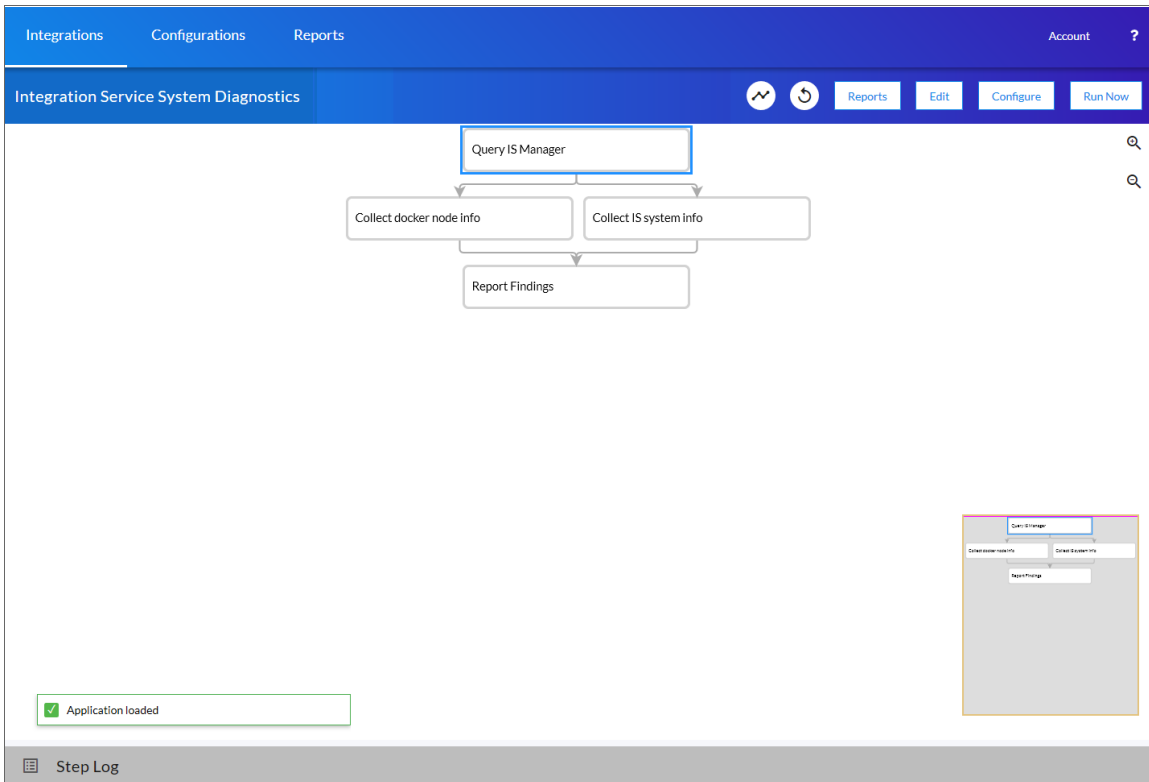
TIP: From the detail page for the integration application, click the **[Reports]** button to return to the **[Reports]** tab.

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 System Diagnostics

The "Integration Service System Diagnostics" integration application lets you view platform diagnostics for the Integration Service. You can use the information displayed in these diagnostics to help you troubleshoot issues with the different tools used by the Integration Service.

To generate the report, select the "Integration Service System Diagnostics" integration application from the **[Integrations]** tab and click the **[Run Now]** button:



TIP: The "IS - System Diagnostic Configuration Example" configuration file contains the structure needed for the "Integration Service System Diagnostics" integration application. You can use this configuration as a template when running this integration application.

Running the "Integration Service System Diagnostics" integration application generates a report that you can access on the **[Reports]** tab:

The screenshot shows the 'is_system_diagnostics' report in the Integration Service interface. The report is divided into three main sections:

- Details:** A table showing application information:

Details	
Application Name	is_system_diagnostics
Application ID	IS_System_Diagnostics
Created (UTC -5)	14 Jan, 2019 16:32:59
- IS Integrations:** A table showing integration details:

Device Mappings ↑	Incident integration run count	Schedule
[{"cmdb_ci_storage_pool_member": [{"Microcom Access Integrator Dual PRI Engin"}, {"cmdb_ci_linux_server": [{"Microcom Access Integrator Dual PRI Engin"}]}]	0	[{"schedule": [{"schedule_info": [{"run_every": 3600.0}, {"schedule_type": "frequency"}], "params": [{"entry_id": "Cache Cls Hourly"}, {"total_runs": 766}, {"last_run": [{"href": "/api/v1/tasks/isapp-dc5034fd-9ce1-4b15-a9d5-e35de9b04a5"}, {"start_time": 1547492689}], "application_id": "cache_cls_and_devclasses"}]}]
- Node 10_2_11_22 System:** A table showing system information:

cpu ↑	docker version	is rpm version	kernel version
processor : 0 vendor_id : GenuineIntel cpu family : 6 model : 45 model name : Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz stepping : 2 microcode : 0x3d cpu MHz : 2299.998 cache size : 25600 KB physical id : 0 siblings : 1 core id : 0 cpu cores : 1 apicid : 0 initial apicid : 0 fpu : yes fpu_exception : yes cpuid level : 13 wp : yes flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts mmx fxsr sse sse2 ss syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts nopl xtopology tsc_reliable nonstop_tsc pni pdm lqtd sse3 cx16 pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx hypervisor lahf_lm tsc_adjust lbpb ibrs stibp arat spec_ctrl	Docker version 18.06.1-ce, build e68fc7a	Installed Packages Name: sl1-integration-services Arch: x86_64 Version: 1.8.0 Release: 1 Size: 3.4 G Repo: Installed Summary: ScienceLogic Platform Integration Scripts and services for: Integration Services URL: http://www.sciencelogic.com License: Copyright 2018, ScienceLogic Inc. All Rights Reserved. Description: This package installs the ScienceLogic platform scripts required:	Linux is22 3.10.0-862.3.2.el7.x86_64 #1 SMP Mon May 21 17:56:51 PDT 2018 x86_64 x86_64 x86_64 GNU/Linux

This diagnostic report displays overall Integration Service settings, such as the Integration Service version, Docker version, kernel version, hostname, cluster settings, scheduled applications, CPU and memory statistics, installation date, and cache information.

TIP: If you are using a specific integration application that you want to monitor with the "Integration Service System Diagnostics" integration application, click the **[Configure]** button and type the name of that integration application in the *incident_create_app* field of the **Configuration** pane.

Backing up Data for Disaster Recovery

You can use a Disaster Recovery option that enables the Integration Service to back up and recover data in the Couchbase database.

This option uses the "Integration Service Backup" integration application in the Integration Service user interface to create a backup file and send that file using secure copy protocol (SCP) to a destination system. You can then use the "Integration Service Restore" integration application get a backup file from the remote system and restore its content.

Creating a Backup

To create a backup:

1. To add the relevant configuration information, go to the **[Configurations]** tab and click the **Plus** icon (+) to create a new configuration, or click the **[Edit]** button to update an existing configuration:

Create a new configuration

Close Save

Version
1.0

Author
admin

Friendly Name
Backup Config

Description
Used for backing up Couchbase

Configuration Data

```
1 [
2   {
3     "encrypted": false,
4     "name": "backup_destination",
5     "value": "/tmp"
6   },
7   {
8     "encrypted": false,
9     "name": "remote_host",
10    "value": "10.2.11.105"
11  },
12  {
13    "encrypted": false,
14    "name": "remote_user",
15    "value": "isadmin"
16  },
17  {
18    "encrypted": true,
19    "name": "remote_password",
20    "value": "05RD15SYmhy6pM5neLitrUQhU47ZWr/W2WzDvNPB12g="
21  },
22  {
23    "encrypted": false,
24    "name": "remote_destination",
25    "value": "/tmp"
26  }
27 ]
28 ]
```


2. In the configuration file, provide values for the following fields:
 - **backup_destination**. The location where the backup file is created.
 - **remote_host**. The hostname for the remote location where you want to send the backup file via SCP from the *backup_destination* location.
 - **remote_user**. The user login for the remote location.
 - **remote_password**. The user password for the remote location. Encrypt this value.
 - **remote_destination**. The remote location where the integration application will send the backup file.
3. Click the **[Save]** button to save the new or updated configuration file.
4. Go to the **[Integrations]** tab and select the "Integration Service Backup" integration application. The **IS Backup Integration Editor** page appears.
5. To change the configuration file used by this integration application, click the **[Configure]** button. The **Configuration** pane appears:

Integration Service Backup
✕

Modify configuration and save.

Configuration
is-system-diagnostic-configuration-example

<p>backup_destination</p> <p>/tmp</p> <p><small>\$(config.backup_destination)</small></p>	<p>remote_host</p> <p>10.2.11.65</p> <p><small>\$(config.remote_host)</small></p>	<p>remote_user</p> <p>root</p> <p><small>\$(config.remote_user)</small></p>
<p>remote_password</p> <p>●●●●●●●●●●●●●●●●</p> <p><small>\$(config.remote_password)</small></p>	<p>remote_destination</p> <p>/tmp</p> <p><small>\$(config.remote_destination)</small></p>	<p><input checked="" type="checkbox"/> data_only</p>
<p><input checked="" type="checkbox"/> compress</p>	<p><input type="checkbox"/> single_node</p>	<p><small>cluster_node</small></p> <p>couchbase.isnet</p>

bucket

logs

document_key

all

Save

6. In the configuration file, provide values for the following fields:

- **Configuration.** Select a configuration to align with this integration applications.
- **data_only.** Select this option if you only want to restore bucket data.
- **compress.** Select this option to compress backups using Gzip.
- **single_node.** Select this option if you want to back up from a single node in the cluster. Specify the node in the **cluster_node** field.
- **cluster_node.** Specify the node from which you want to make the backup (for a single-node backup only).
- **bucket.** Select which bucket in Couchbase you want to back up. Your options include all buckets, content-only buckets, or logs. The default is *all*.
- **document_key:** Select whether you want to back up all record or CI and device cache records.

NOTE: The options you select affect the name of the backup file that this integration application generates. For example, **is_couchbase_backup-data_only-2019-04-01T185527Z.tar** is a uncompressed data only backup, while **is_couchbase_backup-data_only-cache-logs-couchbase.isnet-2019-04-01T185937Z.tar.gz** is a compressed data-only backup of the CI and device cache from the *couchbase.isnet* node in a cluster.

7. Click the **[Save]** button.
8. On the **IS Backup Integration Editor** page, click the **[Run Now]** button. When the application completes, a file named "is_couchbase_backup- <date> .tar" is added to the remote server in the specified remote backup destination.
9. To ensure that the backup was created, click to open the **Step Log** section and look for entries related to backup. Make a note of the of the backup file name, which you will need when you run the "Integration Service Restore" integration application:



TIP: You can schedule the "Integration Service Backup" integration application to run on a regular basis, or you can run the application as needed. To schedule the application, click the **[Schedule]** button for the "Integration Service Backup" application on the **[Integrations]** tab. For more information, see [Scheduling an Integration Application](#).

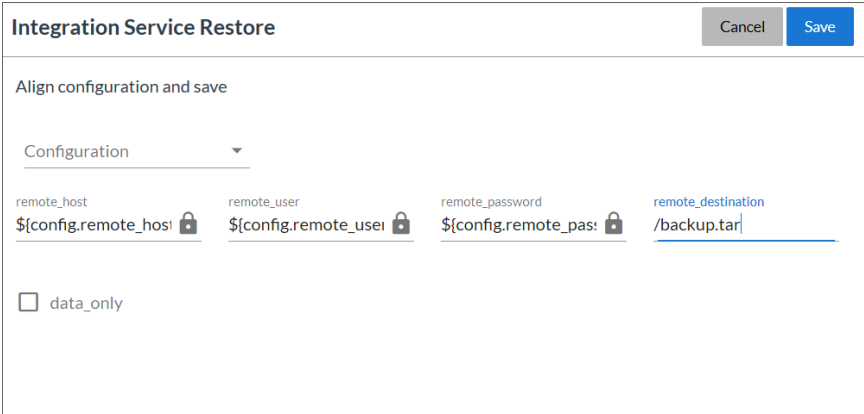
Restoring a Backup

After you have created a backup using the "Integration Service Backup" integration application in the Integration Service user interface, you can use the "Integration Service Restore" integration application to restore that file.

NOTE: Do not restore the Integration Service backup to a system that uses a different encryption key.

To restore a backup:

1. In the Integration Service user interface, go to the **[Integrations]** tab and select the **Integration Service Restore** integration application. The **Integration Service Restore** page appears.
2. To update the configuration file used by this integration application, click the **[Configure]** button. The **Configuration** pane appears:



The screenshot shows the "Integration Service Restore" configuration pane. At the top right, there are "Cancel" and "Save" buttons. Below the title, the text "Align configuration and save" is displayed. A "Configuration" dropdown menu is present. Below this, four configuration fields are shown: "remote_host" with value "\${config.remote_host}", "remote_user" with value "\${config.remote_user}", "remote_password" with value "\${config.remote_password}", and "remote_destination" with value "/backup.tar". Each of the first three fields has a lock icon. At the bottom, there is a checkbox labeled "data_only".

3. In the configuration file, provide values for the following fields:
 - **Configuration**. Select a configuration to align with this integration applications.
 - **remote_destination**. Type the name of the backup file created by the "Integration Service Backup" integration application.
 - **data_only**. Select this option if you only want to restore bucket data.
4. Click the **[Save]** button.
5. On the **Integration Service Restore** page, click the **[Run Now]** button.

- To ensure that the backup was restored, click to open the **Step Log** section and look for entries related to restoring the backup:

Step Log				
6	ipaas_logger	29 Nov, 2018 12:10:41, 681	FLOW	content bucket restored
7	ipaas_logger	29 Nov, 2018 12:10:41, 693	FLOW	[#####] 100.0% (99/estimated 99 msgs)[A bucket: content, msgs transferred... : total last per sec byte : 722761 722761 1226032.5 done
8	ipaas_logger	29 Nov, 2018 12:10:42, 765	FLOW	logs bucket restored
9	ipaas_logger	29 Nov, 2018 12:10:42, 766	FLOW	[#####] 100.0% (2/estimated 2 msgs)[A bucket: logs, msgs transferred... : total last per sec byte : 1921 1921 6703.0 done
10	ipaas_logger	29 Nov, 2018 12:10:42, 770	FLOW	[is_couchbase_backup-2018-11-29T155710Z.tar', 'extract']
11	ipaas_logger	29 Nov, 2018 12:10:42, ---	FLOW	Removing folder: /tmp/restore

NOTE: After running the "Integration Service Restore" application, the "Integration Service Backup" application might display as "Run status pending". This issue occurs because at the time of the last backup from Couchbase, the logs for the "Integration Service Backup" application showed a pending state. This message is addressed during the next run, and does not cause any issues with the backup or restore processes.

Chapter

5

Managing Configurations

Overview

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

This chapter covers the following topics:

<i>What is a Configuration?</i>	86
<i>Creating a Configuration</i>	88
<i>Editing a Configuration</i>	90

What is a Configuration?


A **configuration** is a stand-alone JSON file that lives on the Integration Service system. A configuration file supplies the login credentials and other global variables that can be used by all steps and integration applications. You can add and edit configurations on the **[Configurations]** tab of the Integration Service user interface.

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 **Configurations** tab displays a list of available configurations. From this page you can create and edit configurations:

CONFIG NAME	VER	AUTHOR	MODIFIED (UTC -3)	DESCRIPTION	
IS Diagnostics Config	1.0.0	ScienceLogic	21 Dec, 2018 11:02:07	IS Diagnostics Config	+ Edit
Test Host Settings	1.0.0	ScienceLogic	10 Jan, 2019 20:52:15	A test config with host information for testing.	Edit
shep test config	1.0.0	ShepMcKee	16 Jan, 2019 14:57:13	Description for shep test config	Edit
Test Host Settings	1.0.0	ScienceLogic	10 Dec, 2018 16:26:21	A test config with host information for testing.	Edit
Aligned Integrations					
1. Integration Service Backup					
2. Integration Service Restore					
Ven1055 to SL1 with vmware tree	1.0.0	ScienceLogic	03 Jan, 2019 13:23:11	Configuration for ven1055	Edit

TIP: Click the down arrow icon () for a configuration to see which integration applications use or are "aligned" with that configuration.


For each configuration, the **[Configurations]** tab displays the following information:

- **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.

NOTE: The Integration Service includes an example configuration on the **[Configurations]** tab called "Test Host Settings" that you can use as a template.

Creating a Configuration

To create a new configuration object:

1. In the Integration Service user interface, go to the **[Configurations]** tab.
2. Click the plus icon (). The **Create a new configuration** pane appears:

Create a new configuration

Close Save

Version
1.0

Author
ScienceLogic, Inc.

Friendly Name
SL1 Credentials

Description
Credentials for SL1 system

Configuration Data

```
1- [
2-  {
3-    "encrypted": false,
4-    "name": "sl1_host",
5-    "value": "10.2.11.42"
6-  },
7-  {
8-    "encrypted": false,
9-    "name": "sl1_user",
10-   "value": "em7admin"
11-  },
12-  {
13-    "encrypted": true,
14-    "name": "sl1_password",
15-    "value": "+dqGJe1NwTyvda02EizTWjJ2uj2C1wzBzgNqVhpdTHA="
16-  }
17- ]
```

3. Complete the following fields:
 - **Version**. Version of the configuration object.
 - **Author**. User or organization that created the configuration object.
 - **Friendly Name**. Name of the configuration object.
 - **Description**. A brief description of the configuration object.

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": "+dqGJe1NwTyvda02EizTWjJ2uj2C1wzBzgNqVhpdTHA="
  }
]
```

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 5-6 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 object to resolve the variable.

Editing a Configuration

To edit an existing configuration:

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

```
1 - []
2 - {
3   "encrypted": false,
4   "name": "sl1_host",
5   "value": "10.2.11.42"
6 }
7 - {
8   "encrypted": false,
9   "name": "sl1_user",
10  "value": "em7admin"
11 }
12 - {
13  "encrypted": true,
14  "name": "sl1_password",
15  "value": "+dqGJe1NwTyvda02EizTWjJ2uj2C1wzBzgNqVhpdTHA="
16 }
17 ]]
```

expects type: json

3. 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.

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:

<i>Logging Data in the Integration Service</i>	92
<i>Local Logging</i>	92
<i>Remote Logging</i>	92
<i>Viewing Logs in Docker</i>	92
<i>Logging Configuration</i>	93
<i>Viewing the Step Logs for an Integration Application</i>	93
<i>Removing Logs on a Regular Schedule</i>	94

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.

TIP: You can use the "Timed Removal" integration application to remove logs from Couchbase on a regular schedule. On the **Configuration** pane for that integration application, specify the number of days before the next log removal.

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 the Step 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** page appears.
2. Select a step in the integration application. Required.

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

	MODULE	DATE/TIME (UTC -5)	LOG LEVEL	MESSAGE
1	ipaaS_logger	16 Jan, 2019 14:48:32, 088	FLOW	Start Query IS Manager
2	ipaaS_logger	16 Jan, 2019 14:48:32, 094	FLOW	Collecting diagnostics from primary manager node 10.2.11.22
3	ipaaS_logger	16 Jan, 2019 14:48:32, 095	FLOW	Trying to connect to 10.2.11.22 (1/3)
4	ipaaS_logger	16 Jan, 2019 14:48:32, 943	FLOW	stop Query IS Manager 0.855342149734

- Click the gray area of the **Logs** pane to close the pane.

TIP: Log information for a step is saved for the duration of the **result_expires** setting in the Integration Service system. The **result_expires** setting is defined in the **opt/iservices/scripts/docker-compose.yml** file. The default value for log expiration is 7 days (in Integration Service versions before 1.8.1, the default value for log retention was 1 day). This environment variable is set in seconds.

Removing Logs on a Regular Schedule

The "Timed Removal" integration application lets you remove logs from Couchbase on a regular schedule.

To schedule the removal of logs:

- In the Integration Service user interface, go to the **[Integrations]** tab and select the "Timed Removal" integration application.

2. Click the **[Configure]** button. The **Configuration** pane appears:



Timed Removal

Align configuration and save

Configuration

time_in_days
7

3. Complete the following fields:
 - **Configuration**. Select the relevant configuration to align with this integration application. You cannot edit fields that are populated by the configuration. Required.
 - **time_in_days**. Specify how often you want to remove the logs from Couchbase. The default is 7 days. Required.
4. Click the **[Save]** button and close the **Configuration** pane.
5. Click the **[Run Now]** button to run the "Timed Removal" integration application.

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:

<i>Interacting with the API</i>	96
<i>Available Endpoints</i>	97

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/{appName}/run. Run a single application by name with saved or provided configurations.

/applications/run. Run a single application by name. For more information, see [Querying for the State of an Integration Application](#).

/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 scheduled application integration.

/tasks/{taskId}/replay. Replay a specific integration application. Replayed integration applications run with the same application variables, configuration, and queue as the originally executed application.

/tasks/{taskId}/revoke. Revoke or terminate a specific task or integration application. If an application ID is provided, all tasks associated with that integration application are be revoked.

Querying for the State of an Integration Application

When triggering an integration application from the **applications/run** endpoint, you can query for the state of that integration application in two ways:

1. **Asynchronously**. When you POST a run of an integration application to **/applications/run**, the response is a integration status with a Task ID, such as: *isap-23233-df2f24-etc*. At any time, you can query for the current state of that task from the endpoint **/api/v1/tasks/isap-23233-df2f24-etc**. The response includes all of the steps run by the integration application, along with the status of the steps, and URL links to additional info, such as logs for each step.
2. **Synchronously**. When you POST a run of an integration application, you can tell the Integration Service to wait responding until the integration application is complete by adding the **wait** argument. For example, **/api/v1/applications/run?wait=20** will wait for 20 seconds before responding. The maximum wait time is 30 seconds. When the integration application completes, or 30 seconds has passed, the API returns the current status of the integration run. This process works the same as if you had manually queried **/api/v1/tasks/isapp-w2ef2f2f**. Please note that while the API is waiting for your integration application to complete, you are holding on to a thread. If you have multiple integration applications that run for a long period of time, do not use a synchronous query unless you have no other option. ScienceLogic recommends using an *asynchronous* query whenever possible.

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 paginated reports.

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

/schedule. Retrieve a list of all scheduled application integrations.

/steps. Retrieve a list of all steps.

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

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

REST

/tasks. Terminate all running tasks.

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

DELETE

/applications/{appName}. Delete an integration application by name.

/cache/{cache_id}. Delete a cache entry by name.

/configurations/{configName}. Delete a configuration by name.

/schedule. Delete a scheduled application integration by ID.

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

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

/steps/{stepName}. Delete a specific step by name.

Troubleshooting the Integration Service

Overview

This chapter includes troubleshooting resources, procedures, and frequently asked questions related to working with the Integration Service.

This chapter covers the following topics:

<i>Resources for Troubleshooting</i>	101
<i>Useful Integration Service Ports</i>	101
<i>Helpful Docker Commands</i>	101
Viewing Container Versions and Status	101
Restarting a Service	101
Stopping all Integration Service Services	102
Restarting Docker	102
Viewing Logs for a Specific Service	102
Clearing RabbitMQ Volume	102
Viewing the Process Status of All Services	103
Deploying Services from a Defined Docker Compose File	103
Dynamically Scaling for More Workers	103
Completely Removing Services from Running	103
<i>Helpful Couchbase Commands</i>	103
Checking the Couchbase Cache to Ensure an SL1 Device ID is Linked to a ServiceNow Sys ID	103
Clearing the Internal Integration Service Cache	104
Accessing Couchbase with the Command-line Interface	105



Useful API Commands	105
Getting Integrations from the Integration Service API	105
Creating and Retrieving Schedules with the Integration Service API	105
Diagnosis Tools	106
Identifying Why a Service or Container Failed	107
Step 1: Obtain the ID of the failed container for the service	107
Step 2: Check for any error messages or logs indicating an error	107
Step 3: Check for out of memory events	108
Troubleshooting a Cloud Deployment of the Integration Service	108
Identifying Why an Integration Application Failed	109
Determining Where an Integration Application Failed	109
Retrieving Additional Debug Information (Debug Mode)	109
General Troubleshooting Steps	110
Integration Service	111
ServiceNow	111
Frequently Asked Questions	111
What is the first thing I should do when I have an issue with my Integration Service?	111
Why do I get a "Connection refused" error when trying to communicate with Couchbase?	111
How do I remove a schedule that does not have a name?	112
Why are there client-side timeouts when communicating with Couchbase?	113
What causes a Task Soft Timeout?	113
Why are incident numbers not populated in SL1 on Incident creation in ServiceNow?	113
Why am I not getting any Incidents after disabling the firewall?	114
Why are Incidents not getting created in ServiceNow?	114
How can I point the "latest" container to my latest available images for the Integration Service?	114
How do I manually create a backup of my Integration Service system, and how do I manually restore?	114
What do I do if I get a Code 500 error when I try to access the Integration Service user interface?	115
What are some common examples of using the iscli tool?	116
How do I view a specific run of an application on IS?	116

Resources for Troubleshooting

This section contains port information for the Integration Service and troubleshooting commands for Docker, Couchbase, and the Integration Service API.

Useful Integration Service Ports

- **http://<IP of Integration Service>:8081**. Provides access to Docker Visualizer, a visualizer for Docker Swarm.
- **https://<IP of Integration Service>:8091**. Provides access to Couchbase, a NoSQL database for storage and data retrieval.
- **https://<IP of Integration Service>:15672**. Provides access to the RabbitMQ Dashboard, which you can use to monitor the service that distributes tasks to be executed by Integration Service workers.
- **https://<IP of Integration Service>/flower**. Provides access to Flower, a tool for monitoring and administrating Celery clusters.

Helpful Docker Commands

The Integration Service is a set of services that are containerized using Docker. For more information about Docker, see the [Docker tutorial](#).

Use the following Docker commands for troubleshooting and diagnosing issues with the Integration Service:

Viewing Container Versions and Status

To view the Integration Service version, SSH to your Integration Service instance and run the following command:

```
docker service ls
```

In the results, you can see the container ID, name, mode, status (see the *replicas* column), and version (see the *image* column) for all the services that make up the Integration Service:

```
root@fsunis4lab ~# docker service ls
ID                NAME                MODE                REPLICAS                IMAGE                PORTS
mm1hu35v301      iservices_gui        replicated           1/1                     repository.auto.sciencelogic.local:5000/is-gui:1.7.0      *180->80/tcp, *:443->443/tcp
40vs91ciwvh3     iservices_redis      replicated           1/1                     redis:4.0.2
jlm6h1jtmuf      iservices_flower     replicated           1/1                     repository.auto.sciencelogic.local:5000/is-worker:1.7.0  *:5555->5555/tcp
lh3pt2181zsf     iservices_scheduler  replicated           1/1                     repository.auto.sciencelogic.local:5000/is-worker:1.7.0
html1cvj6kxh     iservices_contentapi replicated           1/1                     repository.auto.sciencelogic.local:5000/is-api:1.7.0     *:8000->8000/tcp
cym8qg9uami      iservices_rabbitmq   replicated           1/1                     rabbitmq:3
k1u19h8jyfs6     iservices_visual     replicated           2/1                     docKerSamples/visualizer:latest
ycy38w8buauw     iservices_couchbase  replicated           1/1                     repository.auto.sciencelogic.local:5000/is-couchbase:1.7.0 *8081->8080/tcp, *:8091->8091/tcp, *:8092->8092/3->8093/tcp, *:8094->8094/tcp, *:11210->11210/tcp
z1bxstxoz7uf     iservices_steprunner replicated           5/5                     repository.auto.sciencelogic.local:5000/is-worker:1.7.0
```

Restarting a Service

Run the following command to restart a single service:

```
docker service update --force iservices_<service_name>
```



Stopping all Integration Service Services

Run the following command:

```
docker stack rm iservices
```

Restarting Docker

Run the following command:

```
systemctl restart docker
```

NOTE: Restarting Docker does not clear the queue.

Viewing Logs for a Specific Service

If you need to view the logs for a certain service to ensure that the service started correctly, run the following command:

```
docker service logs -f iservices_<service_name>
```

For example:

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

NOTE: *Application* logs are stored on the central database as well as on all of the Docker hosts in a clustered environment. These logs are stored at `/var/log/iservices` for both single-node or clustered environments. However, the logs on each Docker host only relate to the services running on that host. For this reason, using the Docker service logs is the best way to get logs from all hosts at once.

Clearing RabbitMQ Volume

RabbitMQ is a service that distributes tasks to be executed by Integration Service workers. This section covers how to handle potential issues with RabbitMQ.

The following error message might appear if you try to run an integration application via the API:

```
Internal error occurred: Traceback (most recent call last):\n File \"/content_api.py", line 199, in kickoff_application\n task_status = ... line 623, in _on_close\n (class_id, method_id), ConnectionError)\nInternalError: Connection.open: (541) INTERNAL_ERROR - access to vhost '/' refused for user 'guest': vhost '/' is down
```

First, verify that your services are up. If there is an issue with your RabbitMQ volume, you can clear the volume with the following commands:

```
docker service rm iservices_rabbitmq\n docker volume rm iservices_rabbitdb
```

If you get a message stating that the volume is in use, run the following command:

```
docker rm <id of container using volume>
```

Re-deploy the Integration Service by running the following command:

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

NOTE: Restarting Docker does not clear the queue, because the queue is persistent. However, clearing the queue with the commands above might result in data loss due to the tasks being removed from the queue.

Viewing the Process Status of All Services

Run the following command:

```
docker ps
```

Deploying Services from a Defined Docker Compose File

Run the following command:

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

Dynamically Scaling for More Workers

Run the following command:

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

Completely Removing Services from Running

Run the following command:

```
docker stack rm iservices
```

Helpful Couchbase Commands

Checking the Couchbase Cache to Ensure an SL1 Device ID is Linked to a ServiceNow Sys ID

You can determine how an SL1 device links to a ServiceNow CI record by using the respective device and sys IDs. You can retrieve these IDs from the Integration Service Couchbase service.

First, locate the correlation ID with the following Couchbase query:

```
select meta().id from logs where meta().id like "lookup%"
```

This query returns results similar to the following:

```
[  
  {
```

```

    "id": "lookup-ScienceLogicRegion+DEV+16"
  },
  {
    "id": "lookup-ScienceLogicRegion+DEV+17"
  }
]

```

After you locate the correlation ID, run the following query:

```
select cache_data from logs where meta().id = "lookup-ScienceLogicRegion+DEV+16"
```

This query returns the following results:

```

[
  {
    "cache_data": {
      "company": "d6406d3bdb372300c40bdec0cf9619c2",
      "domain": null,
      "snow_ci": "u_cmdb_ci_aws_service",
      "sys_id": "0c018f14dbd36300f3ac70adbf9619f7"
    }
  }
]

```

Clearing the Internal Integration Service Cache

Some integration applications pull information from SL1 or ServiceNow and store that data in a cache stored on the Integration Service. Examples of these applications include "Cache SL1 Devices" and "Cache ServiceNow CIs and SL1 Device Classes". Occasionally, these caches might get stale and might not get auto-cleared or updated in Integration Service. The following steps cover how to clear the cache to force IS to re-build that cache.

You can clear the internal Integration Service cache with the command line interface (CLI) or the Couchbase interface.

Clearing the cache with the command line interface

Locate the ID of the Couchbase container using the `docker ps` or `docker service ls` command, and then run the following commands:

```

docker exec -it <container_id> /bin/bash
cbq -u <username> -p <password> -e "https://<localhost>:8091"
delete from logs where log_type == "cache"

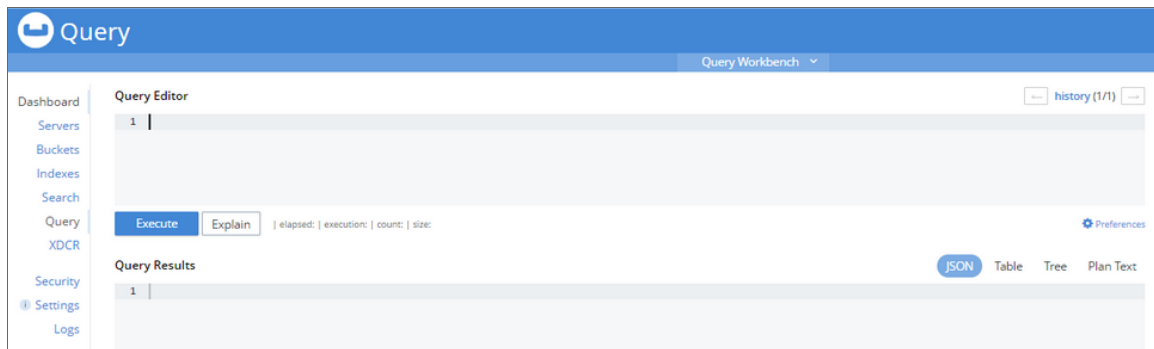
```

Clearing the cache with the Couchbase interface

1. To access the Couchbase interface, navigate to `https://<localhost>:8091`.

2. Sign in using your regular Integration Service credentials.
3. Navigate to the **[Query]** tab and run the following query in the Query Editor:

```
delete from logs where log_type == "cache"
```



Accessing Couchbase with the Command-line Interface

If you don't have access to port 8091 on your Integration Service instance, you can connect to the Couchbase container by using the command-line interface (CLI).

To access Couchbase by using the CLI, run the following commands:

```
docker exec -it <container_id> /bin/bash
cbq -u <username> -p <password> -e "https://<localhost>:8091"
```

Useful API Commands

Getting Integrations from the Integration Service API

You can use the API or cURL to retrieve the application code, which is useful when you are troubleshooting potential code-related issues. You cannot access these API endpoints with a browser, but you can request these API endpoints by using an application such as Postman:

```
https://<integration_service>/api/v1/applications/<application_name>
```

If you do not have access to Postman, you can use cURL to get the same information.

```
curl -iku <username>:<password> -H "Accept: application/json" -H "Content-Type: application/json" -X GET https://<integration_service>/api/v1/applications/<application_name>
```

Creating and Retrieving Schedules with the Integration Service API

You can define and retrieve schedules using the Integration Service API. Using Integration Service version 1.8.0 or later, you can define all of these schedules in the Integration Service user interface as well.

To create a schedule via the API, POST the following payload to the API endpoint: `https://<integration_service>/api/v1/schedule`

```

{
  "application_id": "APP_ID",
  "entry_id": "SCHEDULE_NAME",
  "params": {"a": "B"},
  "schedule": {
    "schedule_info": {
      "day_of_month": "*",
      "day_of_week": "*",
      "hour": "*",
      "minute": "*",
      "month_of_year": "*"
    },
    "schedule_type": "crontab"
  },
  "total_runs": 0
}

```

You can also specify the schedule to run on a frequency in seconds by replacing the schedule portion with the following:

```

"schedule": {
  "schedule_info": {
    "run_every": FREQUENCY_IN_SECONDS
  },
  "schedule_type": "frequency"
}

```

Diagnosis Tools

Multiple diagnosis tools exist to assist in troubleshooting issues with the Integration Service platform and the ServiceNow integration.

- **Docker PowerPack.** This PowerPack can be used to monitor the health and statistics of the Docker containers in the Integration Service. This PowerPack also monitors the Docker Swarm for any clustered deployment.
- **Flower.** This web interface tool can be found at the /flower endpoint. It provides a dashboard displaying the number of tasks in various states as well as an overview of the state of each worker. This tool shows the current number of active, processed, failed, succeeded, and retried tasks on the Integration Service platform. This tool also shows detailed information about each of the tasks that have been executed on the platform. This data includes the UUID, the state, the arguments that were passed to it, as well as the worker and the time of execution. Flower also provides a performance chart that shows the number of tasks running on each individual worker.
- **Debug Mode.** All applications can be run in "debug" mode via the Integration Service API. Running applications in debug mode may slow down the platform, but they will result in much more detailed logging information that is helpful for troubleshooting issues. For more information on running applications in Debug Mode, see [Retrieving Additional Debug Information](#).
- **Application Logs.** All applications generate a log file specific to that application. These log files can be found at /var/log/iservices and each log file will match the ID of the application. These log files combine all the log messages of all previous runs of an application up to a certain point. These log files roll over and will get auto-cleared after a certain point.

- **Step Logs.** Step logs display the log output for a specific step in the application. These step logs can be accessed via the Integration Service user interface by clicking on a step in an integration application and bringing up the **Step Log** tab. These step logs display just the log output for the latest run of that step.
- **Service Logs.** Each Docker service has its own log. These can be accessed via SSH by running the following command:

```
docker service logs -f <service_name>
```

Identifying Why a Service or Container Failed

This section outlines the troubleshooting steps necessary to determine the underlying root cause of why a service or container was restarted. For this section, we use the `iservices_redis` service as an example.

Step 1: Obtain the ID of the failed container for the service

Run the following command for the service that failed previously:

```
docker service ps --no-trunc <servicename>
```

For example:

```
docker service ps --no-trunc iservices_redis
```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE	ERROR
ORTS						
1slu2qwbte	iservices_redis.1	redis:4.0.2	is-scale-04	Running	Running 2 hours ago	
3s7s86n45skf	iservices_redis.1	redis:4.0.2	is-scale-03	Shutdown	Failed 2 hours ago	"task: non-zero exit (137)"

From the command result above, we see that one container with the id `3s7s86n45skf` had failed previously while running on node `is-scale-03`, with the error "non-zero exit", and another container was restarted in its place.

At this point, we can ask the following questions:

- When you run `docker service ps --no-trunc`, is the error something obvious? Does the error say that it cannot mount a volume, or that the image is not found? If so, that's most likely the root cause of the issue and what needs to be addressed
- Did the node on which that container was running go down? Or is that node still up?
- Are the other services running on that node running fine? Was only this single service affected?
- If other services are running fine on that same node, it is probably a problem with the service itself. If all services on that node are not functional, it could mean a node failure.

At this point, we should be confident that the cause of the issue is not a deploy configuration issue, it is not an entire node failure, and the problem exists within the service itself. Continue to Step 2 if this is the case.

Step 2: Check for any error messages or logs indicating an error

Using the id obtained from step 1 we can collect the logs from the failed container with the following commands:

```
docker service logs <failed-id>
```

For example:

```
docker service logs 3s7s86n45skf
```

Search the service logs for any explicit errors or warning messages that might indicate why the failure occurred.

Usually, you can find the error message in those logs, but if the container ran out of memory, it may not be seen here. Continue to Step 3 if the logs provide nothing fruitful.

Step 3: Check for out of memory events

If there were no errors in the logs, or anywhere else that can be seen, a possible cause for a container restart could be that the system ran out of memory.

Perform the following steps to identify if this is the case:

1. Log in to the node where the container failed in our example. As seen in step 1, the container failed on `is-scale-03`.
2. From the node where the container failed, run the following command:

```
journalctl -k | grep -i -e memory -e oom
```

3. Check the result for any out of memory events that caused the container to stop. Such an event typically looks like the following:

```
is-scale-03 kernel: Out of memory: Kill process 5946 (redis-server) score 575  
or sacrifice child
```

Troubleshooting a Cloud Deployment of the Integration Service


After completing the AWS setup instructions, if none of the services start and you see the following error during troubleshooting, the problem is that you need to restart Docker after installing the RPM installation.

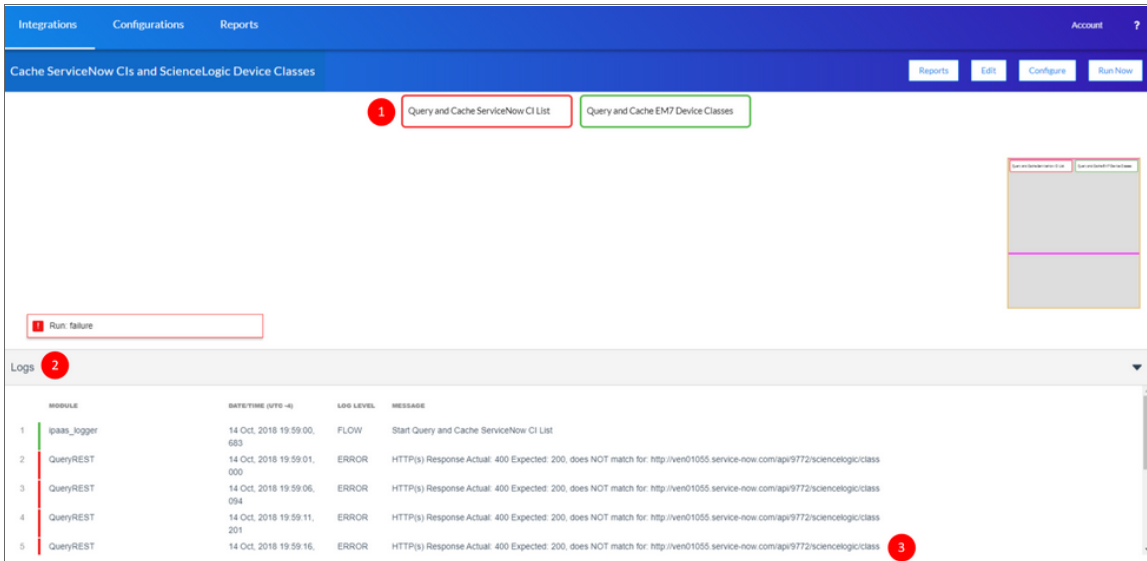
```
sudo docker service ps iservices_couchbase --no-trunc
```

```
"error creating external connectivity network: Failed to Setup IP tables: Unable to  
enable SKIP DNAT rule: (iptables failed: iptables --wait -t nat -I DOCKER -i docker_  
gwbridge -j RETURN: iptables: No chain/target/match by that name."
```

Identifying Why an Integration Application Failed

Determining Where an Integration Application Failed

If an application fails in the Integration Service, a failure icon () appears on the row for that application on the [Integrations] tab.



To determine where the integration application is failing:

1. Open the integration application and locate which step is failing. A failed step is highlighted in red in the image above.
2. Select the step and click the **Step Log** to view the logs for that step.
3. Review the error message to determine the next steps.

Retrieving Additional Debug Information (Debug Mode)

WARNING: If you run integration applications with "loglevel": 10, those integration applications will take longer to run because of increased I/O requirements. Enabling debug logging using the following process is the only recommended method. ScienceLogic does not recommend setting "loglevel": 10 for the whole stack with the docker-compose file.

To run an application in Debug Mode, POST the following to the API endpoint:

```
https://<integration_service>/api/v1/applications/run
```

Request body:

```

{
  "name": "<application_name>",
  "params": {
    "loglevel": 10
  }
}

```

After running the integration application in Debug Mode, go back to the Integration Service user interface and review the step logs to see detailed debug output for each step in the integration application. When run in Debug Mode, the step log output shows additional debug statements such as "Saved data for next step", which displays the data being sent from one step to the next. This information is especially helpful when trying to understand why an application or step failed.

Step	Type	Time	Level	Message
6	MySqlSelect	15 Oct, 2018 11:00:21, 626	INFO	Loaded parameter value: root, type <type 'str'> for parameter: username
7	MySqlSelect	15 Oct, 2018 11:00:21, 529	INFO	Loaded parameter value: em7admin, type <type 'str'> for parameter: password
8	MySqlSelect	15 Oct, 2018 11:00:21, 630	INFO	Loaded parameter value: SELECT did,title FROM master_dev.device_packages., type <type 'str'> for parameter: select_query
9	MySqlSelect	15 Oct, 2018 11:00:21, 633	INFO	Loaded parameter value: *, type <type 'str'> for parameter: fields
10	MySqlSelect	15 Oct, 2018 11:00:21, 633	INFO	Loaded parameter value: 7706, type <type 'int'> for parameter: port
11	BaseStep	15 Oct, 2018 11:00:21, 733	ERROR	Error when connecting to DB Host: 'http://192.168.32.188'. Username: 'root', database: 'master_dev' - (2003, 'Can't connect to MySQL server on 'http://192.168.32.188' (Errno -2) Name or service not known)'

You can also run an integration in debug using curl via SSH:

1. SSH to the Integration Service instance.
2. Run the following command:

```

curl -v -k -u isadmin:em7admin -X POST "https://<your_hostname>/api/v1/applications/run" -H 'Content-Type: application/json' -H 'cache-control: no-cache' -d '{"name": "interface_sync_sciencelogic_to_servicenow", "params": {"loglevel": 10}}'

```

General Troubleshooting Steps

The Integration Service acts as a middle server between data platforms. For this reason, the first steps should always be to ensure that there are no issues with the data platforms with which the Integration Service is talking. There might be additional configurations or actions enabled on ServiceNow or SL1 that result in unexpected behavior. For detailed information about how to perform the steps below, see [Resources for Troubleshooting](#).

Integration Service

1. Run the following command:

```
docker service ls
```

2. Note the Docker container version, and verify that the Docker services are running.
3. If a certain service is failing, make a note the service name and version.
4. If a certain service is failing, run `docker service ps <service_name>` to see the historical state of the service and make a note of this information. For example: `docker service ps iservices_contentapi`.
5. Make a note of any logs impacting the service by running `docker service logs <service_name>`. For example: `docker service logs iservices_couchbase`.

ServiceNow

1. Make a note of the ServiceNow version and SyncPack version, if applicable.
2. Make a note of whether the user is running an update set or the Certified Application.
3. Make a note of the ServiceNow integration application that is failing on the Integration Service.
4. Make a note of what step is failing in the integration application, try running the application in debug mode, and capture any traceback or error messages that occur in the step log.

Frequently Asked Questions

What is the first thing I should do when I have an issue with my Integration Service?

Ensure that all the services are up and running by running the following command:

```
docker service ls
```

Why do I get a "Connection refused" error when trying to communicate with Couchbase?

If you get a "Connection refused to Couchbase:8091" error when you are trying to communicate with Couchbase, check the firewalld service by running the following command:

```
systemctl status firewalld
```

Firewalld is responsible for all of the internal communications between the various Docker services on the Docker Swarm. If firewalld is not active, there will be no communications between the services, and you might see an error like "Connection refused to Couchbase:8091".

To start the firewalld service, run the following command:

```
systemctl start firewalld
```

How do I remove a schedule that does not have a name?

If you encounter a schedule in the Integration Service that was created without a name, the Integration Service cannot update or delete that schedule using the API.

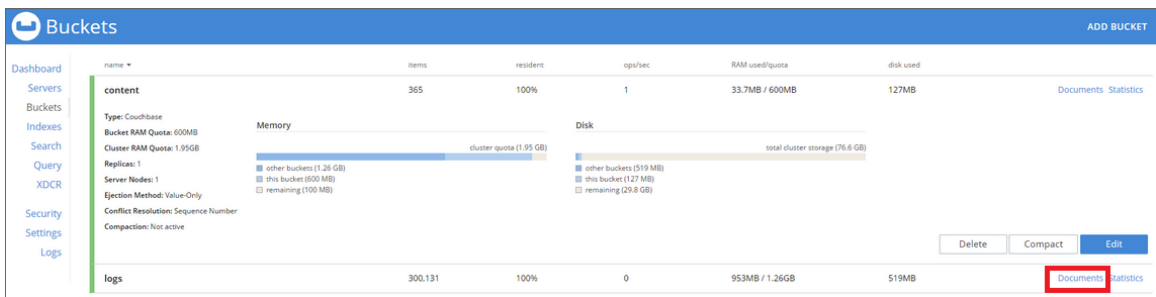
NOTE: This issue only affects versions of the Integration Service prior to version 1.8.2.

To address this issue, you will need to delete *all* schedules on the Integration Service, which involves going into Couchbase and deleting a file.

WARNING: Exercise extreme caution when performing this procedure, as deleting the wrong file will cause your Integration Service instance to stop working.

To delete all schedules, including a schedule without a name:

1. Log in to the Couchbase management interface at <https://<localhost>:8091>.
2. Navigate to the [Buckets] tab.
3. Click the **Documents** link on the **content** bucket:



4. On the **content > documents** page, extend the results to show 100 records per page
5. Search for "schedule". You will see a file similar to the following example:

is-system-diagnostic-configuration-example	("description":"Example Configuration for running Integration Service System Diagnostic Integration.", "href":"api/v1/configuration/s/is-system-diagnostic-configuration-example", "last_modified":1552595
is4_beat_schedule-536c5ae9-514e-462f-abce-2f4fe9c54d0c	(dp0 S'utc_enabled' p1 I01 sS'content_type' p2 S'schedule' p3 s'tz' p4 NsV'schedule' p5 (dp6 VadfjkakdlfDkjsdf p7 ccelery.beat ScheduleEntry p8 (VadfjkakdlfDkjsdf p9 V'cache_servicenow_CL_info' p10 cdate
is_backup	("description":"Backup Integration Service Data", "href":"api/v1/applications/is_backup", "last_modified":1552433810, "meta":{"hid

6. Delete the file.

WARNING: Deleting this file deletes all schedules on your Integration Service instance.

Why are there client-side timeouts when communicating with Couchbase?

If you are running an intensive integration application, or if you are running in Debug Mode, you might see the following stack trace error:

```
(generated, catch TimeoutError): <RC=0x17[Client-Side timeout exceeded for operation.
Inspect network conditions or increase the timeout], HTTP Request failed. Examine
'objextra' for full result, Results=1, C Source=(src/http.c,144),
OBJ=ViewResult<rc=0x17[Client-Side timeout exceeded for operation. Inspect network
conditions or increase the timeout], value=None, http_status=0, tracing_context=0,
tracing_output=None>, Tracing Output={"nokey:0": null}>
```

This error occurs when there is too much load going into the Couchbase database. If you're running with Debug Mode, that mode creates a large number of extra log messages in the database, which can contribute to this error.

To work around this issue, increase the timeout being used by setting the `db_host` environment variable in the steprunner service:

```
db_host: 'couchbase.isnet,localhost?n1ql_timeout=100000000.00'
```

If you increase the timeout, the timeout errors should go away.

NOTE: Increasing the timeout might not always be the correct action. If you are using an especially large system, you might want to allocate additional resources to the Couchbase services, including more memory for indexing and search. If you are encountering timeouts in a non-temporary fashion, such as only running Debug Mode for an integration application to determine what went wrong, you might want to add more resources instead of increasing the timeout.

What causes a Task Soft Timeout?

The following error might occur when you see a long-running task fail:

```
raise SoftTimeLimitExceeded() SoftTimeLimitExceeded: SoftTimeLimitExceeded()
```

This error message means that the default timeout for a task on your Integration Service system is too low. By default the task timeout, which is set by the environment variable `task_soft_time_limit`, is set to 3600 seconds (1 hour).

If you intend to have tasks executing for longer than an hour at a time, you can increase this setting by changing the `task_soft_time_limit` environment variable in your steprunners. Note that the value is set in seconds.

Why are incident numbers not populated in SL1 on Incident creation in ServiceNow?

If an incident exists in ServiceNow, but incident data is not getting back to SL1, and the "Sync ServiceNow Incident State to SL1 Event" integration application fails on the "Get Incident" step (with a 404 error) and eventually times out, the issue might be because the ServiceNow API is overloaded.

Why am I not getting any Incidents after disabling the firewall?

If you disabled the firewall to enable SNMP monitoring on the Integration Service, but were not able to connect, you should add the additional rule you need.

Why are Incidents not getting created in ServiceNow?

1. In SL1, locate the event that was created, and check the mailbox icon for that event.
2. Verify that the Run Book Action was triggered, and that the Run Book Action successfully posted to the Integration Service.
3. Get the associated integration ID, such as *isapp-24f2f1-23etc*, for that run of the "Create or Update ServiceNow Incident from SL1 Event" to see where the application failed.
4. Look at the logs for that run of the Integration Service integration application.

How can I point the "latest" container to my latest available images for the Integration Service?

If you force upgraded an RPM on top of an existing Integration Service RPM of the same version (such as version 1.8.0 force installed on 1.8.0), and you have custom worker types pointing to specific images, the *latest* tag gets created incorrectly.

To address this issue:

1. Modify the `docker-compose.yml` and update all SL1 images to point to the correct version that you expect to be latest.
2. Change any custom worker or custom services using SL1 containers to point to: *latest*.
3. Re-install the RPM of the same version via force.

How do I manually create a backup of my Integration Service system, and how do I manually restore?

To create a Couchbase backup:

1. Execute into the Couchbase container by running the following command:

```
cbbackup http://couchbase.isnet:8091 /opt/couchbase/var/backup -u [user] -p [password] -x data_only=1
```
2. Exit the couchbase shell and then copy the backup file in `/var/data/couchbase/backup` to a safe location, such as `/home/isadmin`.
3. Delete the Couchbase container by running the following command:

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

To do a manual restore:

1. Copy the backup file into `/var/data/couchbase/backup`.
2. Execute into the Couchbase container.
3. To restore the content, run the following command:

```
cbrestore /opt/couchbase/var/backup http://couchbase.isnet:8091 -b content -u
[user] -p [password]
```

4. To restore the logs, run the following command:

```
cbrestore /opt/couchbase/var/backup http://couchbase.isnet:8091 -b logs -u [user]
-p [password]
```

What do I do if I get a Code 500 error when I try to access the Integration Service user interface?

To address this issue:

1. SSH to your Integration Service instance.
2. Check your Docker services with the following command:

```
docker service ls
```

3. Ensure that all of your services are up and running:

```
root@fsmis4lab ~# docker service ls
ID                NAME                MODE                REPLICAS            IMAGE                PORTS
163k47dmc96m     services_contentapi replicated           1/1                 repository.auto.science.../is-apl:1.0.2
9qz3s3c9f4       services_couchbase  replicated           1/1                 repository.auto.science.../is-couchbase:1.0.2
4g1w0nkkqtc      services_flower     replicated           1/1                 repository.auto.science.../is-worker:1.0.2
4btoq1362a3n     services_gui        replicated           1/1                 repository.auto.science.../is-gui:1.0.2
*180->80/tcp, *1443->443/tcp, *18091->8091/tcp, *115672->115672/tcp
42201a9400       services_habbitmq   replicated           1/1                 repository.auto.science.../is-habbit:3.7.7-2
8k8ae5sq46m      services_redis      replicated           1/1                 repository.auto.science.../is-redis:4.0.11-2
717qm7hyqy       services_scheduler  replicated           1/1                 repository.auto.science.../is-worker:1.0.2
warch3mye9p      services_ssgrunner  replicated           0/0                 repository.auto.science.../is-worker:1.0.2
kml0v9gd29p      services_visual      replicated           1/1                 docker.samples/visualizer:latest
*18081->8080/tcp
root@fsmis4lab ~#
```

4. If your services are all up and running but you are still getting Code 500 errors, navigate to the Couchbase management portal of your Integration Service at port 8091 over HTTPS.
5. In the Couchbase portal, navigate to the **[Indexes]** tab and verify that all of your indexes are in a ready state:

Indexes		Global Indexes	Views			
bucket	node	index name	storage type	status	build progress	
content	couchbase.isnet:8091	#primary	Standard GSI	ready	100%	
content	couchbase.isnet:8091	idx_content_configuration_fb6ae58a_36fa_4453_...	Standard GSI	ready	100%	
content	couchbase.isnet:8091	idx_content_content_type_app_b0cbbd48_7e96_...	Standard GSI	ready	100%	
content	couchbase.isnet:8091	idx_content_content_type_config_fb6ae58a_36fa_...	Standard GSI	ready	100%	
content	couchbase.isnet:8091	idx_content_content_type_step_fb6ae58a_36fa_4_...	Standard GSI	ready	100%	
content	couchbase.isnet:8091	idx_content_report_id_fb6ae58a_36fa_4453_8f42_...	Standard GSI	ready	100%	
logs	couchbase.isnet:8091	#primary	Standard GSI	ready	100%	
logs	couchbase.isnet:8091	idx_logs_log_meta_id_v01_34f27325_097d_4ab8_...	Standard GSI	ready	100%	
logs	couchbase.isnet:8091	idx_logs_log_type_application_log_v02_1ece77b_...	Standard GSI	ready	100%	

6. Wait until all **status** entries are **ready** all **build progress** entries are **100%**, and then navigate back to your Integration Service user interface.
7. Verify that the code 500 error no longer exists.

8. If an index is stuck in a non-ready state, find the index name, copy that value and execute the following command in the Couchbase Query Editor:

```
BUILD INDEX ON content (INDEX_NAME_HERE)
```

What are some common examples of using the iscli tool?

The Integration Service system includes a command line tool called the iscli utility. You can use the iscli utility to upload components such as steps, configurations, and integration applications from the local file system onto the Integration Service.

For more information on how to use this tool, SSH to your Integration Service instance and type the following command:

```
iscli --help
```

You can use iscli to add drop files or additional content onto the Integration Service. You can also use the utility to upload content to a remote host. Examples of common syntax include the following:

```
iscli -usf <STEP_FILE.PY> -U isadmin -p em7admin  
iscli -uaf <APPLICATION_FILE.JSON> -U isadmin -p em7admin  
iscli -ucf <CONFIG_FILE.JSON> -U isadmin -p em7admin  
iscli -usf <STEP_FILE.PY> -U isadmin -p em7admin -H <IS_HOST>
```

How do I view a specific run of an application on IS?

To view the log results of a previous execution or run of an integration application in the Integration Service:

1. Use Postman or another API tool to locate the appID and the name of the integration application.
2. In the Integration Service, update the Integration Service URL with the appID, in the following format:

```
https://<IS_HOST>/integrations/<APP_NAME>?runid=<APP_ID>
```

For example:

```
https://<IS_HOST>/integrations/CreateServiceNowCI?runid=isapp-d8d1afad-74f8-42d4-b3ed-4a2ebcaef751
```

Using SL1 to Monitor the Integration Service

Overview

This manual describes how to monitor the Integration Service in SL1 using the *ScienceLogic: Integration Service PowerPack*.

The following topics provide an overview of the *ScienceLogic: Integration Service PowerPack*:

<i>What Does the Integration ServicePowerPack Monitor?</i>	118
<i>Monitoring Integration Applications in the Integration Service</i>	118
<i>Installing the ScienceLogic: Integration ServicePowerPack</i>	120
<i>Creating a SOAP/XML Credential for the Integration Service</i>	121
<i>Creating a Virtual Device for the Integration Service PowerPack</i>	122
<i>Aligning the Integration Service PowerPack Dynamic Applications</i>	123

NOTE: ScienceLogic provides this documentation for the convenience of ScienceLogic customers. Some of the configuration information contained herein pertains to third-party vendor software that is subject to change without notice to ScienceLogic. ScienceLogic makes every attempt to maintain accurate technical information and cannot be held responsible for defects or changes in third-party vendor software. There is no written or implied guarantee that information contained herein will work for all third-party variants. See the End User License Agreement (EULA) for more information.

What Does the Integration Service PowerPack Monitor?

To monitor an Integration Service instance with SL1, you must install the *ScienceLogic: Integration Service* PowerPack. This PowerPack lets you configure SL1 to create an alert if an integration application in the Integration Service fails.

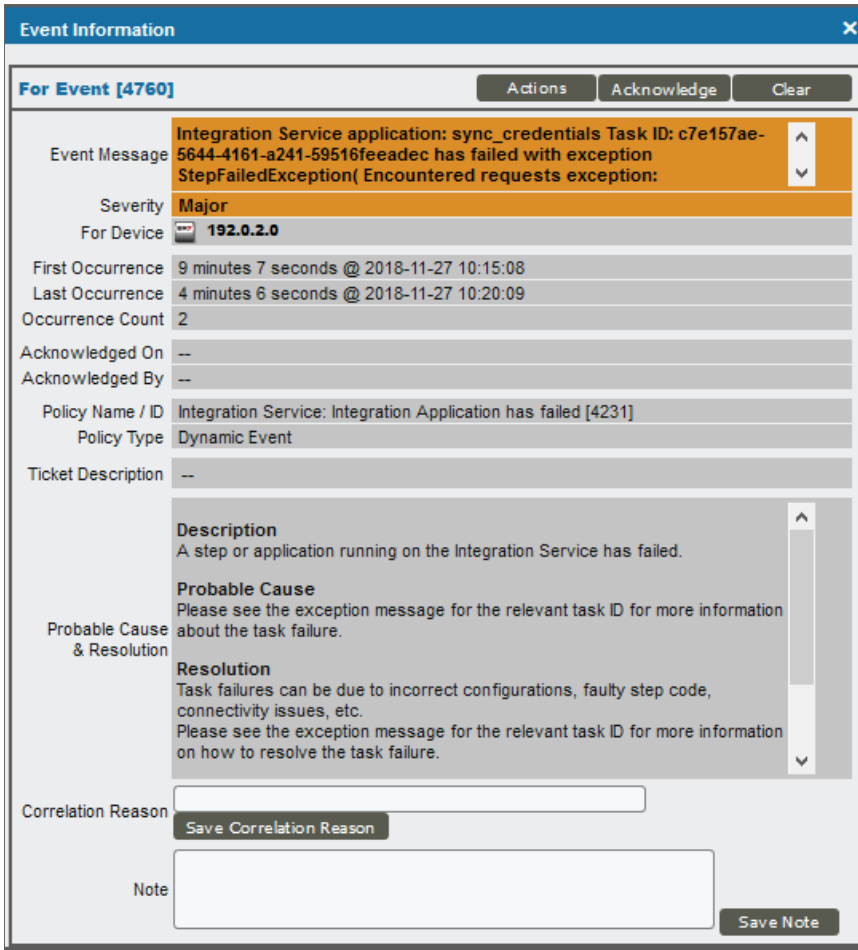
The *ScienceLogic: Integration Service* PowerPack includes:

- The "ScienceLogic: Integration Service Queue Configuration" Dynamic Application, which monitors the status of the Integration Service RabbitMQ service
- This PowerPack also includes the "REST: Performance Metrics Monitor" Dynamic Application, which monitors outgoing REST requests from SL1 to the Integration Service
- Event Policies and corresponding alerts that are triggered when an integration application in the Integration Service fails
- A Device Class for the Integration Service
- A Sample Credential for connecting to the Integration Service
- ScienceLogic Libraries that are utilized by this PowerPack:
 - content
 - content_cache
 - silo_core
 - silo_core_rest
 - silo_credentials

Monitoring Integration Applications in the Integration Service

The *ScienceLogic: Integration Service* PowerPack includes the ability to create an alert in the associated SL1 system if an integration application in the Integration Service fails.

The "ScienceLogic: Integration Service Queue Configuration" Dynamic Application generates a Major event in SL1 if an integration application fails in the Integration Service:



The related Event Policy includes the name of the application, the Task ID, and the traceback of the failure. You can use the application name to identify the integration application that failed on the Integration Service. You can use the Task ID to determine the exact execution of the application that failed, which you can then use for debugging purposes.

To view more information about the execution of an application in the Integration Service, navigate to the relevant page in the Integration Service by formatting the URL in the following manner:

```
https://<integration_service_hostname>/integrations/<application_name>?runid=<task_id>
```

For example:

```
https://192.0.2.0/integrations/sync_credentials?runid=c7e157ae-5644-4161-a241-59516feeadee
```

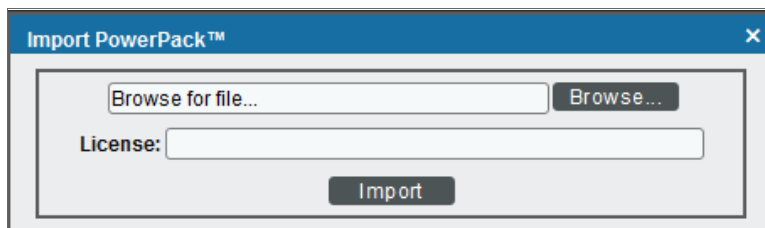
Installing the ScienceLogic: Integration ServicePowerPack

Before completing the steps in this section, you must import and install the latest version of the *ScienceLogic: Integration Service PowerPack*.

TIP: By default, installing a new version of a PowerPack overwrites all content in that PowerPack that has already been installed on the target system. You can use the **Enable Selective PowerPack Field Protection** setting in the **Behavior Settings** page (System > Settings > Behavior) to prevent new PowerPacks from overwriting local changes for some commonly customized fields. (For more information, see the **System Administration** manual.)

To download and install a PowerPack:

1. Download the PowerPack from the [ScienceLogic Customer Portal](#).
2. Go to the **PowerPack Manager** page (System > Manage > PowerPacks).
3. In the **PowerPack Manager** page, click the **[Actions]** button, then select *Import PowerPack*.
4. The **Import PowerPack** dialog box appears:



5. Click the **[Browse]** button and navigate to the PowerPack file.
6. When the **PowerPack Installer** modal page appears, click the **[Install]** button to install the PowerPack.

NOTE: If you exit the **PowerPack Installer** modal page without installing the imported PowerPack, the imported PowerPack will not appear in the **PowerPack Manager** page. However, the imported PowerPack will appear in the **Imported PowerPacks** modal page. This page appears when you click the **[Actions]** menu and select *Install PowerPack*.

Creating a SOAP/XML Credential for the Integration Service

To configure SL1 to monitor the Integration Service, you must first create a SOAP/XML credential. This credential allows the Dynamic Applications in the *ScienceLogic: Integration Service* PowerPack to communicate with the Integration Service.

The PowerPack includes an example SOAP/XML credential that you can edit for your own use.

To configure a SOAP/XML credential to access the Integration Service:

1. Go to the **Credential Management** page (System > Manage > Credentials).
2. Locate the **IS - Example** credential, and then click its wrench icon (🔧). The **Edit SOAP/XML Credential** modal page appears:

The screenshot shows the 'Edit SOAP/XML Credential #87' modal page. It features a blue header with 'New' and 'Reset' buttons. The main content is organized into four panels: 'Basic Settings' (Profile Name, Content Encoding, Method, HTTP Version, URL, HTTP Auth User, HTTP Auth Password, Timeout), 'Soap Options' (Embedded Password, Embed Value fields), 'Proxy Settings' (Hostname/IP, Port, User, Password), and 'CURL Options' (a list of options and a text area). The 'HTTP Headers' panel includes an 'Add a header' button and a field for 'Content-Type: application/json'. At the bottom, there are 'Save' and 'Save As' buttons.

3. Complete the following fields:
 - **Profile Name**. Type a name for the Integration Service credential.
 - **Content Encoding**. Select *text/xml*.
 - **Method**. Select *POST*.
 - **HTTP Version**. Select *HTTP/1.1*.
 - **URL**. Type the URL for your Integration Service system.
 - **HTTP Auth User**. Type the Integration Service administrator username.
 - **HTTP Auth Password**. Type the Integration Service administrator password.
 - **Timeout (seconds)**. Type "20".

- **Embed Value [%1]**. Type "False".
- **HTTP Headers**. Use the attached "Content-Type: application/json" header or click **Add a header** to define a custom HTTP header. Click the "bomb" icon to remove a header you do not need.

4. Click the **[Save As]** button.

Creating a Virtual Device for the Integration Service PowerPack

To monitor the Integration Service, you must create a **virtual device** that represents the Integration Service. You can use the virtual device to store information gathered by policies or Dynamic Applications.

To create a virtual device that represents the Integration Service:

1. Go to the **Device Manager** page (Registry > Devices > Device Manager).
2. Click the **[Actions]** button and select *Create Virtual Device* from the menu. The **Virtual Device** modal page appears:

The screenshot shows a modal window titled "Virtual Device" with a close button (X) in the top right. Inside the modal, there is a sub-header "Create Virtual Device" and a "Reset" button. Below this, there are four rows of input fields:

- Device Name:** A text input field containing "Integration Service Docs".
- Organization:** A dropdown menu with "System" selected.
- Device Class:** A dropdown menu with "ScienceLogic | Integration Service" selected.
- Collector:** A dropdown menu with "CUG" selected.

At the bottom of the form is an "Add" button.

3. Complete the following fields:

- **Device Name**. Type a name for the device.
- **Organization**. Select the organization for this device. The organization you associate with the device limits the users that will be able to view and edit the device. Typically, only members of the organization will be able to view and edit the device.
- **Device Class**. Select *ScienceLogic | Integration Service*.
- **Collector**. Select the collector group that will monitor the device.

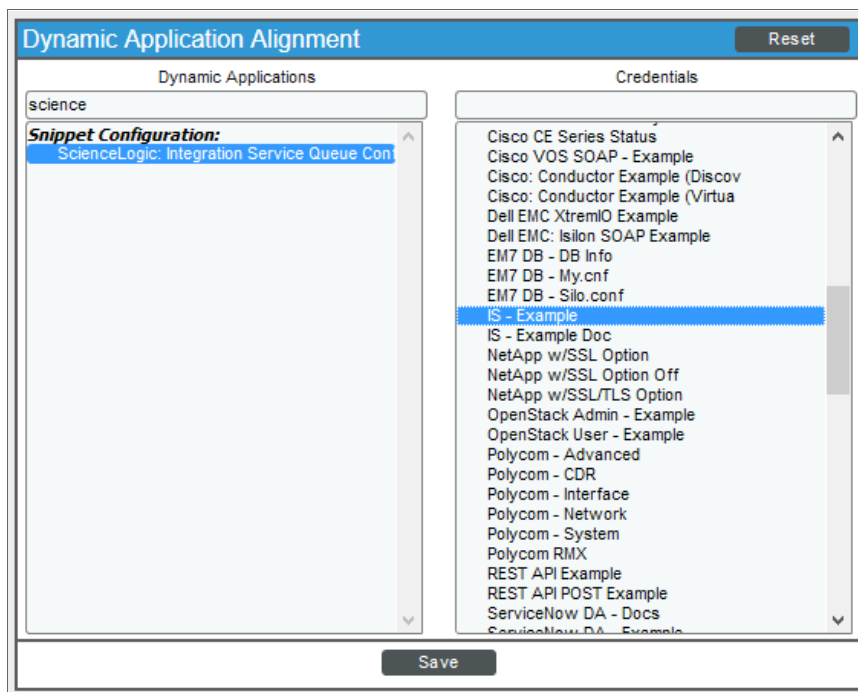
4. Click the **[Add]** button to create the virtual device.

Aligning the Integration Service PowerPack Dynamic Applications

Before you can run the Dynamic Applications in the *ScienceLogic: Integration Service PowerPack*, you must manually align each Dynamic Application to the virtual device you created in the previous step. Use the credential based on the **IS - Example** credential when you align the Dynamic Applications.

To align the Integration Service Dynamic Applications with the Integration Service virtual device:

1. Go to the **Device Manager** page (Registry > Devices > Device Manager).
2. Click the wrench icon (🔧) for the virtual device you created in the previous step. The **Device Properties** page appears.
3. Click the **[Collections]** tab. The **Dynamic Application Collections** page appears.
4. Click the **[Actions]** button and select *Add Dynamic Application*. The **Dynamic Application Alignment** modal page appears:



5. In the **Dynamic Applications** field, select the first of the Integration Service Dynamic Applications.
6. In the **Credentials** field, select the credential you created based on **IS - Example** credential.
7. Click the **[Save]** button.
8. Repeat steps 4-7 for each remaining Dynamic Application.

A

Integration Service for Multi-tenant Hosted Environments

Overview

This appendix describes the best practices and troubleshooting solutions for deploying the Integration Service in a hosted environment that supports multiple customers in a highly available fashion. This document also covers how to perform an upgrade of the Integration Service with minimal downtime.

This document covers the following topics:

<i>Quick Start Checklist for Deployment</i>	125
<i>Deployment</i>	125
<i>Onboarding a Customer</i>	129
<i>Failure Scenarios</i>	133
<i>Examples and Reference</i>	138
<i>Test Cases</i>	144
<i>Backup Considerations</i>	145
<i>Resiliency Considerations</i>	146
<i>Additional Sizing Considerations</i>	148
<i>Node Placement Considerations</i>	149
<i>Common Problems, Symptoms, and Solutions</i>	150
<i>Common Resolution Explanations</i>	158
<i>Integration Service Multi-tenant Upgrade Process</i>	162

Quick Start Checklist for Deployment

1. Deploy and cluster the initial High Availability stack. Label these nodes as "core".
2. For a desired customer, create the Integration Service configuration for the customer systems. This configuration information includes the SL1 IP address, the ServiceNow user and domain, and other related information.
3. Deploy and cluster the worker node or nodes for the customer.
4. Label the worker node or nodes specifically for the customer.
5. Update the **docker-compose.yml** file on a core node:
 - Add two steprunner services for each customer, one for real-time eventing, and one for backlogged events, labeled based on the organization name: *acme* and *acme-catchups*.
 - Update the new steprunner hostnames to indicate who the steprunner works for.
 - Update the new steprunner deploy constraints to deploy only to the designated labels.
 - Update the new steprunner *user_queues* environment variable to only listen on the desired queues.
6. Schedule the required integrations for this customer :
 - Device Sync daily, if desired
 - Correlation queue manager running on the catchup queue
7. Modify the Run Book Automations in SL1 to trigger the integration to run on the queue for this customer:
 - Modify the *IS_PASSTHROUGH* dictionary with "queue" setting.
 - Specify the configuration to use in the Integration Service for this SL1 instance.

Deployment

The following sections describe how to deploy the Integration Service in a hosted environment. After the initial High Availability (HA) core services are deployed, the hosted environment differs in the deployment and placement of workers and use of custom queues.

Core Service Nodes

For a hosted deployment, ScienceLogic recommends that you dedicate at least three nodes to the **core** Integration Service services. These core Integration Service services are shared by all workers and customers. As a result, it is essential that these services are clustered to handle failovers.

Because these core services are critical, ScienceLogic recommends that you initially allocate a fairly large amount of resources to these services. Allocating more resources than necessary to these nodes allows you to further scale workers in the future. If these nodes become overly taxed, you can add another node dedicated to the core services in the cluster.



These core services nodes are dedicated to the following services:

- API
- UI
- RabbitMQ
- Couchbase
- Redis

It is critical to monitor these core service nodes, and to always make sure these nodes have enough resources for new customers and workers as they are on-boarded.

To ensure proper failover and persistence of volumes and cluster information, the core services must be pinned to each of the nodes. For more information, see *Configuring Core Service Nodes*, below.

Requirements

3 nodes (or more for additional failover support) with 6 CPUs and 56 GB memory each

Configuring Core Service Nodes

- Install the Integration Service RPM on your core three nodes.
- See the [High Availability section](#) for information about how to join the cluster as a manager, and copy **the** `/etc/iservices/encryption_key` and `/etc/iservices/is_pass` file from a core service node to the new worker node (same location and permissions).
- [Create a label on the node](#) and label these nodes as "core node".
- See the [Configuring Clustering and High Availability](#) section for details on clustering Couchbase and RabbitMQ, and an example compose file of this setup.
- [Update the contentapi, UI, and redis services](#) so that those services are only ever deployed onto the core nodes.

Critical Elements to Monitor on Core Nodes

- Memory utilization: Warnings at 80%
- CPU utilization: Warnings at 80%
- RabbitMQ queue sizes (can also be monitored from the Flower API, or the Integration Service user interface)

Worker Service Nodes

Separate from the core services are the **worker services**. These worker services are intended to be deployed on nodes separate from the core services, and other workers, and these worker services aim to provide processing only for specified dedicated queues. Separating the VMs or modes where worker services are deployed will ensure that one customer's workload, no matter how heavy it gets, will not negatively affect the other core services, or other customer workloads.

Requirements

The resources allocated to the worker nodes depends on the worker sizing chosen, the more resources provided to a worker, the faster their throughput. Below is a brief guideline for sizing. Please note that even if you exceed the number of event syncs per minute, events will be queued up, so the sizing does not have to be exact. The below sizing just provides a suggested guideline.

Event Sync Throughput Node Sizing

CPU	Memory	Worker count	Time to sync a queue full of 10,000 events	Events Synced per second
2	16 GB	6	90 minutes	1.3
4	32 GB	12	46 minutes	3.6
8	54 GB	25	16.5 minutes	10.1

Test Environment and Scenario

1. Each Event Sync consists of Integration Service workers reading from the pre-populated queue of 10000 events. The sync interprets, transforms, and then POSTS the new event as a correlated ServiceNow incident into ServiceNow. This process goes on to then query ServiceNow for the new sysID generated for the incident, transforms it, and then POSTs it back to SL1 as an external ticket to complete the process.
 - Tests were performed on a node of workers only.
 - Tests were performed with a 2.6 GHz virtualized CPU in a vCenter VM. Both SL1 and ServiceNow were responding quickly when doing so.
 - Tests were performed with a pre-populated queue of 10000 events.
 - Tests were performed with the current deployed version of Cisco custom integration. Data will again be gathered for the next version when it is completed by Pro Services.
 - Each event on the queue consisted of a single correlated event.

Configuring the Worker Node

- Install the Integration Service RPM on the new node.
- See the [High Availability section](#) for information about how to join the cluster as a manager or worker, and copy the `/etc/iservices/encryption_key` and `/etc/iservices/is_pass` file from a core service node to the new worker node (same location and permissions).
- By default, the worker will listen on and accept work from the default queue, which is used primarily by the user interface, and any integration run without a custom queue.
- To configure this worker to run customer-specific workloads with custom queues, see [Onboarding a Customer](#).
- Modify the `docker-compose.yml` on a core service node accordingly.



- If you just want the node to accept default work, the only change necessary is to increase worker count using the table provided in the requirements section
- If you want the node to be customer specific, be sure to add the proper labels and setup custom queues for the worker in the docker-compose when deploying. This information is contained in the Onboarding a customer section.

Initial Worker Node Deployment Settings

It is required that there is always at least one worker instance listening on the default queue for proper functionality. The default worker can run in any node.

Worker Failover Considerations and Additional Sizing

When deploying a new worker, especially if it is going to be a custom queue dedicated worker, it is wise to consider deploying an extra worker listening on the same queues. If you have on a single worker node listening to a dedicated customer queue, there is potential for that queue processing to stop completely if that single node worker fails.

For this reason, ScienceLogic recommends that for each customer dedicated worker you deploy, you deploy a second one as well. This way there are two nodes listening to the customer dedicated queue, and if one node fails, the other node will continue processing from the queue with no interruptions.

When deciding on worker sizing, it's important to take this into consideration. For example, if you have a customer that requires a four-CPU node for optimal throughput, an option would be to deploy two nodes with two CPUs, so that there is failover if one node fails.

- How to know when more resources are necessary
- Extra worker nodes ready for additional load or failover

Knowing When More Resources are Necessary for a Worker

Monitoring the memory, CPU and pending integrations in queue can give you an indication of whether more resources are needed for the worker. Generally, when queue times start to build up, and tickets are not synced over in an acceptable time frame, more workers for task processing are required.

Although more workers will process more tasks, they will be unable to do so if the memory or CPU required by the additional workers is not present. When adding additional workers, it is important to watch the memory or CPU utilization, so long as the utilization is under 75%, it should be okay to add another worker. If utilization is consistently over 80%, then you should add more resources to the system before adding additional workers.

Keeping a Worker Node on Standby for Excess Load Distribution

Even if you have multiple workers dedicated to a single customer, there are still scenarios in which a particular customer queue spikes in load, and you'd like an immediate increase in throughput to handle this load. In this scenario you don't have the time to deploy a new IS node and configure it to distribute the load for greater throughput, as you need increased load immediately.

This can be handled by having a node on standby. This node has the same IS RPM version installed, and sits idle in the stack (or is turned off completely). When a spike happens, and you need more resources to distribute the load, you can then apply the label to the corresponding to the customer who's queues spiked. After setting the label on the standby node, you can scale up the worker count for that particular customer. Now, with the stand-alone node labeled for work for that customer, additional worker instances will be distributed to and started on the standby node.

When the spike has completed, you can return the node to standby by reversing the above process. Decrease the worker count to what it was earlier, and then remove the customer specific label from the node.

Critical Elements to Monitor in a Steprunner

- Memory utilization: Warnings at 80%
- CPU utilization: Warnings at 80%
- Successful, failed, active tasks executed by steprunner (retrievable from Flower API or PowerPack)
- Pending tasks in queue for the worker (retrievable by Flower API or PowerPack)
- Integrations in queue (similar information here as in pending tasks in queue, but this is retrievable from the Integration Service API).

Onboarding a Customer

When a new SL1 system is to be onboarded into the Integration Service, by default their integrations are executed on the default queue. In large hosted environments, ScienceLogic recommends separate queues for each customer. If desired, each customer can also have specific queues.

Create the Configuration

The first step to onboarding a new customer is to create a configuration with variables that will satisfy all integrations. The values of these should be specific to the new customer you're on boarding (such as SL1 IP address, username, password).

See the [example configuration](#) for a template you can fill out for each customer.

Because integrations might update their variable names from EM7 to SL1 in the future, ScienceLogic recommends to cover variables for both `em7_` and `sl1_`. The [example configuration](#) contains this information.

Label the Worker Node Specific to the Customer

For an example label, if you want a worker node to be dedicated to a customer called "acme", you could create a node label called "customer" and make the value of the label "acme". Setting this label now makes it easier to cluster in additional workers and distribute load dynamically in the future.

Creating a Node Label

This topic outlines creating a label for a node. Labels provide the ability to deploy a service to specific nodes (determined by labels) and to categorize the nodes for the work they will be performing. Take the following actions to set a node label:

```
# get the list of nodes available in this cluster (must run from a manager node)
docker node ls

# example of adding a label to a docker swarm node
docker node update --label-add customer=acme <node id>
```

Placing a Service on a Labeled Node

After you create a node label, refer to the example below for updating your **docker-compose-override.yml** file and ensuring the desired services deploy to the matching labeled nodes:

```
# example of placing workers on a specific labeled node
steprunner-acme:
  ...
  deploy:
    placement:
      constraints:
        - node.labels.customer == acme
    resources:
      limits:
        memory: 1.5G
      replicas: 15
  ...
```

Dedicating Queues Per Integration or Customer

Dedicating a separate queue for each customer is beneficial in that work and events created from one system will not affect or slow down work created from another system, provided the hosted system has enough resources allocated. In the example below, we created two new queues in addition to the default queue, and allocated workers to it. Both of these worker services use separate queues as described below, but run on the same labeled worker node.

Example Queues to Deploy:

- **acmequeue**. The queue we use to sync events specific from a customer called "acme". Only events syncs and other integrations for "acme" will run on this queue.
- **acmequeue-catchup**. The queue where any old events that should have synced over already (but failed due to Integration Service not being available or other reason) will run. Running these catchup integrations on a separate queue ensures that real-time event syncing isn't delayed in favor of an older event catching up.

Add Workers for the New Queues

First, define additional workers in our stack that are responsible for handling the new queues. All modifications are made in **docker-compose-override.yml**:

1. Copy an existing steprunner service definition.
2. Change the steprunner service name to something unique for the stack:
 - steprunner-acmequeue
 - steprunner-acmequeue-catchup
3. Modify the "replicas" value to specify how many workers should be listening to this queue:
 - steprunner-acmequeue will get 15 workers as it's expecting a very heavy load
 - steprunner-acmequeue-catchup will get 3 workers as it's not often that this will run
4. Add a new environment variable labeled "user_queues". This environment variable tells the worker what queues to listen to:
 - steprunner-acmequeue will set user_queues= "acmequeue"
 - steprunner-acmequeue-catchup will set user_queues="acmequeue-catchup"
5. To ensure that these workers can be easily identified for the queue to which they are assigned, modify the hostname setting :
 - Hostname: "acmequeue-{{.Task.ID}}"
 - Hostname "acmequeue-catchup-{{.Task.ID}}"
6. After the changes have been made, run `/opt/iservices/script/compose-override.sh` to validate that the syntax is correct.
7. When you are ready to deploy, re-run the docker stack deploy with the new compose file.

After these changes have been made, your docker-compose entries for the new steprunners should look similar to the following:

```

steprunner-acme-catchup:
  image: sciencelogic/is-worker:latest
  depends_on:
  - couchbase
  - rabbitmq
  - redis
  hostname: "acme-catchup-{{.Task.ID}}"
  deploy:
    placement:
      constraints:
        - node.labels.customer == acme
  resources:
    limits:
      memory: 2G
    replicas: 3
  environment:
    user_queues: 'acmequeue-catchup'
  ..
  ..
  ..

steprunner-acme:
  image: sciencelogic/is-worker:latest
  depends_on:
  - couchbase
  - rabbitmq

```

```

- redis
hostname: "acmequeue-{{.Task.ID}}"
deploy:
  placement:
    constraints:
      - node.labels.customer == acme
  resources:
    limits:
      memory: 2G
    replicas: 15
  environment:
    user_queues: 'acmequeue'
  ..
  ..
  ..

```

Once deployed via docker stack deploy, you should see the new workers in Flower, as in the following image:

Worker Name
celery@acme-queue-ikb81pi54ver91sxoysltcgbi
celery@acme-queue-9bkuebtrnfi349krlw8jlc0tn6
celery@acme-queue-koo0bg18bpsw4uvc2iz2l0vf9
celery@acme-queue-gilscrfx1giosh14pye73ui8
celery@acme-queue-u5kbd5m977wjfdbcbofhl230x3

You can verify the queues being listened to by looking at the "broker" section of Flower, or by clicking into a worker and clicking the **[Queues]** tab:

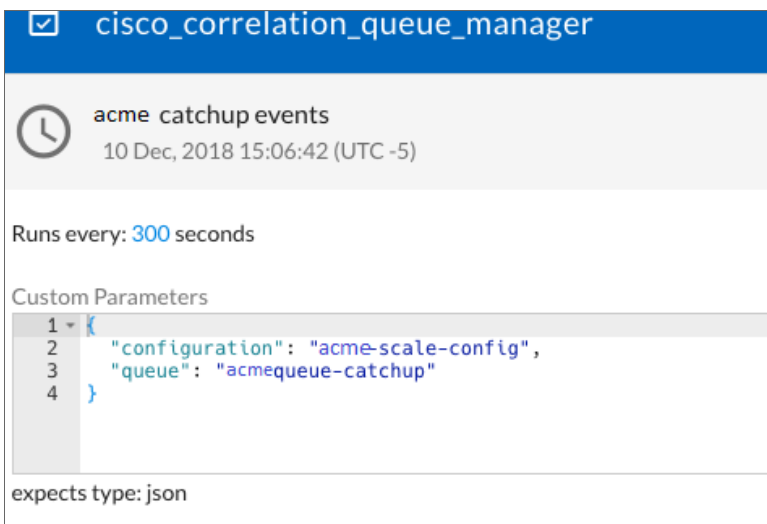
Queues					
Name	Messages	Unacked	Ready	Consumers	Idle since
celery	0	0	0	5	2018-12-10 20:08:45
acmequeue	886	420	466	15	N/A
acmequeue-catchup	0	0	0	2	2018-12-10 20:09:02
priority.high	0	0	0	5	2018-12-10 20:08:41

Create Integration Schedules and Automation Settings to Utilize Separate Queues

After the workers have been configured with specific queue assignments, schedule your integrations to run on those queues, and configure Run Book Automations (RBAs) to place the integrations on those queues.

Scheduling an Integration with a Specific Queue and Configuration

To run an integration on a specific queue using a configuration for a particular system, you can use the "params" override available in the scheduler. Below is an example of the scheduled integration which utilizes the acmequeue-catchup queue:



The screenshot shows a configuration for a scheduled integration named 'cisco_correlation_queue_manager'. It is checked and has a clock icon. The name is 'acme catchup events' and it was last run on '10 Dec, 2018 15:06:42 (UTC -5)'. It runs every 300 seconds. The custom parameters are a JSON object: `{ "configuration": "acme-scale-config", "queue": "acmequeue-catchup" }`. It expects a JSON type.

In the example above, the `correlation_queue_manager` is scheduled to run every 300 seconds, using the `acme` configuration, and will run on the `acmequeue`. You can have any number of scheduled integration runs per integration. If we were to add additional customers, we would add a new schedule entry with differing configurations, and queues for each.

Configuring Automations to Utilize a Specific Queue and Configuration

The last step to ensuring integrations for your newly onboarded SL1 system is to update the Run Book Automations in SL1 to provide the configuration and queue to use when the Run Book Automation triggers an event.

Modify the Event-correlation policy with the following changes:

1. `IS4_PASSTHROUGH= {"queue":"acmequeue"}`
2. `CONFIG_OVERRIDE= 'acme-scale-config'`

Failure Scenarios

This topic cover how the Integration Service handles situations where certain services fail.

Worker Containers

In case of failure, when can the worker containers be expected to restart?



- The worker containers have a strict memory limit of 2 GB. These containers may be restarted if the service requests more memory than the 2 GB limit.
- The restart is done by a SIGKILL of the OOM_Killer on the Linux system.

What happens when a worker container fails?

- Worker containers are ephemeral, and simply execute the tasks allotted to them. There is no impact to a worker instance restarting.

What processing is affected when service is down?

- The `task_reject_on_worker_lost` environment variable dictates whether the task being executed at the time the worker was restarted will be re-queued for execution by another worker. (default false)
- For more information about Celery, the task-processing framework used by the Integration Service): <http://docs.celeryproject.org/en/latest/userguide/configuration.html#task-reject-on-worker-lost>

What data can be lost?

- Workers contain no persistent data, so there is no data to lose, other than the data from the current task that is being executed on that worker when its shut down (if there is one)
- Any Integration Service integration application that fails can be replayed (and re-executed by the workers) on demand with the `/api/v1/tasks/<task-id>/replay` endpoint.

API

When can the API be expected to restart?

- The API also has a default memory limit. As with the steprunners (worker containers), if the memory limit is reached, the API is restarted by a SIGKILL of the OOM_Killer on the Linux system to prevent a memory leak.

What happens when it fails?

- On the clustered system, there are always three contentapi services, so if one of the API containers fails, API requests will still be routed to the functioning containers through the internal load balancer.

What processing is affected when service is down?

- If none of the API services are up and running, any Run Book Automation calls to sync an incident through the Integration Service results in an error. These failed event syncs are then placed in a database table in SL1 which the Integration Service queries on a schedule every few minutes. The next time that scheduled integration runs, the integration recognizes the events that failed to send to Integration Service, and the integration will process them so that the events sync.
- Even if the API is down, the events that were generated while it was down will be synced by the scheduled application. The Integration Service will reach out to SL1 for those items that SL1 failed to post to the Integration Service.

What data can be lost?

- The API contains no persistent data, so there is no risk of data loss.

Couchbase

If a core service node running Couchbase fails, the database should continue to work normally and continue processing events, as long as a suitable number of clustered nodes are still up and running. Three core service nodes provides automatic failover handling of one node, five core service nodes provides automatic failover handling of two nodes, and so on. See the [High Availability section](#) for more information.

If there are enough clustered core nodes still running, the failover will occur with no interruptions, and the failing node can be added back at any time with no interruptions.

NOTE: For optimal performance and data distribution after rejoining a cluster, you can click the **[Re-balance]** button from the Couchbase user interface, if needed.

If there are not enough clustered core nodes still running, then you will manually have to fail over the Couchbase Server. In this scenario, since automatic failover could not be performed (due to too few nodes available), there will be disruption in event processing. For more information, see the [Manual Failover section](#).

In case of failure, when can Couchbase be expected to restart?

- In ideal conditions, the Couchbase database should not restart, although Couchbase might be restarted when the node it is running on is over-provisioned. For more information, see the [known issue](#).

What happens when it fails?

- Each Couchbase node in the cluster contains a fully replicated set of data. If any single node fails, automatic failover will occur after the designated time (120 seconds by default). Automatic failover, processing, and queries to the database will continue without issue.
- If the database simply is restarted and not down for a long period of time (120 seconds), then the system will not automatically fail over, and the cluster will still be maintained.
- If two out of three of the database nodes fail for a period of time, processing may be paused until a user takes manual failover action. These manual actions are documented in the [Manual Failover section](#).

What processing is affected when service is down?

- In the event of an automatic failover (1/3 node failure), no processing will be affected and queries to the database will still be functional.
- In the event of a large failure (2/3 node failure) automatic failover will not happen, and manual intervention may be needed to so you can query the database again.

What data can be lost?

- Every Couchbase node has full data replication between each of the three nodes. In the event of a failure of any of the nodes, no data is lost, as a replicated copy exists across the cluster.

RabbitMQ

RabbitMQ clustered among all core service nodes provides full mirroring to each node. So long as there is at least one node available running RabbitMQ, the queues should exist and be reachable. This means that a multiple node failure will have no affect on the RabbitMQ services, and it should continue to operate normally.

In case of failure, when can RabbitMQ be expected to restart?

- Similar to the Couchbase database, in a smooth-running system, RabbitMQ should never really restart.
- As with other containers, RabbitMQ might be restarted when the node it is running on is over-provisioned. For more information, see the [known issue](#).

What happens when RabbitMQ fails?

- All RabbitMQ nodes in the cluster mirror each others' queues and completely replicate the data between them. The data is also persisted.
- In the event of any RabbitMQ node failure, the other nodes in the cluster will take over responsibility for processing its queues.
- If all RabbitMQ nodes are restarted, their messages are persisted to disk, so any tasks or messages sitting in queue at the time of the failure are not lost, and are reloaded once the system comes back up.
- In the event of a network partition ("[split-brain state](#)") RabbitMQ will follow the configured partition handling strategy (default autoheal).
- For more information, see <https://www.rabbitmq.com/partitions.html#automatic-handling>.

What processing is affected when service is down?

- When this service is down completely (no nodes running), the API may fail to place event sync tasks onto the queue. As such, any Run Book Automation calls to sync an incident through the Integration Service will result in an error.
- These failed event syncs are then placed in a database table in SL1 which the Integration Service queries on a schedule every few minutes. The next time that scheduled integration runs, the integration recognizes the events that failed to send to Integration Service, and the integration will process them so that the events sync.

What data can be lost?

- All data is replicated between nodes, so there is little risk of data loss.
- The only time there may be loss of tasks in queues is if there is a network partition, also called a "[split-brain state](#)".

Integration Service User Interface

In case of failure, when can the user interface be expected to restart?

- The Integration Service user interface (GUI) should never be seen as restarted unless a user forcefully restarted the interface.

- The Integration Service user interface might be restarted when the node it is running on is over-provisioned. For more information, see the [known issue](#).

What happens when it fails?

- The GUI service provides the proxy routing throughout the stack, so if the GUI service is not available Run Book Automation POSTS to the Integration Service will fail. However, as with an API failure, if the Run Book Actions can not POST to the Integration Service, those events will be placed in a database table in SL1 which the Integration Service queries on a schedule every few minutes. The next time that scheduled integration runs, the integration recognizes the events that failed to send to Integration Service, and the integration will process them so that the events sync.
- When the GUI service is down and SL1 cannot POST to it, the syncing of the events might be slightly delayed, as the events will be pulled in and created with the next run of the scheduled integration.

What data can be lost?

- The Integration Service user interface persists no data, so there is no risk of any data loss.

Redis

If the Redis service fails, it will automatically be restarted, and will be available again in a few minutes. The impact of this happening, is that task processing in the Integration Service is delayed slightly, as the worker services pause themselves and wait for the Redis service to become available again.

Consistent Redis failures

Consistent failures and restarts in Redis typically indicate your system has too little memory, or the Redis service memory limit is set too low, or not low at all. By default the Integration Service version 1.8.1 and later ships with a default memory limit of 8 GB to ensure that the Redis service only ever uses 8 GB of memory, and it ejects entries if it is going to go over that limit. This limit is typically sufficient, though if you have enough workers running large enough integrations to overfill the memory, you may need to increase the limit.

Before increasing Redis memory limit, be sure that there is suitable memory available to the system.

Known Issue for Groups of Containers

If you see multiple containers restarting at the same time on the same node, it indicates an over-provisioning of resources on that node. This only occurs on Swarm manager nodes, as the nodes are not only responsible for the services they are running, but also for maintaining the Swarm cluster and communicating with other manager nodes.

If resources become over-provisioned on one of those manager nodes (as they were with the core nodes when we saw the failure), the Swarm manager will not be able to perform its duties and may cause a docker restart on that particular node. This failure is indicated by “context deadline exceeded”, and “heartbeat failures” in the `journalctl -no-page |grep docker |grep err` logs.



This is one of the reasons why docker recommends running “manager-only” nodes, in which the manager nodes are only responsible for maintaining the Swarm, and not responsible for running other services. If any nodes that are running Integration Service services are also a Swarm manager, make sure that the nodes are not over-provisioned, otherwise the containers on that node may restart. For this reason, ScienceLogic recommends monitoring and placing thresholds at 80% utilization.

To combat the risk of over-provisioning affecting the docker Swarm manager, apply resource constraints on the services for the nodes that are also Swarm managers, so that docker operations always have some extra memory or CPU on the host to do what they need to do. Alternatively, you can only use drained nodes, which are not running any services, as Swarm managers, and not apply any extra constraints.

For more information about Swarm management, see https://docs.docker.com/engine/Swarm/admin_guide/.

Examples and Reference

Example of an Integration Service Configuration Object

```
[
  {
    "encrypted": false,
    "name": "em7_host",
    "value": "<ip address>"
  },
  {
    "encrypted": false,
    "name": "sll_host",
    "value": "${config.em7_host}"
  },
  {
    "encrypted": false,
    "name": "sll_id",
    "value": "${config.em7_id}"
  },
  {
    "encrypted": false,
    "name": "sll_db_port",
    "value": 7706
  },
  {
    "encrypted": false,
    "name": "snow_host",
    "value": "<arecord>.service-now.com"
  },
  {
    "encrypted": true,
    "name": "em7_password",
    "value": "<password>"
  },
  {
    "encrypted": false,
    "name": "sll_user",
    "value": "${config.em7_user}"
  },
  {
    "encrypted": false,
    "name": "sll_password",
```

```

    "value": "${config.em7_password}"
  },
  {
    "encrypted": false,
    "name": "s11_db_user",
    "value": "${config.em7_db_user}"
  },
  {
    "encrypted": false,
    "name": "s11_db_password",
    "value": "${config.em7_db_password}"
  },
  {
    "encrypted": false,
    "name": "em7_user",
    "value": "<username>"
  },
  {
    "encrypted": false,
    "name": "em7_db_user",
    "value": "root"
  },
  {
    "encrypted": false,
    "name": "em7_db_password",
    "value": "<password>"
  },
  {
    "encrypted": false,
    "name": "snow_user",
    "value": "<username>"
  },
  {
    "encrypted": true,
    "name": "snow_password",
    "value": "<password>"
  },
  {
    "encrypted": false,
    "name": "Domain_Credentials",
    "value": {
      "c9818d2c4a36231201624433851894bb": {
        "password": "3m7Admin!",
        "user": "is4DomainUser2"
      }
    }
  },
  {
    "name": "region",
    "value": "ACMEScaleStack"
  },
  {
    "encrypted": false,
    "name": "em7_id",
    "value": "${config.region}"
  },
  {
    "encrypted": false,
    "name": "generate_report",
    "value": "true"
  }
}

```



]

Example of a Schedule Configuration

```
[
  {
    "application_id": "device_sync_sciencelogic_to_servicenow",
    "entry_id": "dsync every 13 hrs acme",
    "last_run": null,
    "params": {
      "configuration": "acme-scale-config",
      "mappings": {
        "cmbd_ci_ip_router": [
          "Cisco Systems | 12410 GSR",
          "Cisco Systems | AIR-AP1141N",
          "Cisco Systems | AP 1200-IOS",
          "Cisco Systems | Catalyst 5505"
        ],
        "cmdb_ci_esx_resource_pool": [
          "VMware | Resource Pool"
        ],
        "cmdb_ci_esx_server": [
          "VMware | ESXi 5.1 w/HR",
          "VMware | Host Server",
          "VMware | ESX(i) 4.0",
          "VMware | ESX(i) w/HR",
          "VMware | ESX(i) 4.0 w/HR",
          "VMware | ESX(i)",
          "VMware | ESX(i) 4.1 w/HR",
          "VMware | ESXi 5.1 w/HR",
          "VMware | ESXi 5.0 w/HR",
          "VMware | ESX(i) 4.1",
          "VMware | ESXi 5.1",
          "VMware | ESXi 5.0"
        ],
        "cmdb_ci_linux_server": [
          "ScienceLogic, Inc. | EM7 Message Collector",
          "ScienceLogic, Inc. | EM7 Customer Portal",
          "ScienceLogic, Inc. | EM7 All-In-One",
          "ScienceLogic, Inc. | EM7 Integration Server",
          "ScienceLogic, Inc. | EM7 Admin Portal",
          "ScienceLogic, Inc. | EM7 Database",
          "ScienceLogic, Inc. | OEM",
          "ScienceLogic, Inc. | EM7 Data Collector",
          "NET-SNMP | Linux",
          "RHEL | Redhat 5.5",
          "Virtual Device | Content Verification"
        ],
        "cmdb_ci_vcenter": [
          "VMware | vCenter",
          "Virtual Device | Windows Services"
        ],
        "cmdb_ci_vcenter_cluster": [
          "VMware | Cluster"
        ],
        "cmdb_ci_vcenter_datacenter": [
          "VMware | Datacenter"
        ],
        "cmdb_ci_vcenter_datastore": [
          "VMware | Datastore",

```

```

    "VMware | Datastore Cluser"
  ],
  "cmdb_ci_vcenter_dv_port_group": [
    "VMware | Distributed Virtual Portgroup"
  ],
  "cmdb_ci_vcenter_dvs": [
    "VMware | Distributed Virtual Switch"
  ],
  "cmdb_ci_vcenter_folder": [
    "VMware | Folder"
  ],
  "cmdb_ci_vcenter_network": [
    "VMware | Network"
  ],
  "cmdb_ci_vmware_instance": [
    "VMware | Virtual Machine"
  ]
  ],
  "queue": "acmequeue",
  "region": "ACMEScaleStack"
},
"schedule": {
  "schedule_info": {
    "run_every": 47200
  },
  "schedule_type": "frequency"
},
"total_runs": 0
},
{
  "application_id": "device_sync_sciencelogic_to_servicenow",
  "entry_id": "dsync every 12 hrs on .223",
  "last_run": null,
  "params": {
    "configuration": "em7-host-223",
    "mappings": {
      "cmdb_ci_esx_resource_pool": [
        "VMware | Resource Pool"
      ],
      "cmdb_ci_esx_server": [
        "VMware | ESXi 5.1 w/HR",
        "VMware | Host Server",
        "VMware | ESX(i) 4.0",
        "VMware | ESX(i) w/HR",
        "VMware | ESX(i) 4.0 w/HR",
        "VMware | ESX(i)",
        "VMware | ESX(i) 4.1 w/HR",
        "VMware | ESXi 5.1 w/HR",
        "VMware | ESXi 5.0 w/HR",
        "VMware | ESX(i) 4.1",
        "VMware | ESXi 5.1",
        "VMware | ESXi 5.0"
      ],
      "cmdb_ci_linux_server": [
        "ScienceLogic, Inc. | EM7 Message Collector",
        "ScienceLogic, Inc. | EM7 Customer Portal",
        "ScienceLogic, Inc. | EM7 All-In-One",
        "ScienceLogic, Inc. | EM7 Integration Server",
        "ScienceLogic, Inc. | EM7 Admin Portal",
        "ScienceLogic, Inc. | EM7 Database",
        "ScienceLogic, Inc. | OEM",
      ]
    }
  }
}

```



```

        "ScienceLogic, Inc. | EM7 Data Collector",
        "NET-SNMP | Linux",
        "RHEL | Redhat 5.5",
        "Virtual Device | Content Verification"
    ],
    "cmdb_ci_vcenter": [
        "VMware | vCenter",
        "Virtual Device | Windows Services"
    ],
    "cmdb_ci_vcenter_cluster": [
        "VMware | Cluster"
    ],
    "cmdb_ci_vcenter_datacenter": [
        "VMware | Datacenter"
    ],
    "cmdb_ci_vcenter_datastore": [
        "VMware | Datastore",
        "VMware | Datastore Cluser"
    ],
    "cmdb_ci_vcenter_dv_port_group": [
        "VMware | Distributed Virtual Portgroup"
    ],
    "cmdb_ci_vcenter_dvs": [
        "VMware | Distributed Virtual Switch"
    ],
    "cmdb_ci_vcenter_folder": [
        "VMware | Folder"
    ],
    "cmdb_ci_vcenter_network": [
        "VMware | Network"
    ],
    "cmdb_ci_vmware_instance": [
        "VMware | Virtual Machine"
    ]
    ]
},
"schedule": {
    "schedule_info": {
        "run_every": 43200
    },
    "schedule_type": "frequency"
},
"total_runs": 0
},
{
    "application_id": "cisco_correlation_queue_manager",
    "entry_id": "acme catchup events",
    "last_run": {
        "href": "/api/v1/tasks/isapp-a20d5e08-a802-4437-92ef-32d643c6b777",
        "start_time": 1544474203
    },
    "params": {
        "configuration": "acme-scale-config",
        "queue": "acmequeue-catchup"
    },
    "schedule": {
        "schedule_info": {
            "run_every": 300
        },
        "schedule_type": "frequency"
    },
}

```

```

    "total_runs": 33
  },
  {
    "application_id": "cisco_incident_state_sync",
    "entry_id": "incident sync every 5 mins on .223",
    "last_run": {
      "href": "/api/v1/tasks/isapp-52b19097-e0bf-450b-948c-487aff33fc3b",
      "start_time": 1544474203
    },
    "params": {
      "configuration": "em7-host-223"
    },
    "schedule": {
      "schedule_info": {
        "run_every": 300
      },
      "schedule_type": "frequency"
    },
    "total_runs": 2815
  },
  {
    "application_id": "cisco_incident_state_sync",
    "entry_id": "incident sync every 5 mins acme",
    "last_run": {
      "href": "/api/v1/tasks/isapp-dde1dba5-2343-4026-8801-35a02e4e57a1",
      "start_time": 1544474202
    },
    "params": {
      "configuration": "acme-scale-config",
      "queue": "acmequeue"
    },
    "schedule": {
      "schedule_info": {
        "run_every": 300
      },
      "schedule_type": "frequency"
    },
    "total_runs": 1587
  },
  {
    "application_id": "cisco_correlation_queue_manager",
    "entry_id": "qmanager .223",
    "last_run": {
      "href": "/api/v1/tasks/isapp-cb7cc2e5-eab1-474a-907a-055f26dbc36d",
      "start_time": 1544474203
    },
    "params": {
      "configuration": "em7-host-223"
    },
    "schedule": {
      "schedule_info": {
        "run_every": 300
      },
      "schedule_type": "frequency"
    },
    "total_runs": 1589
  }
]

```

Test Cases

Load Throughput Test Cases

Event throughput testing with the Integration Service only:

The following test cases can be attempted with any number of dedicated customer queues. The expectation is that each customer queue will be filled with 10,000 events, and then you can time how long it takes to process through all 10,000 events in each queue.

1. Disable any steprunners.
2. Trigger 10,000 events through SL1, and let them build up in the Integration Service queue.
3. After all 10,000 events are waiting in queue, enable the steprunners to begin processing.
4. Time the throughput of all event processing to get an estimate of how many events per second and per minute the Integration Service will handle.
5. The results from the ScienceLogic test system are listed in the [sizing section of worker nodes](#).

Event throughput testing with SL1 triggering the Integration Service:

This test is executed in the same manner as the event throughput test described above, but in this scenario you never disable the steprunners, and you let the events process through the Integration Service as they are alerted to by SL1.

1. Steprunners are running.
2. Trigger 10,000 events through SL1, and let the steprunners handle the events as they come in.
3. Time the throughput of all event processing to get an estimate of how many events per second and per minute the Integration Service will handle.

The difference between the timing of this test and the previous test can show how much of a delay the SL1 is taking to alert the Integration Service about an event, and subsequently sync it.

Failure Test Cases

1. Validate that bringing one of the core nodes down does not impact the overall functionality of the Integration Service system. Also, validate that bringing the core node back up rejoins the cluster and the system continues to be operational.
2. Validate that bringing down a dedicated worker node only affects that specific workers processing. Also validate that adding a new "standby" node is able to pick up the worker where the previous failed worker left off.
3. Validate that when the Redis service fails on any node, it is redistributed and is functional on another node.
4. Validate that when an integration application fails, you can view the failure in the Integration Service Timeline.
5. Validate that you can query for and filter only for failing tasks for a specific customer.

Separated queue test scenarios

1. Validate that scheduling two runs of the same integration application with differing configurations and queues works as expected:
 - Each scheduled run should be placed on the designated queue and configuration for that schedule.
 - The runs, their queues, and configurations should be viewable from the Integration Service Timeline, or can be queried from the log's endpoint.
2. Validate that each SL1 triggering event is correctly sending the appropriate queue and configuration that the event sync should be run on:
 - This data should be viewable from the Integration Service Timeline.
 - The queue and configuration should be correctly recognized by the Integration Service and executed by the corresponding worker.
3. Validate the behavior of a node left "on standby" waiting for a label to start picking up work. As soon as a label is assigned and workers are scaled, that node should begin processing the designated work.

Backup Considerations

This section covers the items you should back up in your Integration Service system, and how to restore backups.

What to Back Up

When taking backups of the Integration Service environment, collect the following information from the host level of your primary manager node (this is the node from which you control the stack):

Files in `/opt/iservices/scripts`:

- `/opt/iservices/scripts/docker-compose.yml`
- `/opt/iservices/scripts/docker-compose-override.yml`

All files in `/etc/iservices/`

- `/etc/iservices/is_pass`
- `/etc/iservices/encryption_key`

In addition to the above files, make sure you are storing Couchbase dumps somewhere by using the `cbbbackup` command, or the "Integration Service Backup" integration application.

Fall Back and Restore to a Disaster Recovery (Passive) System

You should do a data-only restore if:

- The system you're restoring to is a different configuration or cluster setup than the system you made the backup on
- The backup system has all the indexes added and already up to date

You should do a *full* restore if:



- The deployment where the backup was made is identical to the deployment where it is being restored (same amount of nodes, etc)
- There are no indexes defined on the system you're backing up

Once failed over, be sure to disable the "Integration Service Backup" integration application from being scheduled.

Resiliency Considerations

The RabbitMQ Split-brain Handling Strategy (SL1 Default Set to Autoheal)

If multiple RabbitMQ cluster nodes are lost at once, the cluster might enter a "Network Partition" or "Split-brain" state. In this state, the queues will become paused if there is no auto-handling policy applied. The cluster will remain paused until a user takes manual action. To ensure that the cluster knows how to handle this scenario as the user would want, and not pause waiting for manual intervention, it is essential to set a partition handling policy.

For more information on RabbitMQ Network partition (split-brain) state, how it can occur, and what happens, see: <http://www.rabbitmq.com/partitions.html>.

By default, ScienceLogic sets the partition policy to *autoheal* in favor of continued service if any nodes go down. However, depending on the environment, you might wish to change this setting.

For more information about the automatic split-brain handling strategies that RabbitMQ provides, see: <http://www.rabbitmq.com/partitions.html#automatic-handling>.

autoheal is the default setting set by SL1, and as such, queues should always be available, though if multiple nodes fail, some messages may be lost.

NOTE: If you are using `pause_minority` mode and a "split-brain" scenario occurs for RabbitMQ in a single cluster, when the split-brain situation is resolved, new messages that are queued will be mirrored (replicated between all nodes once again).

ScienceLogic Policy Recommendation

ScienceLogic's recommendations for applying changes to the default policy include the following:

- If you care more about **continuity of service** in a data center outage, with queues always available, even if the system doesn't retain some messages from a failed data center, use *autoheal*. This is the SL1 default setting.
- If you care more about **retaining message data** in a data center outage, with queues that might not be available until the nodes are back, but will recover themselves once nodes are back online to ensure that no messages are lost, use *pause_minority*.
- If you prefer **not to have RabbitMQ handle this scenario automatically**, and you want to manually recover your queues and data, where queues will be paused and unusable until manual intervention is made to determine where to fallback, use *ignore*.

Changing the RabbitMQ Default Split-brain Handling Policy

The best way to change the SL1 default split-brain strategy is to make a copy of the RabbitMQ config file from a running rabbit system, add your change, and then mount that config back into the appropriate place to apply your overrides.

1. Copy the config file from a currently running container:

```
docker cp <container-id>:/etc/rabbitmq/rabbitmq.conf /destination/on/host
```

2. Modify the config file:

```
change cluster_partition_handling value
```

3. Update your **docker-compose.yml** file to mount that file over the config for all rabbitmq nodes:

```
mount "[/path/to/config]/rabbitmq.conf:/etc/rabbitmq/rabbitmq.conf"
```

Using Drained Managers to Maintain Swarm Health

To maintain Swarm health, ScienceLogic recommends that you deploy some swarm managers that do not take any of the workload of the application. The only purpose for these managers is to maintain the health of the swarm. Separating these workloads ensures that a spike in application activity will not affect the swarm clustering management services.

ScienceLogic recommends that these systems have 2 CPU and 4 GB of memory.

To deploy a drained manager node:

1. Add your new manager node into the swarm.
2. Drain it with the following command:

```
docker node update --availability drain <node>
```

Draining the node ensures that no containers will be deployed to it.

For more information, see https://docs.docker.com/engine/swarm/admin_guide/.

Updating the Integration Service Cluster with Little to No Downtime

There are two potential update workflows for updating the Integration Service cluster. The first workflow involves using a Docker registry that is connectable to swarm nodes on the network. The second workflow requires manually copying the Integration Service RPM or containers to each individual node.

Updating Offline (No Connection to a Docker Registry)

1. Copy the Integration Service RPM over to all swarm nodes.
2. Only install the RPM on all nodes. Do not stack deploy. This RPM installation automatically extracts the latest Integration Service containers, making them available to each node in the cluster.
3. From the primary manager node, make sure your **docker-compose** file has been updated, and is now using the appropriate version tag: either *latest* for the latest version on the system, or *1.x.x*.
4. If all swarm nodes have the RPM installed, the container images should be runnable and the stack should update itself. If the RPM was missed installing on any of the nodes, it may not have the required images, and as a result, services might not deploy to that node.

Updating Online (All Nodes Have a Connection to a Docker Registry)

1. Install the Integration Service RPM only onto the master node.
2. Make sure the RPM doesn't contain any host-level changes, such as Docker daemon configuration updates. If there are host level updates, you might want to make that update on other nodes in the cluster.
3. Populate your Docker registry with the latest Integration Service images.
4. From the primary manager node, make sure your **docker-compose** file has been updated, and is now using the appropriate version tag: either *latest* for the latest version on the system, or *1.x.x*.
5. Docker stack deploy the services. Because all nodes have access to the same Docker registry, which has the designated images, all nodes will download the images automatically and update with the latest versions as defined by the **docker-compose** file.

Additional Sizing Considerations

This section covers the sizing considerations for the Couchbase, RabbitMQ, Redis, contentapi, and GUI services.

Sizing for Couchbase Services

The initial sizing provided for Couchbase nodes in the hosted cluster for 6 CPUs and 56 GB memory should be more than enough to handle multiple customer event syncing workloads.

ScienceLogic recommends monitoring the CPU percentage and Memory Utilization percentage of the Couchbase nodes to understand when a good time to increase resources is, such as when Memory and CPU are consistently above 80%.

Sizing for RabbitMQ Services

The only special considerations for RabbitMQ sizing is how many events you will plan for in the queue at once.

Every 10,000 events populated in the Integration Service queue will consume approximately 1.5 GB of memory.

NOTE: This memory usage is drained as soon as the events leave the queue.

Sizing for Redis Services

The initial sizing deployment for redis should be sufficient for multiple customer event syncing.

The only time memory might need to be increased to redis is if you are attempting to view logs from a previous run, and the logs are not available. A lack of run logs from a recently run integration indicates that the redis cache does not have enough room to store all the step and log data from recently executed runs.

Sizing for contentapi Services

The contentapi services sizing should remain limited at 2 GB memory, as is set by default.

If you notice timeouts, or 500s when there is a large load going through the Integration Service system, you may want to increase the number of contentapi replicas.

For more information, see [placement considerations](#), and ensure the API replicas are deployed in the same location as the redis instance.

Sizing for the GUI Service

The GUI service should not need to be scaled up at all, as it merely acts as an ingress proxy to the rest of the Integration Service services.

Sizing for Workers: Scheduler, Steprunner, Flower

Refer to the worker sizing charts provided by ScienceLogic for the recommended steprunner sizes.

Flower and Scheduler do not need to be scaled up at all.

Node Placement Considerations

Preventing a Known Issue: Place contentapi and Redis services in the Same Physical Location

An issue exists where if there latency exists between the contentapi and redis, the integrations page may not load. This issue is caused by the API making too many calls before returning. The added latency for each individual call can cause the overall endpoint to take longer to load than the designated timeout window of thirty seconds.

The only impact of this issue is the applications/ page won't load. There is no operational impact on the integrations as a whole, even if workers are in separate geos than redis.

There is also no risk to High Availability (HA) by placing the API and Redis services on the same geo. If for whatever reason that geo drops out, the containers will be restarted automatically in the other location.



Common Problems, Symptoms, and Solutions

Tool	Issue	Symptoms	Cause	Solution
Docker Visualizer	Docker Visualizer shows some services as "undefined".	<p>When viewing the Docker Visualizer user interface, some services are displayed as "undefined", and states aren't accurate.</p> <p>Impact: Cannot use Visualizer to get the current state of the stack.</p>	<p>Failing docker stack deployment: https://github.com/dockersamples/docker-swarm-visualizer/issues/110</p>	<p>Ensure your stack is healthy, and services are deployed correctly. If no services are failing and things are still showing as undefined, <i>elect a new swarm leader</i>.</p> <p>To prevent: Ensure your configuration is valid before deploying.</p>

Tool	Issue	Symptoms	Cause	Solution
RabbitMQ	RabbitMQ queues encountered a node failure and are in a "Network partition state" (split-brain scenario).	<p>The workers are able to connect to the queue, and there are messages on the queue, but the messages are not being distributed to the workers.</p> <p>Log in to the RabbitMQ admin user interface, which displays a message similar to "RabbitMQ experienced a network partition and the cluster is paused".</p> <p>Impact: The RabbitMQ cluster is paused and waiting for user intervention to clean the split-brain state.</p>	<p>Multi-node failure occurred, and rabbit wasn't able to determine who the new master should be. This also will only occur if there is NO partition handling policy in place (see the resiliency section for more information)</p> <p>Note: ScienceLogic sets the <i>autoheal</i> policy by default</p>	<p>Handle the split-brain partition state and resynchronize your RabbitMQ queues.</p> <p>Note: This is enabled by default.</p> <p>To prevent: Set a partition handling policy.</p> <p>See the Resiliency section for more information.</p>



Tool	Issue	Symptoms	Cause	Solution
RabbitMQ, <i>continued</i>		<p>Execing into the RabbitMQ container and running rabbitmqcli cluster-status shows nodes in a partition state like the following:</p> <pre data-bbox="548 470 945 1346"> [{nodes, [{disc, ['rabbit@rabbit_ node1.isnet', 'rabbit@rabbit_ node2.isnet', 'rabbit@rabbit_ node3.isnet', 'rabbit@rabbit_ node4.isnet', 'rabbit@rabbit_ node5.isnet', 'rabbit@rabbit_ node6.isnet']}]}], {running_nodes, ['rabbit@rabbit_ node4.isnet']}, {cluster_name, <<"rabbit@rabbit_ node1">>}, {partitions, [{'rabbit@rabbit_ node4.isnet', ['rabbit@rabbit_ node1.isnet', 'rabbit@rabbit_ node2.isnet', 'rabbit@rabbit_ node3.isnet', 'rabbit@rabbit_ node5.isnet', 'rabbit@rabbit_ node6.isnet']}]}], {alarms, [{'rabbit@rabbit_ node4.isnet', []}]}]} </pre>		
Integration Service steprunners and RabbitMQ	Workers constantly restarting, no real error message.	<p>Workers of a particular queue are not stable and constantly restart.</p> <p>Impact: One queue's workers will not be processing.</p>	<p>Multi-node failure in RabbitMQ, when it loses majority and can not failover.</p> <p>Queues go out of sync because of broken swarm.</p>	<p>Recreate queues for the particular worker.</p> <p>Resynchronize queues.</p> <p>To prevent: Deploy enough nodes to ensure quorum for failover.</p>

Tool	Issue	Symptoms	Cause	Solution
Couchbase	Couchbase node is unable to restart due to indexer error.	<p>This issue can be monitored in the Couchbase logs:</p> <pre>Service 'indexer' exited with status 134. Restarting. Messages: sync.runtime_Semacquire (0xc4236dd33c)</pre> <p>Impact: One couchbase node becomes corrupt.</p>	<p>Memory is removed from the database while it is in operation (memory must be dedicated to the VM running Couchbase).</p> <p>The Couchbase node encounters a failure, which causes the corruption.</p>	<p>Ensure that the memory allocated to your database nodes is dedicated and not shared among other VMs.</p> <p>To prevent: Ensure that the memory allocated to your database nodes is dedicated and not shared among other VMs.</p>
Couchbase	Couchbase is unable to rebalance.	<p>Couchbase nodes will not rebalance, usually with an error saying "exited by janitor".</p> <p>Impact: Couchbase nodes cannot rebalance and provide even replication.</p>	<p>Network issues: missing firewall rules or blocked ports.</p> <p>The Docker swarm network is stale because of a stack failure.</p>	<p>Validate that all firewall rules are in place, and that no external firewalls are blocking ports.</p> <p>Reset the Docker swarm network status by electing a new swarm leader.</p> <p>To prevent: Validate the firewall rules before deployment.</p> <p>Use drained managers to maintain swarm</p>



Tool	Issue	Symptoms	Cause	Solution
Integration Service steprunners to Couchbase	Steprunners unable to communicate to Couchbase	<p>Steprunners unable to communicate to Couchbase database, with errors like "client side timeout", or "connection reset by peer".</p> <p>Impact: Steprunners cannot access the database.</p>	<p>Missing Environment variables in compose:</p> <p>Check the db_host setting for the steprunner and make sure they specify all Couchbase hosts available .</p> <p>Validate couchbase settings, ensure that the proper aliases, hostname, and environment variables are set.</p> <p>Stale docker network.</p>	<p>Validate the deployment configuration and network settings of your docker-compose. Redeploy with valid settings.</p> <p>In the event of a swarm failure, or stale swarm network, reset the Docker swarm network status by electing a new swarm leader.</p> <p>To prevent: Validate hostnames, aliases, and environment settings before deployment.</p> <p>Use drained managers to maintain swarm</p>

Tool	Issue	Symptoms	Cause	Solution
Flower	Worker display in flower is not organized and hard to read, and it shows many old workers in an offline state.	Flower shows all containers that previously existed, even if they failed, cluttering the dashboard. Impact: Flower dashboard is not organized and hard to read.	Flower running for a long time while workers are restarted or coming up/coming down, maintaining the history of all the old workers. Another possibility is a known issue in task processing due to the <code>--max-tasks-per-child</code> setting. At high CPU workloads, the <code>max-tasks-per-child</code> setting causes workers to exit prematurely.	Restart the flower service by running the following command: <pre>docker service update --force iservices_flower</pre> You can also remove the <code>--max-tasks-per-child</code> setting in the steprunners.



Tool	Issue	Symptoms	Cause	Solution
All containers on a particular node	All containers on a particular node do not deploy.	Services are not deploying to a particular node, but instead they are getting moved to other nodes. Impact: The node is not running anything.	One of the following situations could cause this issue: Invalid label deployment configuration. The node does not have the containers you are telling it to deploy. The node is missing a required directory to mount into the container.	Make sure the node that you are deploying to is labeled correctly, and that the services you expect to be deployed there are properly constrained to that system. Go through the troubleshooting steps of "When a docker service doesn't deploy" to check that the service is not missing a requirement on the host. Check the node status for errors: <code>docker node ls</code> To prevent: Validate your configuration before deploying.

Tool	Issue	Symptoms	Cause	Solution
All containers on a particular node	All containers on a particular node periodically restart at the same time.	All containers on a particular node restart at the same time. The system logs indicate an error like: "error="rpc error: code = DeadlineExceeded desc = context deadline exceeded" Impact: All containers restart on a node.	This issue only occurs in single-node deployments when the only manager allocates too many resources to its containers, and the containers all restart since the swarm drops. The manager node gets overloaded by container workloads and is not able to handle swarm management, and the swarm loses quorum.	Use some drained manager nodes for swarm management to separate the workloads. To prevent: <i>Use drained managers to maintain swarm.</i>
General Docker service	Docker service does not deploy. Replicas remain at 0/3	Docker service does not deploy.	There are a variety of reasons for this issue, and you can reveal most causes by checking the service logs to address the issue.	<i>Identify the cause of the service not deploying.</i>
Integration Service user interface	The Timeline or the Integrations page do not appear in the user interface.	The Timeline is not showing accurate information, or the Integrations page is not rendering.	One of the following situations could cause these issues: Indexes do not exist on a particular Couchbase node. Latency between the API and the redis service is too great for the API to collect all the data it needs before the 30-second timeout is reached. The indexer can't keep up to a large number of requests, and Couchbase requires additional resources to service the requests.	Solutions: Verify that indexes exist. Place the API and redis containers in the same geography so there is little latency. This issue will be fixed in a future IS release Increase the amount of memory allocated to the Couchbase indexer service.

Common Resolution Explanations

This section contains a set of solutions and explanations for a variety of issues.

Elect a New Swarm Leader

Sometimes when managers lose connection to each other, either through latency or a workload spike, there are instances when the swarm needs to be reset or refreshed. By electing a new leader, you can effectively force the swarm to redo service discovery and refresh the metadata for the swarm. This procedure is highly preferred over removing and re-deploying the whole stack.

To elect a new swarm leader:

1. Make sure there at least three swarm managers in your stack.
2. To identify which node is the current leader, run the following command:

```
docker node ls
```

3. Demote the current leader with the command:

```
docker node demote <node>
```

4. Wait until a new node is elected leader:

```
docker node ls
```

5. After a new node is elected leader, promote the old node back to swarm leader:

```
docker node promote <node>
```

Recreate RabbitMQ Queues and Exchanges

NOTE: If you do not want to retain any messages in the queue, the following procedure is the best method for recreating the queues. If you do have data that you want to retain, you can [resynchronize RabbitMQ queues](#).

To recreate RabbitMQ queues:

1. Identify the queue or queues you need to delete:
 - If default workers are restarting, you need to delete queues celery and priority.high.
 - If a custom worker cannot connect to the queue, simply delete that worker's queue.
2. Delete the queue and exchange through the RabbitMQ admin console:
 - Log in to the RabbitMQ admin console and go to the **[Queues]** tab.
 - Find the queue you want to delete and click it for more details.
 - Scroll down and click the **[Delete Queue]** button.

- Go to the **[Exchanges]** tab and delete the exchange with the same name as the queue you just deleted.

3. Delete the queue and exchange through the command line interface:

- exec into a rabbitmq container
- Delete the queue needed:

```
rabbitmqadmin delete queue name=name_of_queue
```

- Delete the exchange needed:

```
rabbitmqadmin delete exchange name=name_of_queue
```

After you delete the queues, the queues will be recreated the next time a worker connects.

Resynchronize RabbitMQ Queues

If your RabbitMQ cluster ends up in a "split-brain" or partitioned state, you might need to manually decide which node should become the master. For more information, see <http://www.rabbitmq.com/partitions.html#recovering>.

To resynchronize RabbitMQ queues:

1. Identify which node you want to be the master. In most cases, the master is the node with the most messages in its queue.
2. After you have identified which node should be master, scale down all other RabbitMQ services:


```
docker service scale iservices_rabbitmqx=x0
```
3. After all other RabbitMQ services except the master have been scaled down, wait a few seconds, and then scale the other RabbitMQ services back to 1. Bringing all nodes but your new master down and back up again forces all nodes to sync to the state of the master that you chose.

Identify the Cause of a Service not Deploying

Step 1: Obtain the ID of the failed container for the service

Run the following command for the service that failed previously:

```
docker service ps --no-trunc <servicename>
```

For example:

```
docker service ps --no-trunc iservices_redis
```

```
root@is-scale-03 ~]# docker service ps iservices_redis
ID                NAME                IMAGE                NODE                DESIRED STATE    CURRENT STATE    ERROR
ORTS
1slu2qwbte        iservices_redis.1    redis:4.0.2         is-scale-04        Running          Running 2 hours ago
s7s86n45skf        \ iservices_redis.1    redis:4.0.2         is-scale-03        Shutdown        Failed 2 hours ago    "task: non-zero exit (137)"
```



From the command result above, we see that one container with the ID **3s7s86n45skf** failed previously running on node **is-scale-03** (non-zero exit) and another container was restarted in its place.

At this point, you can ask the following questions:

- Is the error when using `docker service ps --no-trunc` something obvious? Does the error say that it cannot mount a volume, or that the image was not found? If so, that is most likely the root cause of the issue and needs to be addressed.
- Did the node on which that container was running go down? Or is that node still up?
- Are the other services running on that node running fine, and was only this service affected? If other services are running fine on that same node, it is probably a problem with the service itself. If all services on that node are not functional, it could mean a node failure.

At this point, the cause of the issue is not a deploy configuration issue, and it is not an entire node failure. The problem exists within the service itself. Continue to Step 2 if this is the case.

Step 2: Check for any interesting error messages or logs indicating an error

Using the ID obtained in Step 1, collect the logs from the failed container with the following command:

```
docker service logs <failed-id>
```

For example:

```
docker service logs 3s7s86n45skf
```

Review the service logs for any explicit errors or warning messages that might indicate why the failure occurred.

Repair Couchbase Indexes

Index stuck in “created” (not ready) state

This situation usually occurs when a node starts creating an index, but another index creation was performed at the same time by another node. After the index is created, you can run a simple query to build the index which will change it from created to “ready”:

```
BUILD index on 'content'('idx_content_content_type_config_a3f867db_7430_4c4b_b1b6_138f06109edb') using GSI
```

Deleting an index

If you encounter duplicate indexes, such as a situation where indexes were manually created more than once, you can delete an index:

```
DROP index content.idx_content_content_type_config_d8a45ead_4bbb_4952_b0b0_2fe227702260
```

Recreating all indexes on a particular node

To recreate all indexes on a particular Couchbase node, exec into the couchbase container and run the following command:


```
Initialize_couchbase -s
```

NOTE: Running this command recreates all indexes, even if the indexes already exist.

Add a Broken Couchbase Node Back into the Cluster

To remove a Couchbase node and re-add it to the cluster:

1. Stop the node in Docker.
2. In the Couchbase user interface, you should see the node go down, failover manually, or wait the appropriate time until it automatically fails over.
3. Clean the Couchbase data directory on the necessary host by running the following command:

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

4. Restart the Couchbase node and watch it get added back into the cluster.
5. Click the **Rebalance** button to replicate data evenly across nodes.

Restore Couchbase Manually

Backup

1. Exec into the couchbase container

```
cbbbackup http://couchbase.isnet:8091 /opt/couchbase/var/backup -u [user] -p  
[password] -x data_only=1
```

2. Exit the couchbase shell and then copy the backup file in `/var/data/couchbase/backup` to a safe location, such as `/home/isadmin`.

Delete Couchbase

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

Restore

1. Copy the backup file into `/var/data/couchbase/backup`.
2. Execute into the Couchbase container.
3. The following command restores the content:

```
cbrestore /opt/couchbase/var/backup http://couchbase.isnet:8091 -b content -u  
<user> -p <password>
```

3. The following command restores the logs:

```
cbrestore /opt/couchbase/var/backup http://couchbase.isnet:8091 -b logs -u <user>  
-p <password>
```

Integration Service Multi-tenant Upgrade Process

This section describes how to upgrade the Integration Service in a multi-tenant environment with as little downtime as possible.

Perform Environment Checks Before Upgrading

Validate Cluster states

- Validate that all Couchbase nodes in the cluster are replicated and fully balanced.
- Validate that the RabbitMQ nodes are all clustered and queues have *ha-v1-all* policy applied.
- Validate that the RabbitMQ nodes do not have a large number of messages backed up in queue.

Validate Backups exist

- Ensure that you have a backup of the database before upgrading.
- Ensure that you have a copy of your most recently deployed docker-compose file. If all user-specific changes are only populated in *docker-compose-override*, this is not necessary, but you might want a backup copy.
- Make sure that each node in Couchbase is fully replicated, and no re-balancing is necessary.

Clean out old container images if desired

Before upgrading to the latest version of the Integration Service, check the local file system and see if there are any older versions taking up space that you might want to remove. These containers exist both locally on the fs and the internal docker registry. To view any old container version,s check the **/opt/iservices/images** directory. ScienceLogic recommends that you keep at a minimum the last version of containers, so you can downgrade if necessary.

Cleaning out images is not mandatory, but it is just a means of clearing out additional space on the system if necessary.

To remove old images:

1. Delete any unwanted versions in **/opt/iservices/images**.
2. Identify any unwanted images known to Docker with `docker images`.
3. Remove the images with the ID `docker rmi <id>`.

Prepare the Systems

Install the new RPM

The first step of upgrading is to install the new RPM on all systems in the stack. Doing so will ensure that the new containers are populated onto the system (if using that particular RPM), and any other host settings are changed. RPM installation does not pause any services or affect the docker system in any way, other than using some resources.

The Integration Service has two RPMs, one with containers and one without. If you have populated an internal docker registry with docker containers, you can install the RPM without containers built in. If no internal docker repository is present, you must install the RPM which has the containers built in it. Other than the containers, there is no difference between the RPMs.

For advanced users, installing the RPM can be skipped. However this means that the user is completely responsible for maintaining the docker-compose and host level configurations.

To install the RPM:

1. SSH into each node.
2. If installing the RPM which contains the container images built in, you may want to upgrade each core node one by one, so that the load of extracting the images doesn't affect all core nodes at once
3. Run the following command:

```
rpm -Uvh <new-rpm-file>
```

Compare compose file changes and resolve differences

After the RPM is installed, you will notice a new docker-compose file is placed at **/etc/iservices/scripts/docker-compose.yml**. As long as your environment-specific changes exist solely in the compose-override file, all user changes and new version updates will be resolved into that new docker-compose.yml file.

ScienceLogic recommends that you check the differences between the two docker-compose files. You should validate that:

1. All environment-specific and custom user settings that existed in the old docker-compose also exist in the new docker-compose file.
2. The image tags reference the correct version in the new docker-compose. If you are using an internal docker registry, be sure these image tags represent the images from your internal registry.
3. Make sure that any new environment variables added to services are applied to replicated services. To ensure these updates persist through the next upgrade, also make the changes in docker-compose-override.yml. In other words, if you added a new environment variable for Couchbase, make sure to apply that variable to couchbase-worker1 and couchbase-worker2 as well. If you added a new environment variable for the default steprunner, make sure to set the same environment variable on each custom worker as well.
4. If you are using the *latest* tag for images, and you are using a remote repository for downloading, be sure that the *latest* tag refers to the images in your repository.
5. The old docker-compose is completely unchanged, and it matches the current deployed environment. This enables the Integration Service to update services independently without restarting other services.
6. After you resolve any differences between the compose files has been resolved, proceed with the upgrade using the old docker-compose.yml (the one that matches the currently deployed environment).

Make containers available to systems

After you apply the host-level updates, you should make sure that the containers are available to the system.

If you upgraded using the RPM with container images included, the containers should already be on all of the nodes, you can run `docker images` to validate the new containers are present. If this is the case you may skip to the next section.

If the upgrade was performed using the RPM which did *not* contain the container images, ScienceLogic recommends that you run the following command to make sure all nodes have the latest images:

```
docker-compose -f <new_docker_compose_file> pull
```

This command validates that the containers specified by your compose file can be pulled and reached from the nodes. While not required, you might to make sure that the images can be pulled before starting the upgrade. If the images are not pulled manually, they will automatically be pulled by Docker when the new image is called for by the stack.

Perform the Upgrade

To perform the upgrade on a clustered system with little downtime, the Integration Service re-deploys services to the stack in groups. To do this, the Integration Service gradually makes the updates to groups of services and re-runs *docker stack deploy* for each change. To ensure that no unintended services are updated, start off using the same `docker-compose` file that was previously used to deploy. Reusing the same `docker-compose` file and updating only sections at a time ensures that only the intended services to be updated are affected at any given time.

Avoid putting all the changes in a single `docker-compose` file, and do a new *docker stack deploy* with all changes at once. If downtime is not a concern, you can update all services, but updating services gradually allows you to have little or no downtime.

WARNING: Before upgrading any group of services, be sure that the `docker-compose` file you are deploying from is *exactly* identical to the currently deployed stack (the previous version). Start with the same `docker-compose` file and update it for each group of services as needed,

Upgrade Redis, Scheduler, and Flower

The first group to update includes the Redis, Scheduler and Flower. If desired, this group can be upgraded along with any other group.

To update:

1. Copy the service entries for Redis, Scheduler and Flower from the new compose file into the old `docker-compose` file (the file that matches the currently deployed environment). Copying these entries makes it so that the only changes in the `docker-compose` file (compared to the deployed stack) are changes for Redis, Scheduler and Flower.
2. Run the following command:

```
docker stack deploy -c <old_compose_with_small_changes> iservices
```

3. Monitor the update, and wait until all services are up and running before proceeding.

Example image definition of this upgrade group:

```
services:
  contentapi:
    image: repository.auto.sciencelogic.local:5000/is-api:1.8.1

  couchbase:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:1.8.1

  couchbase-worker:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:1.8.1

  flower:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-1.8.3

  gui:
    image: repository.auto.sciencelogic.local:5000/is-gui:1.8.1

  rabbitmq:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2

  rabbitmq2:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2

  rabbitmq3:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2

  redis:
    image: repository.auto.sciencelogic.local:5000/is-redis:4.0.11-2

  scheduler:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-1.8.3

  steprunner:
    image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1

  couchbase-worker2:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:1.8.1

  steprunner2:
    image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1
```

Redis Version

As the Redis version might not change with every release of the Integration Service, there might not be any changes needed in the upgrade for Redis. This can be expected and is not an issue.

Flower Dashboard

Due to a known issue addressed in version 1.8.3 of the Integration Service, the Flower Dashboard might not display any workers. Flower eventually picks up the new workers when they are restarted in the worker group. If this is a concern, you can perform the Flower upgrade in the same group as the workers.

Upgrade Core Services (Rabbit and Couchbase)

The next group of services to update together are the RabbitMQ/Couchbase database services, as well as the GUI. Because the core services are individually defined and "pinned" to specific nodes, upgrade these two services at the same time, on a node-by-node basis. In between each node upgrade, wait and validate that the node rejoins the Couchbase and Rabbit clusters and re-balances appropriately.



Because there will always be two out of three nodes running these core services, this group should not cause any downtime for the system.

Rabbit/Couchbase Versions

The Couchbase and RabbitMQ versions used might not change with every release of the Integration Service. If there is no update or change to be made to the services, you can ignore this section for RabbitMQ or Couchbase upgrades, or both. Assess the differences between the old and new docker-compose files to check if there is an image or environment change necessary for the new version. If not, you can move on to the next section.

Update Actions (assuming three core nodes)

To update first node services:

1. Update just core *node01* by copying service entries for couchbase, rabbitmq1 from the new compose file (compared and resolved as part of above prepare steps) into the old docker-compose file. At this point, the compose file you use to deploy should also contain the updates for the previous groups
2. Before deploying, access the Couchbase user interface, select the first server node, and click "failover". Select graceful failover. Manually failing over before updating ensures that the system is still operational when the container comes down.
3. For the failover command that can be run through the command-line interface if the user interface is not available, see the [Manual Failover section](#).
4. Run the following command:

```
docker stack deploy -c <compose_file>
```
5. Monitor the process to make sure the service updates and restarts with the new version. To make sure that as little time as possible is used when updating the database, the database containers should already be available on the core nodes.
6. After the node is back up, go back to the Couchbase UI and add the node back, and rebalance the cluster to make it whole again.
7. For more information on how to re-add the node and rebalance the cluster if the user interface is not available, see the [Manual Failover section](#).

First node Couchbase update considerations:

- When updating the first couchbase node, be sure to set the environment variable JOIN_ON: "couchbase-worker2", so that the couchbase master knows to rejoin the workers after restarting
- Keep in mind by default, only the primary Couchbase node user interface is exposed. Because of this, when the first Couchbase node is restarted, the Couchbase admin user interface will be inaccessible. If you would like to have the Couchbase user interface available during the upgrade of this node, ensure that at least one other Couchbase-worker services port is exposed.

Special GUI consideration with 1.8.3

In the upgrade to version 1.8.3 of the Integration Service, the Couchbase and RabbitMQ user interface ports will be exposed through the Integration Service user interface with HTTPS. To ensure there is no port conflict between services and the Integration Service user interface, ensure that the Couchbase and RabbitMQ user interface port mappings are removed or modified from the default (8091) admin port. To avoid conflicts, make sure the new Integration Service user interface definition does not conflict with the Couchbase or RabbitMQ definitions.

- Modifying the port mapping (exposing to a different port than 8091) means it will still be exposed via HTTP through a port other than 8091 until the Integration Service user interface is upgraded, at which point that port can then be closed.
- Removing the port means no port mapping will be defined to 8091 for the Couchbase service, and the Couchbase user interface will be inaccessible until the Integration Service user interface is updated.

The Integration Service user interface will not update until all port conflicts are resolved. You can upgrade the Integration Service user interface at any time after this has been done, but be sure to first review the [Update the GUI](#) topic, below.

NOTE: You can manually remove port mappings from a service with the following command, though the command will restart the service: `docker service update --publish-rm published=8091,target=8091 iservices_couchbase`

Example docker-compose with images and JOIN_ON for updating the first node:

```
services:
  contentapi:
    image: repository.auto.sciencelogic.local:5000/is-api:1.8.1

  couchbase:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-1.8.3
    environment:
      JOIN_ON: "couchbase-worker2"

  couchbase-worker:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:1.8.1

  flower:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-1.8.3

  gui:
    image: repository.auto.sciencelogic.local:5000/is-gui:hotfix-1.8.3

  rabbitmq:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2

  rabbitmq2:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2

  rabbitmq3:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2

  redis:
    image: repository.auto.sciencelogic.local:5000/is-redis:4.0.11-2

  scheduler:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-1.8.3
```



```
steprunner:  
  image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1  
  
couchbase-worker2:  
  image: repository.auto.sciencelogic.local:5000/is-couchbase:1.8.1  
  
steprunner2:  
  image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1
```

Update second, and third node services

To update the second and third node services, repeat the steps from the first node on each node until all nodes are re-clustered and available. Be sure to check the service port mappings to ensure that there are no conflicts (as described above), and remove any HTTP ports if you choose.

Update the GUI

You can update the GUI service along with any other group, but due to the port mapping changes in version 1.8.3 of the Integration Service, you should update this service *after* the databases and RabbitMQ nodes have been updated, and their port mappings no longer conflict.

Since the GUI service provides all ingress proxy routing to the services, there might be a very small window where the Integration Service might not receive API requests as the GUI (proxy) is not running. This downtime is limited to the time it takes for the GUI container to restart.

To update the user interface:

1. Make sure that any conflicting port mappings are handled and addressed.
2. Replace the docker-compose GUI service definition with the new one.
3. Re-deploy the docker-compose file, and validate that the new GUI container is up and running.
4. Make sure that the HTTPS ports are accessible for Couchbase/RabbitMQ.

Update Workers and contentapi

You should update the workers and contentapi last. Because these services use multiple replicas (multiple steprunner or containerapi containers running per service), you can rely on Docker to incrementally update each replica of the service individually. By default, when a service is updated, it will update one container of the service at a time, and only after the previous container is up and stable will the next container be deployed.

You can utilize additional Docker options in docker-compose to set the behavior of how many containers to update at once, when to bring down the old container, and what happens if a container upgrade fails. See the *update_config* and *rollback_config* options available in Docker documentation:

<https://docs.docker.com/compose/compose-file/>.

Upgrade testing was performed by ScienceLogic using default options. An example where these settings are helpful is to change the parallelism of *update_config* so that all worker containers of a service update at the same time.

The update scenario described below takes extra precautions and only updates one node of workers per customer at a time. If you decide, you can also safely update all workers at once.

To update the workers and contentapi:

1. Modify the docker-compose file, the contentapi, and "worker_node1" services of all customers to use the new service definition.
2. Run a `docker stack deploy` of the new compose file. Monitor the update, which should update the API container one instance at a time, always leaving a container available to service requests. The process updates the workers of node1 one container instance at a time by default.
3. After workers are back up and the API is fully updated, modify the docker-compose file and update the second node's worker's service definitions.
4. Monitor the upgrade, and validate as needed.

Example docker-compose definition with one of two worker nodes and contentapi updated:

```
services:
  contentapi:
    image: repository.auto.sciencelogic.local:5000/is-api:hotfix-1.8.3
    deploy:
      replicas: 3

  couchbase:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-1.8.3
    environment:
      JOIN_ON: "couchbase-worker2"

  couchbase-worker:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-1.8.3

  flower:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-1.8.3

  gui:
    image: repository.auto.sciencelogic.local:5000/is-gui:hotfix-1.8.3

  rabbitmq:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2

  rabbitmq2:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2

  rabbitmq3:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2

  redis:
    image: repository.auto.sciencelogic.local:5000/is-redis:4.0.11-2

  scheduler:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-1.8.3

  steprunner:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-1.8.3

  couchbase-worker2:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-1.8.3
```

```
steprunner2:  
  image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1
```



© 2003 - 2019, ScienceLogic, Inc.

All rights reserved.

LIMITATION OF LIABILITY AND GENERAL DISCLAIMER

ALL INFORMATION AVAILABLE IN THIS GUIDE IS PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED. SCIENCELOGIC™ AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT.

Although ScienceLogic™ has attempted to provide accurate information on this Site, information on this Site may contain inadvertent technical inaccuracies or typographical errors, and ScienceLogic™ assumes no responsibility for the accuracy of the information. Information may be changed or updated without notice. ScienceLogic™ may also make improvements and / or changes in the products or services described in this Site at any time without notice.

Copyrights and Trademarks

ScienceLogic, the ScienceLogic logo, and EM7 are trademarks of ScienceLogic, Inc. in the United States, other countries, or both.

Below is a list of trademarks and service marks that should be credited to ScienceLogic, Inc. The ® and ™ symbols reflect the trademark registration status in the U.S. Patent and Trademark Office and may not be appropriate for materials to be distributed outside the United States.

- ScienceLogic™
- EM7™ and em7™
- Simplify IT™
- Dynamic Application™
- Relational Infrastructure Management™

The absence of a product or service name, slogan or logo from this list does not constitute a waiver of ScienceLogic's trademark or other intellectual property rights concerning that name, slogan, or logo.

Please note that laws concerning use of trademarks or product names vary by country. Always consult a local attorney for additional guidance.

Other

If any provision of this agreement shall be unlawful, void, or for any reason unenforceable, then that provision shall be deemed severable from this agreement and shall not affect the validity and enforceability of any remaining provisions. This is the entire agreement between the parties relating to the matters contained herein.

In the U.S. and other jurisdictions, trademark owners have a duty to police the use of their marks. Therefore, if you become aware of any improper use of ScienceLogic Trademarks, including infringement or counterfeiting by third parties, report them to Science Logic's legal department immediately. Report as much detail as possible about the misuse, including the name of the party, contact information, and copies or photographs of the potential misuse to: legal@sciencelogic.com



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

International: +1-703-354-1010