Integration Service Platform

Version 2.0.2

Table of Contents

Chapter

1

Introduction to the Integration Service

Overview

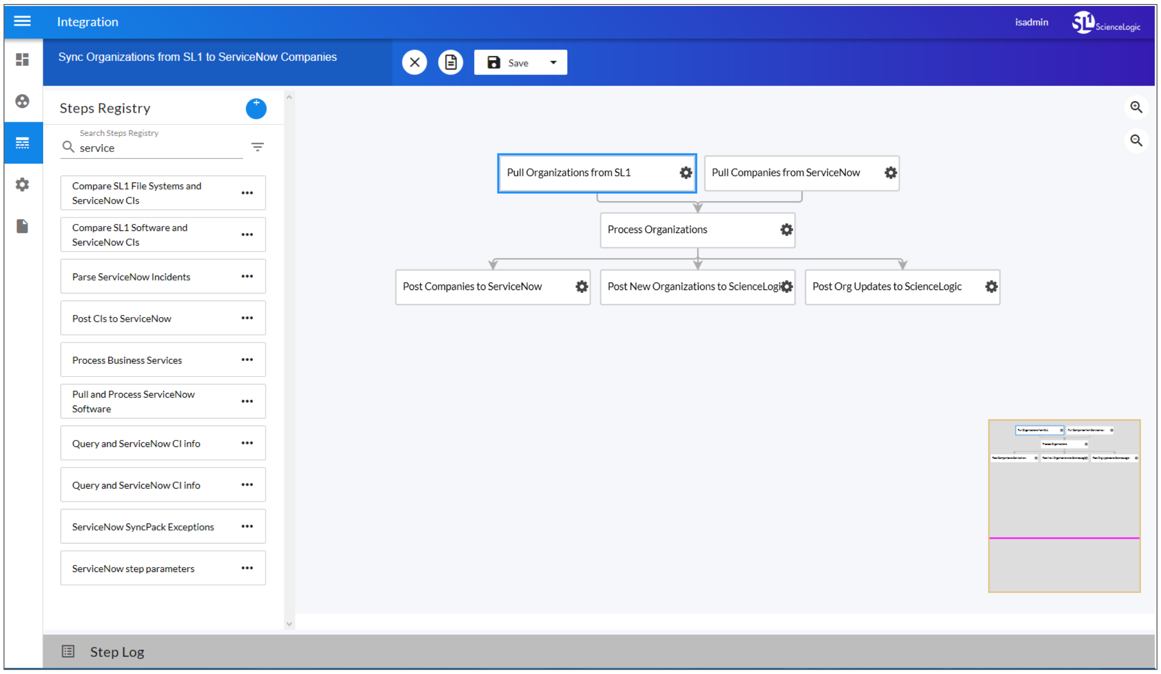
The Integration Service provides a generic platform for integrations between SL1 and third-party platforms.

This chapter covers the following topics:

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 following image shows an example of an integration application and its steps in the Integration Service user interface:


Integration
isadmin
ScienceLogic
Sync Organizations from SL1 to ServiceNow Companies
Save
Steps Registry
Search Steps Registry
service
Compare SL1 File Systems and ServiceNow CIs
Compare SL1 Software and ServiceNow CIs
Parse ServiceNow Incidents
Post CIs to ServiceNow
Process Business Services
Pull and Process ServiceNow Software
Query and ServiceNow CI Info
Query and ServiceNow CI Info
ServiceNow SyncPack Exceptions
ServiceNow step parameters
Pull Organizations from SL1
Pull Companies from ServiceNow
Process Organizations
Post Companies to ServiceNow
Post New Organizations to ScienceLogic
Post Org Updates to ScienceLogic
Step Log

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

- ***Steps***. A step is a generic Python class that performs a single action. Steps can accept zero or many input parameters or data from previous steps, and steps can specify output to be used by other steps. The input parameters are configurable variables and values used during execution. Steps can be re-used in multiple integrations applications. When these steps are combined in an integration application, they provide a workflow that satisfies a business requirement. All Python step code should be Python 3.7 or later. In the image above, the steps display as part of the flowchart in the main viewing pane as well as the **Steps Registry** pane.
- ***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 combines a set of steps that execute a workflow. The input 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. In the image above, the group of connected steps in the large pane make up the "Sync Organizations from SL1 to ServiceNow Companies" integration application. You can access all integration applications on the **Integrations** page (☰).

- ***Configuration Objects***. A configuration object is a stand-alone JSON file that contains a set of configuration variables used as input for an application. Configurations can include variables like hostname, user name, password, or other credential information. Configuration objects allow the same application to be deployed in multiple Integration Service instances, with different configurations. Click the **[Configure]** button from an integration application in the Integration Service user interface to access the configuration object for that integration application. You can access all configuration objects on the **Configurations** page (⚙).
- ***Synchronization PowerPack***. A Synchronization PowerPack (also called a SyncPack) contains all the code and logic needed to perform integrations on the Integration Service platform. You can access the latest steps, integration applications, and configurations for the Integration Service or a third-party integration (such as ServiceNow) by downloading the most recent Synchronization PowerPack for that integration from ScienceLogic. You can access all Synchronization PowerPacks on the **SyncPacks** page (❂).

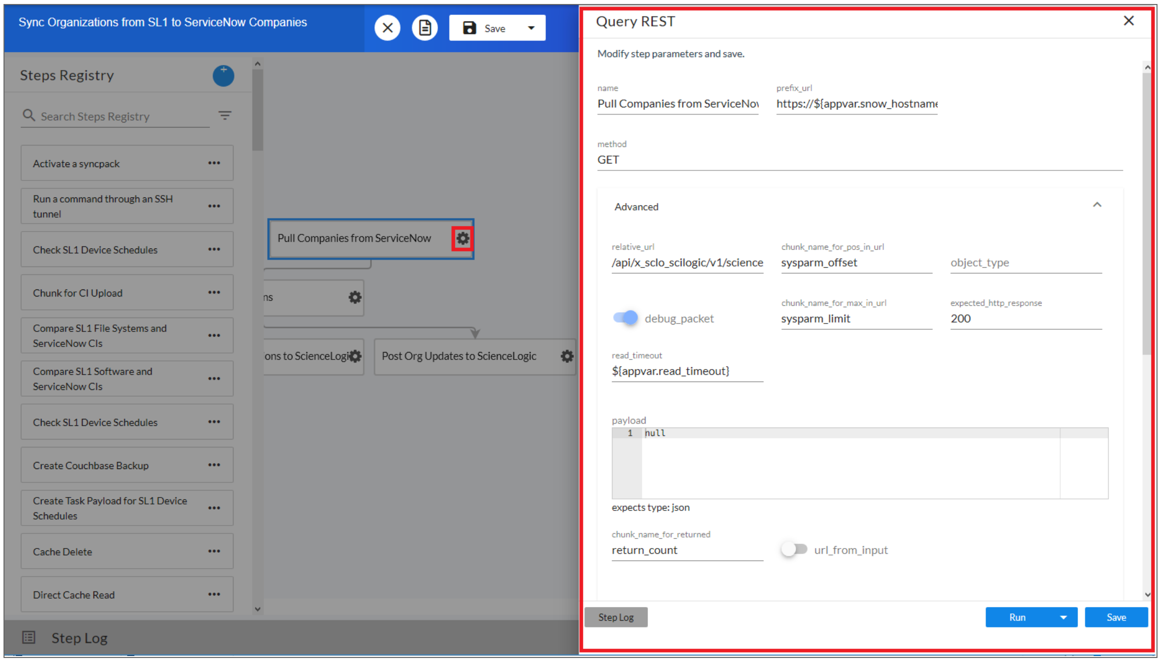
What is a Step?

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

Pull Companies from ServiceNow

Steps accept arguments called ***input parameters***. The parameters 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 step to query both the local system and another remote system; only the arguments, such as hostname, username, and password change.

You can view and edit the parameters for a step by opening the integration application, clicking **[Open Editor]** (✎), and then clicking the gear icon (⚙) on a step to open the **Step Editor** pane:


Sync Organizations from SL1 to ServiceNow Companies
Save
Steps Registry
Search Steps Registry
Activate a syncpack
Run a command through an SSH tunnel
Check SL1 Device Schedules
Chunk for CI Upload
Compare SL1 File Systems and ServiceNow CIs
Compare SL1 Software and ServiceNow CIs
Check SL1 Device Schedules
Create Couchbase Backup
Create Task Payload for SL1 Device Schedules
Cache Delete
Direct Cache Read
Step Log
Pull Companies from ServiceNow
ns
ons to ScienceLogi
Post Org Updates to ScienceLogic
Query REST
Advanced
Step Log
Run
Save

Modify step parameters and save.

name: Pull Companies from ServiceNov

prefix_url: https://${appvar.snow_hostname

method: GET

relative_url: /api/x_sclo_scilogic/v1/science

chunk_name_for_pos_in_url: sysparm_offset

object_type

debug_packet

chunk_name_for_max_in_url: sysparm_limit

expected_http_response: 200

read_timeout: ${appvar.read_timeout}

payload:

```
null
```

expects type: json

chunk_name_for_returned: return_count

url_from_input

A step can pass the data it generates during execution to a subsequent step. A step can use the data generated by another step. Also, you can add test data to the step and click **Custom Run** to run test data for that 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:

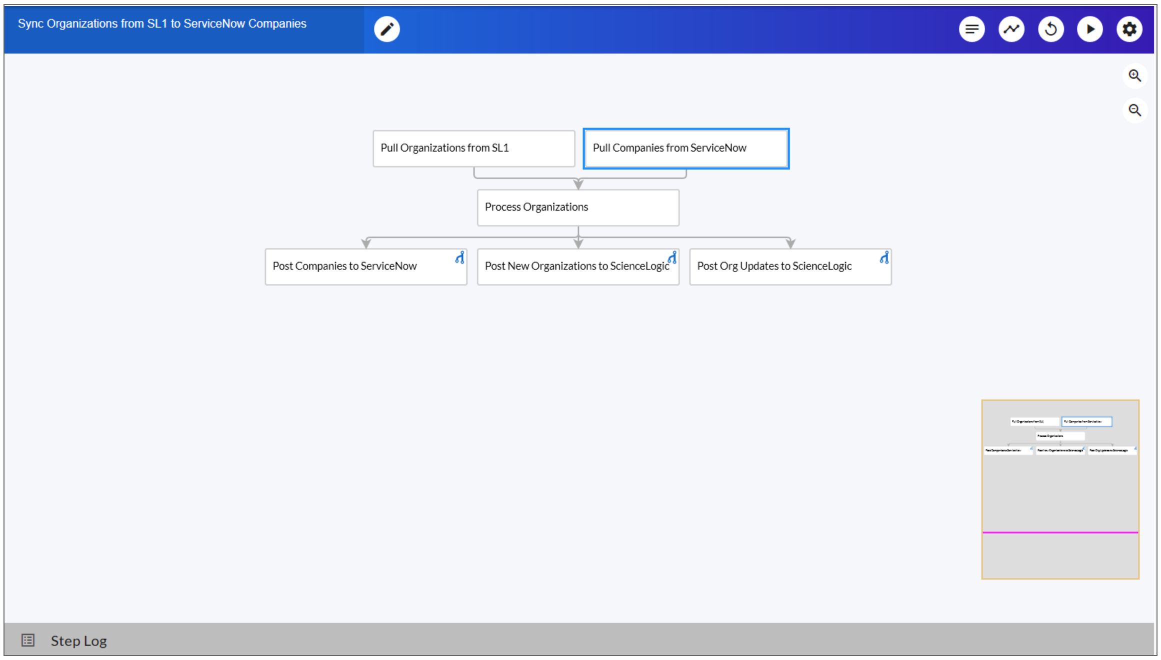
- ***Standard***. Standard steps do not require any previously collected data to perform. Standard steps are generally used to generate data to perform a transformation or a database insert. These steps can be run independently and concurrently.
- ***Aggregated***. Aggregated steps require data that 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:


Sync Organizations from SL1 to ServiceNow Companies
Pull Organizations from SL1
Pull Companies from ServiceNow
Process Organizations
Post Companies to ServiceNow
Post New Organizations to ScienceLogic
Post Org Updates to ScienceLogic
Step Log

Integration application JSON objects are defined by configuration settings, steps that make up the integration, and application-wide variables used 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.

For more information about integration applications, see *Managing Integration Applications*.

What is a Configuration Object?

Configuration variables are defined in a stand-alone JSON file called a ***configuration object*** 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"
}
```

Configuration objects can map variables from the SL1 platform to a third-party platform. For instance, SL1 has device classes and ServiceNow has CI classes; the configuration object map these two sets of variables together

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.

For more information about configuration objects, see ***Managing Configuration Objects***.

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 tool (iscli).

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

Elements of the Integration Service User Interface

With the 2.0.0 release of the Integration Service platform, the Integration Service user interface was updated to match the 8.12.0 and later release of the SL1 user interface, with the navigation tabs now located on the left-hand side of the window. The tabs provide access to the following pages: **Dashboard**, **SyncPacks**, **Integrations**, **Configurations**, **Reports**, and **Admin Panel**.

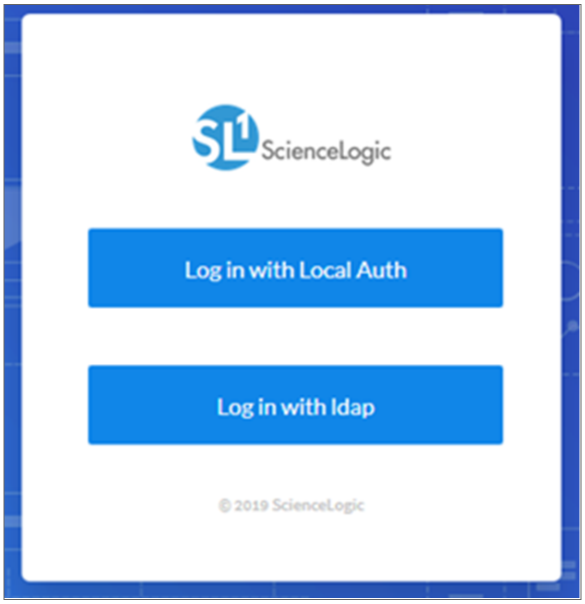
TIP: To view a pop-out list of menu options, click the menu icon (☰). Click the menu icon again to close the pop-out menu.

Logging In and Out of the Integration Service User Interface

Starting with version 2.0.0 of the Integration Service, you can log in to the Integration Service using one of the following authorization types:

- ***Local Authentication***. The same local Administrator user (*isadmin*) is supported by default with 2.0.0 installations. If you are migrating from a previous version of the Integration Service to version 2.0.0, you can log in and authenticate with the same user and password.
- ***Basic Authentication***. Integration Service 2.0.0 continues to support Basic Authentication as well. Because the Integration ServicePowerPacks, diagnostic scripts, and the iscli tool continue to use Basic Authentication, ScienceLogic does not recommend disabling Basic Authentication with Integration Service version 2.0.0.
- ***OAuth***. Lets Integration Service administrators use their own authentication providers to enforce user authentication and lockout policies. Authentication using a third-party provider, such as LDAP or Active Directory, requires additional configuration. For optimal security, ScienceLogic recommends that you disable the local Administrator user (*isadmin*) and exclusively use your own authentication provider.

Depending on the authentication used by your Integration Service system, your login page will display a single option for logging in, or more than one option, as in the following example:



For more information about configuring authorization for users, see ***Managing Users in the Integration Service***.

To log out of the Integration Service, click your user name in the navigation bar in the top right of any window and select *Log off*.

TIP: If you get a "SyncPacks service is not reachable" pop-up message in the user interface and the various pages are empty, log out of the Integration Service user interface and log back in again. You can also click **[Refresh]** in your browser to automatically log out. This situation occurs only if the user interface is idle for a long period of time.

Integration Service Pages

The **Dashboard** page (▦) provides a graphical view of the various tasks, workers, and integration applications that are running on you Integration Service system. For more information, see *Monitoring the Integration Service on the Dashboard Page*.

The **SyncPacks** page (⚙) lets you import, install, view, and edit Synchronization PowerPacks, which are also called "SyncPacks". For more information, see *Managing Synchronization PowerPacks*.

The **Integrations** page (☰) provides a list of the integration applications on your Integration Service system. From this page you can run, schedule, and create integration applications. For more information, see *Managing Integration Applications*.

The **Configurations** page (⚙) lets you create or use a configuration object to define a set of variables that all steps and integration applications can use. For more information, see *Managing Configuration Objects*.

The **Reports** page (📄) contains a list of reports associated with integration applications that have the reporting feature enabled, such as the "Integration Service System Diagnostics" application. For more information, see *Viewing a Report for an Integration Application*.

The **Admin Panel** page (👥) contains a list of user groups, which lets you determine the roles and access for your users. Only users with the *Administrator* role for this Integration Service system can edit this page. For more information, see *Managing Users in the Integration Service*.

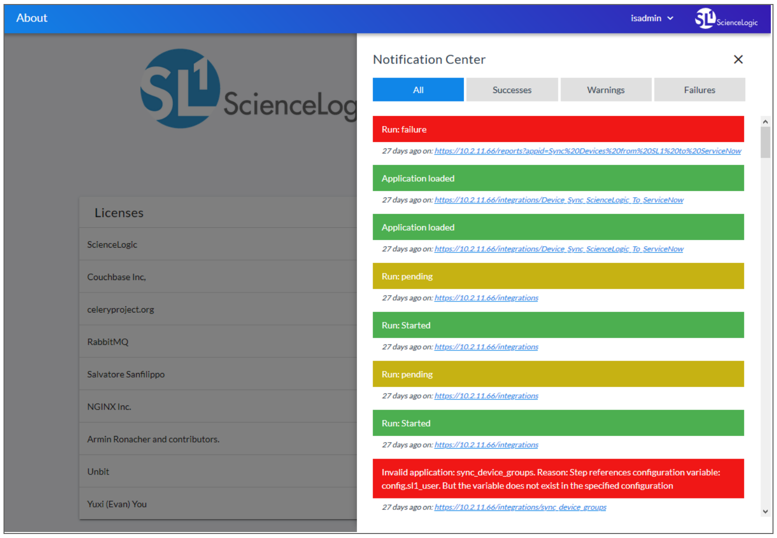
TIP: While the **SyncPacks**, **Integrations**, **Configurations**, **Reports** and **Admin Panel** pages are loading or running a procedure, you will see a dark green, horizontal line at the top of the page until the process completes.

Additional Navigation

The user name dropdown, which is found in the navigation bar in the top right of any window in the Integration Service user interface, contains the following options:

- *About*. Displays version information about the Integration Service version and the licenses used by the Integration Service.
- *Help*. Displays online Help for the Integration Service in a new browser window.

- *Notifications*. Opens the **Notification Center** pane, which contains a log of all previous notifications that appeared on the Integration Service system about integration applications that were run successfully or with warnings or failures:



Licenses

ScienceLogic

Couchbase Inc.

celeryproject.org

RabbitMQ

Salvatore Sanfilippo

NGINX Inc.

Armin Ronacher and contributors.

Unbit

Yuxi (Evan) You

Run: failure

27 days ago on: https://10.2.11.66/reports?appid=Sync%20Devices%20from%20SL1%20to%20ServiceNow

Application loaded

27 days ago on: https://10.2.11.66/integrations/Device_Sync_ScienceLogic_To_ServiceNow

Application loaded

27 days ago on: https://10.2.11.66/integrations/Device_Sync_ScienceLogic_To_ServiceNow

Run: pending

27 days ago on: https://10.2.11.66/integrations

Run: Started

27 days ago on: https://10.2.11.66/integrations

Run: pending

27 days ago on: https://10.2.11.66/integrations

Run: Started

27 days ago on: https://10.2.11.66/integrations

Invalid application: sync_device_groups. Reason: Step references configuration variable: config.sl1_user. But the variable does not exist in the specified configuration

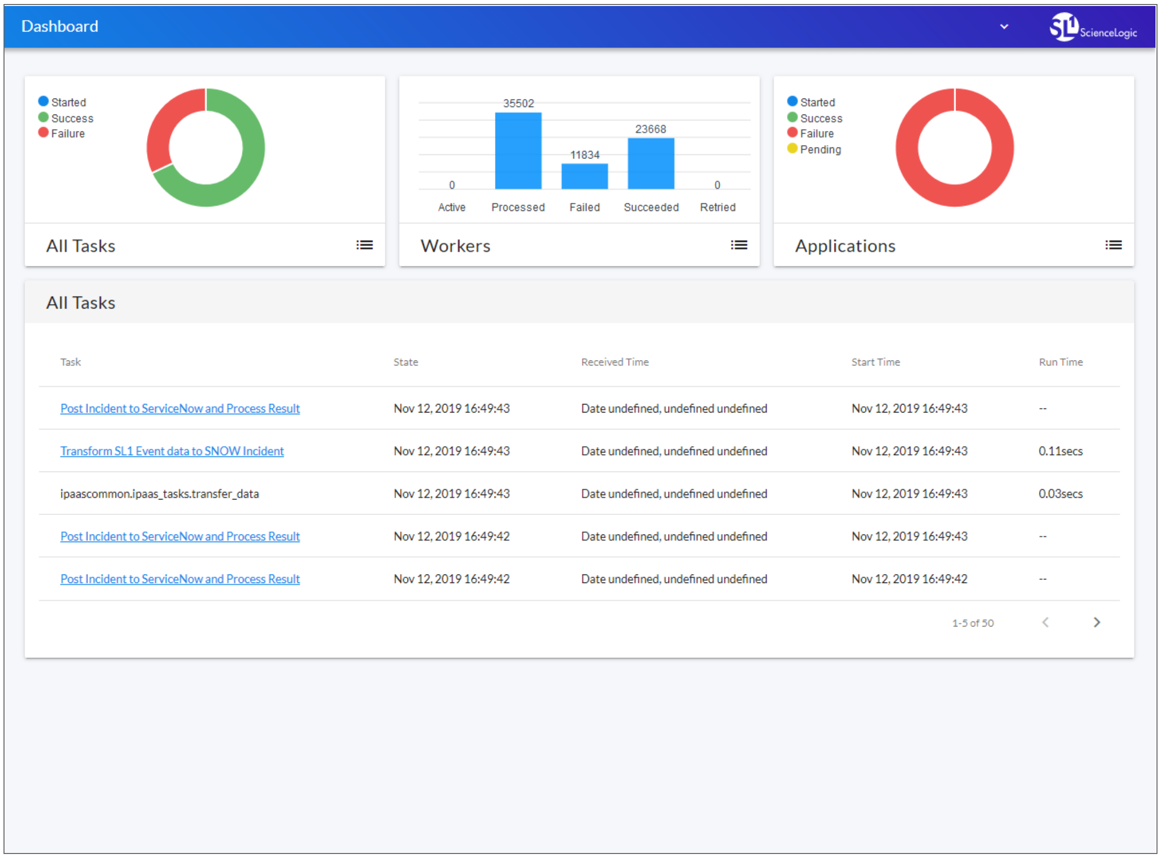
27 days ago on: https://10.2.11.66/integrations/sync_device_groups

The different notifications are color-coded: green for success, yellow for warning, and red for failure. The number of notifications displays as a badge in the menu. For more information about a notification, click the link for the page where the notification appeared and review the **Step Log** details for the application steps.

- *Log off*. Logs you out of the Integration Service user interface.

Monitoring the Integration Service on the Dashboard Page

You can use the **Dashboard** page (▦) in the Integration Service user interface to monitor the status of the various tasks, workers, and integration applications that are running on your Integration Service system. You can use this information to quickly determine if your Integration Service instance is performing as expected:



All Tasks

Task	State	Received Time	Start Time	Run Time
Post Incident to ServiceNow and Process Result	Nov 12, 2019 16:49:43	Date undefined, undefined undefined	Nov 12, 2019 16:49:43	--
Transform SL1 Event data to SNOW Incident	Nov 12, 2019 16:49:43	Date undefined, undefined undefined	Nov 12, 2019 16:49:43	0.11secs
ipaascommon.ipaas_tasks.transfer_data	Nov 12, 2019 16:49:43	Date undefined, undefined undefined	Nov 12, 2019 16:49:43	0.03secs
Post Incident to ServiceNow and Process Result	Nov 12, 2019 16:49:42	Date undefined, undefined undefined	Nov 12, 2019 16:49:43	--
Post Incident to ServiceNow and Process Result	Nov 12, 2019 16:49:42	Date undefined, undefined undefined	Nov 12, 2019 16:49:42	--

1-5 of 50

The **Dashboards** page is the initial landing page after you log in to the Integration Service. This page displays high-level statistics about the health of the worker services that are being used by the Integration Service instance.

To view more information on the **Dashboard** page:

1. Hover over a circle graph or a bar chart item to view a pop-up field that contains the count for that item on the graph or chart, such as "Success: 48" for successful tasks on the **All Tasks** graph.
2. Click the List icon (≔) for the **All Tasks**, **Workers**, or **Applications** graphs to view a list of relevant tasks, workers, or applications. Use the left and right arrow icons to move through the list of items.
3. To view additional details about a specific tasks, click the List icon (≔) for the **All Tasks** graph and then click the link for the task, where relevant. The integration application aligned with that task appears, and you can select a step and view the **Step Log** details for that step.

Chapter 2

Installing and Configuring the Integration Service

Overview

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

This chapter covers the following topics:

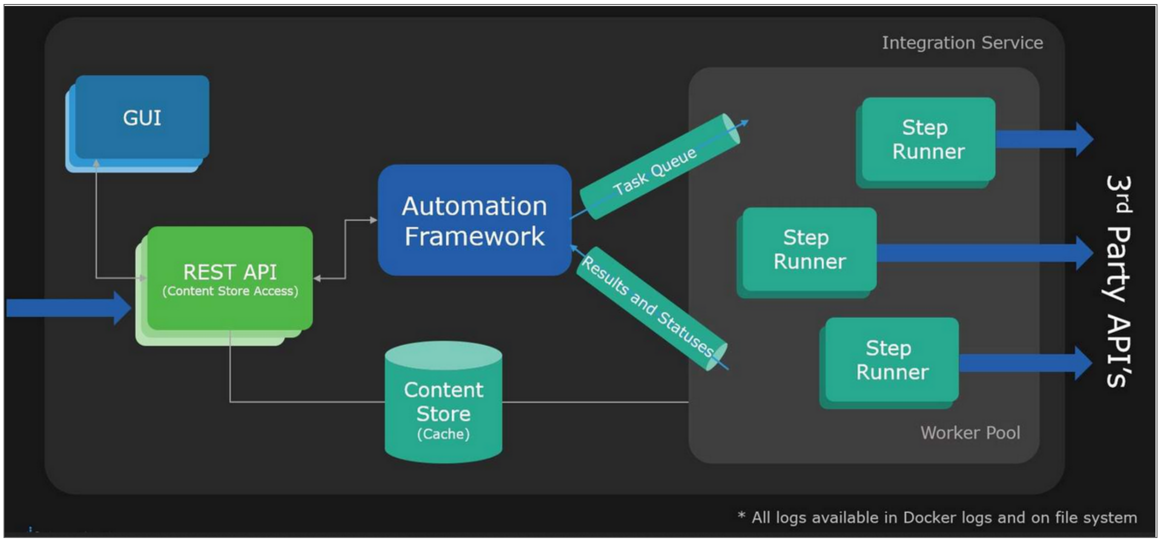
Integration Service Architecture

This topic describes the different aspects of the Integration Service architecture.

Integration Service Container Architecture

The Integration Service is a collection of purpose-built containers that are charged to pass information to and from SL1. Building the Integration Service architecture in containers allows you to add more processes to handle the workload as needed.

The following diagram describes the container architecture for the Integration Service:

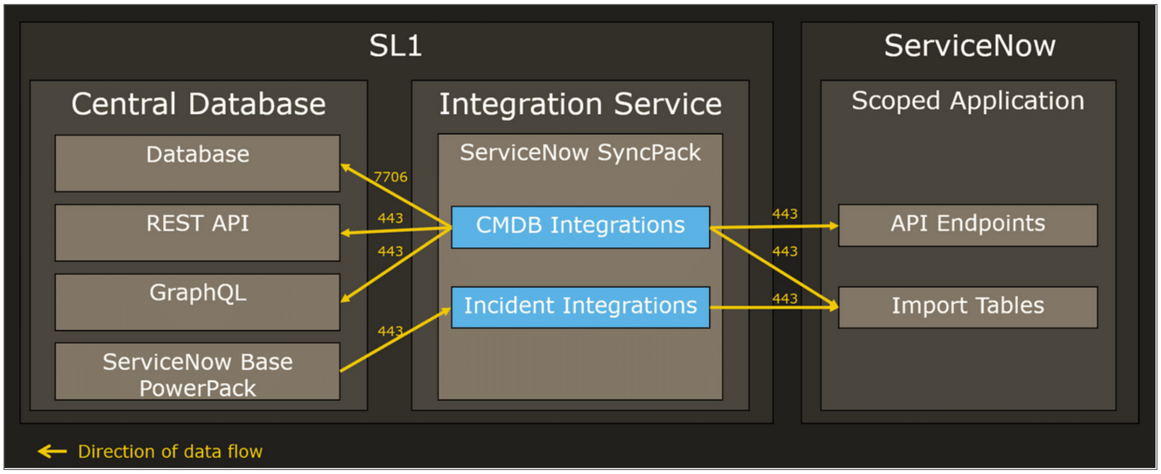


The Integration Service includes the following containers:

- ***GUI***. The GUI container provides the user interface for the Integration Service.
- ***REST API***. The REST API container provides access to the Content Store on the Integration Service instance.
- ***Content Store***. The Content Store container is basically a database service that contains all the reusable steps, integration applications, and containers in the Integration Service instance.
- ***Step Runners***. Step Runner containers execute steps independently of other Step Runners. All Step Runners belong to a Worker Pool and can run steps in order, based on the instructions in the integration applications. By default there are five Step Runners (worker nodes) include in the Integration Service platform. Integration Service users can scale up or scale down the number of worker nodes, based on the workload requirements.

Integration Workflow

The following high-level diagram for a ServiceNow Integration provides an example of how the Integration Service communicates with both the SL1 Central Database and the third-party (ServiceNow) APIs:



The workflow includes the following components and their communication methods:

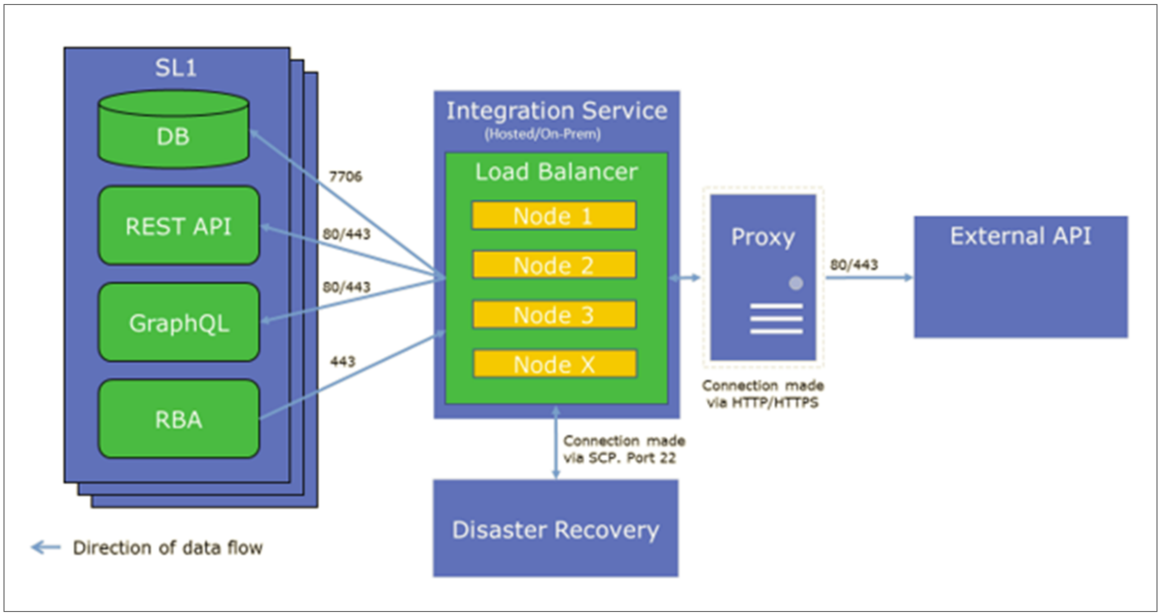
- ***SL1 Central Database***. The Integration Service communicates with the SL1 database over port 7706.
- ***SL1 REST API***. The Integration Service communicates with the SL1 REST API over port 443.
- ***GraphQL***. The Integration Service communicates with GraphQL over port 443.
- ***ServiceNow Base PowerPack***. In this example, the Run Book Automations from the ServiceNow Base PowerPack (and other SL1 PowerPacks) communicate with the Integration Service over port 443.
- ***Integration Service***. The Integration Service communicates with both the SL1 Central Database and an external endpoint.
- ***ServiceNow API***. In this example, the ServiceNow integration applications on the Integration Service communicate with the ServiceNow API over port 443.

NOTE: The Integration Service both pulls data from SL1 and has data pushed to it from SL1. The Integration Service both sends and retrieves information to and from ServiceNow, but the Integration Service is originating the requests.

High-Availability, Disaster Recovery, and Proxy Architecture

You can deploy the Integration Service as a High Availability cluster, which requires at least *three* nodes to achieve automatic failover. While the Integration Service *can* be deployed as a single node, the single-node option does not provide redundancy through High Availability. Integration Service also supports off-site backup and connection through a proxy server.

The following diagram describes these different configurations:



- ***High Availability*** for the Integration Service is a cluster of Integration Service nodes with a Load Balancer managing the workload. In the above scenario, if one Integration Service node fails, the workload will be redistributed to the remaining Integration Service nods. High Availability provides local redundancy. For more information, see *Configuring the Integration Service for High Availability*.
- ***Off-site Backup*** can be configured by using the Integration Service to back up and recover data in the Couchbase database. The backup process creates a backup file and sends that file using Secure Copy Protocol (SCP) to a user-defined, off-site destination system. You can then use the backup file from the remote system and restore its content. For more information, see *Backing up Data*.
- A ***Proxy Server*** is a dedicated computer or software system running as an intermediary. The proxy server in the above scenario handles the requests between the Integration Service and ServiceNow. For more information, see *Configuring a Proxy Server*.

Prerequisites for the Integration Service

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

- Linux and vi (or another text editor).
- Python.
- Postman or another API tool for interacting with the Integration Service API.
- Couchbase. For more information, see ***Helpful Couchbase Commands***.
- Docker. For more information, 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.

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. Thin provisioning is not supported.

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-SL1 stack support.

NOTE: The Integration Service itself does not have specific minimum required versions for SL1 or AP2. However, certain Integration Service Synchronization PowerPacks have minimum version dependencies. Please see the documentation for those Synchronization PowerPacks for more information on those dependencies.

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

The following table lists the port access required by the Integration Service and this Synchronization PowerPack:

Source IP	Integration Service Destination	Integration Service Source Port	Destination Port	Requirement
Integration Service	SL1 API	Any	TCP 443	SL1 API Access
Integration Service	ServiceNow API	Any	TCP 443	ServiceNow API Access
Integration Service	SL1 Database	Any	TCP 7706	SL1 Database Access

Hardened Operating System

The operating system for the Integration Service is pre-hardened by default, with firewalls configured only for essential port access and all services and processes running inside Docker containers, communicating on a secure, encrypted overlay network between nodes. Please refer to the table, above, for more information on essential ports.

You can apply additional Linux hardening policies or package updates as long as Docker and its network communications are operational.

NOTE: The Integration Service operating system is an Oracle Linux distribution, and all patches are provided within the standard Oracle Linux repositories. The patches are not provided by ScienceLogic.

Installing the Integration Service

You can install the Integration Service for the first time in the following ways:

- *Via ISO to a server on your network*
- *Via RPM to a cloud-based server*

If you are *upgrading* an existing version of the Integration Service, see ***Upgrading the Integration Service***.

Installing the Integration Service via ISO

Locating the ISO Image

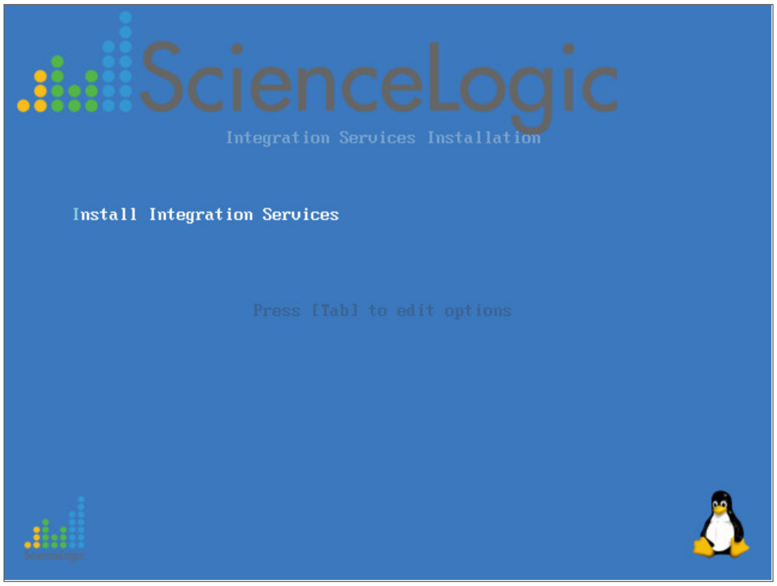
To locate the Integration Service ISO image:

1. Go to the ScienceLogic Support site at https://support.sciencelogic.com/s/.
2. Click the **Product Downloads** tab and select *Integration Service*. The **Integration Service Release Version** page appears.
3. Click the "Integration Service 2.0" link. The **Integration Service 2.0 Release Version** page appears.
4. In the **Release Files** section, click the "2.0.*x* ISO" link for the Integration Service image. A **Release File** page appears.
5. Click **[Download File]** at the bottom of the **Release File** page.

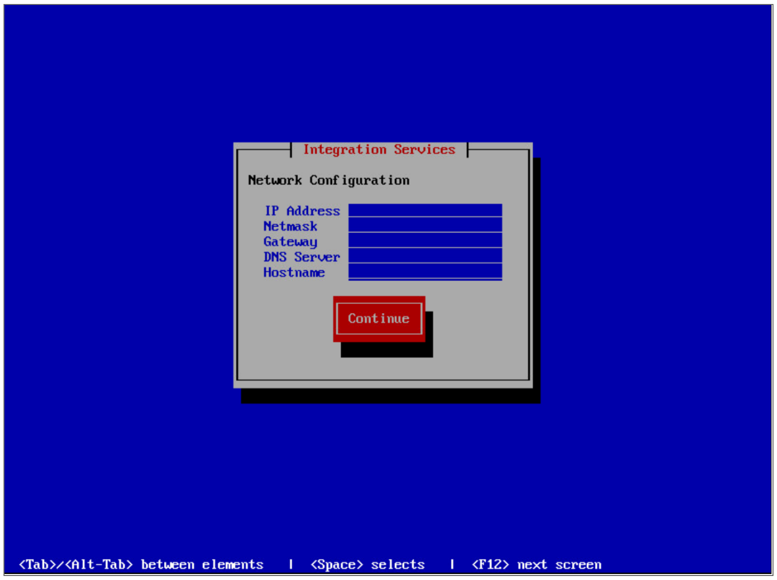
Installing from the ISO Image

To install the Integration Service via ISO image:

1. Download the latest Integration Service ISO file to your computer or a virtual machine center.
2. 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:


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

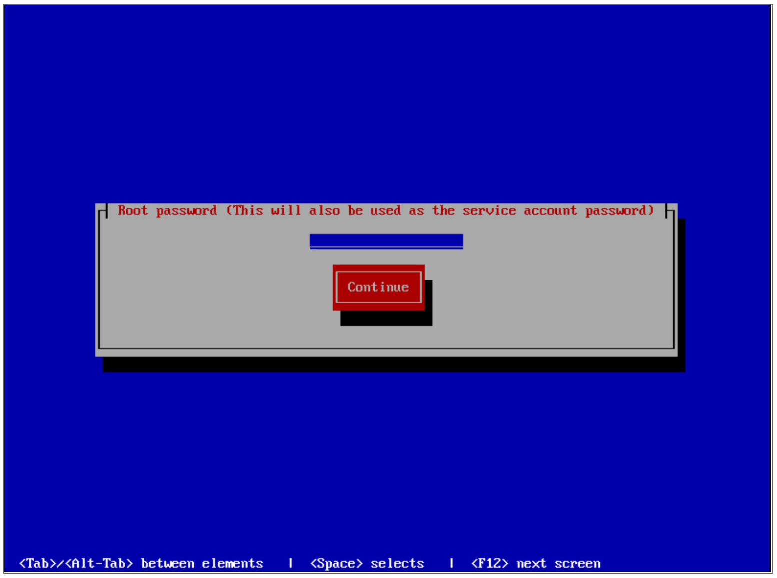
3. Select *Install Integration Service*. After the installer loads, the **Network Configuration** window appears:



4. Complete the following fields:

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

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

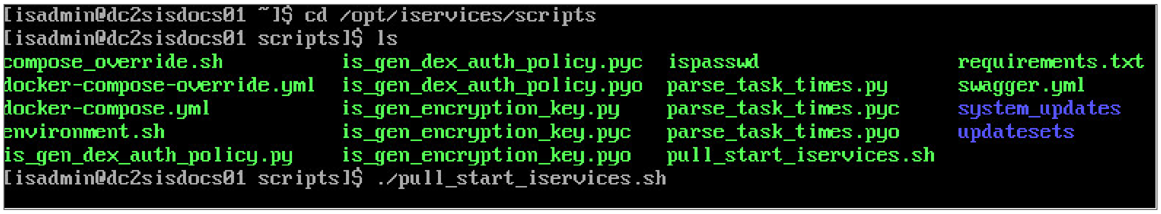

Root password (This will also be used as the service account password)
Continue
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen

6. Type the password you want to set for the root user on the Integration Service host and press **[Enter]**. The password must be at least six characters and no more than 24 characters, and all special characters are supported.

> NOTE: You use this password to log into the Integration Service user interface and to verify API requests and database actions. This password is set as both the "Linux host isadmin" user and in the **/etc/iservices/is_pass** file that is mounted into the Integration Service stack as a "Docker secret". Because it is mounted as a secret, all necessary containers are aware of this password in a secure manner. For more information, see *Changing the Integration Service Password*.

7. Type the password for the root user again and press **[Enter]**. The Integration Service installer runs, and the system reboots automatically. This process will take a few minutes.
8. After the installation scripts run and the system reboots, SSH into your system using PuTTY or a similar application. The default username for the system is *isadmin*.
9. To start the Docker services, change directory to **/opt/iservices/scripts** and run the following command:

```
./pull_start_iservices.sh
```



```
[isadmin@dc2sisdocs01 ~]$ cd /opt/iservices/scripts
[isadmin@dc2sisdocs01 scripts]$ ls
compose_override.sh           is_gen_dex_auth_policy.pyc  ispasswd                  requirements.txt
docker-compose-override.yml   is_gen_dex_auth_policy.pyo  parse_task_times.py       swagger.yml
docker-compose.yml            is_gen_encryption_key.py    parse_task_times.pyc      system_updates
environment.sh                is_gen_encryption_key.pyc   parse_task_times.pyo      updatesets
is_gen_dex_auth_policy.py     is_gen_encryption_key.pyo   pull_start_iservices.sh
[isadmin@dc2sisdocs01 scripts]$ ./pull_start_iservices.sh
```

10. To validate that iservices is running, run the following command to view each service and the service versions for services throughout the whole stack:

```
docker service ls
```

11. 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>
    ```

12. Log in with the default username of *isadmin* and the password you specified in step 6.
13. After installation, you must license your Integration Service system if you want to enable all of the features. For more information, see ***Licensing the Integration Service***.
14. If you are setting up High Availability for the Integration Service on a multiple-node cluster, see ***Preparing the Integration Service System for High Availability***.

> **NOTE:** The **HOST_ADDRESS** value in the **/etc/iservices/isconfig.yml** file should be the fully qualified domain name (FQDN) of either the host if there is no load balancer, or the FQDN of the load balancer if one exists. If you change the **HOST_ADDRESS** value, you will need to restart the Integration Service stack.

Troubleshooting the ISO Installation

To verify that your stack is deployed, view your Couchbase logs by executing the following command:

```
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
```

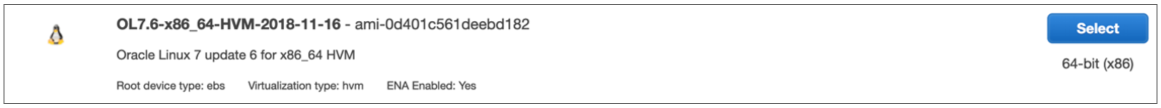
Installing the Integration Service via RPM

> **NOTE:** If you install the Integration Service on any operating system other than Oracle Linux 7, ScienceLogic will only support the running application and associated containers. ScienceLogic will not assist with issues related to host configuration for operating systems other than Oracle Linux 7.

> **NOTE:** The **HOST_ADDRESS** value in the **/etc/iservices/isconfig.yml** file should be the fully qualified domain name (FQDN) of either the host if there is no load balancer, or the FQDN of the load balancer if one exists. If you change the **HOST_ADDRESS** value, you will need to restart the Integration Service stack.

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

1. In Amazon Web Service (AWS) EC2, click **[Launch instance]**. The **Choose an Amazon Machine Image (AMI)** page appears.
2. Deploy a new Oracle Linux 7.6 virtual machine by searching for **OL7.6-x86_64-HVM** in the ***Search for an AMI*** field.
3. Click the ***results*** link for Community AMIs.
4. Click **[Select]** for a virtual machine running Oracle Linux 7.6 or greater for installation. The following image shows an example of an **OL7.6-x86_64-HVM-*** AMI file:



OL7.6-x86_64-HVM-2018-11-16 - ami-0d401c561deebd182

Oracle Linux 7 update 6 for x86_64 HVM

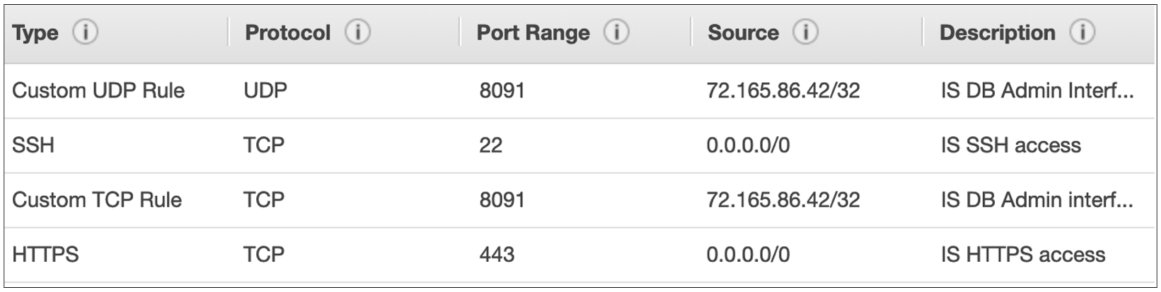
Root device type: ebs   Virtualization type: hvm   ENA Enabled: Yes

5. From the **Choose an Instance Type** page, select at least a t2.xlarge AMI instance, depending on your configuration:

   - *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 multi-tenant environment guide for more sizing information. ScienceLogic recommends that you allocate at least 50 GB or more for storage.

6. Go to the **Step 6: Configure Security Group** page and 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

7. Upload the **sl1-integration-services-2.x.x-1.x86_64.rpm** file using SFTP or SCP.
8. Ensure that the latest required packages are installed by running the following commands on the AMI instance:

```
sudo yum install -y wget
pip install docker-compose
wget https://download.docker.com/linux/centos/7/x86_64/stable/Packages/docker-ce-
19.03.5-3.el7.x86_64.rpm && sudo yum install docker-ce-19.03.5-3.el7.x86_64.rpm
```

> **NOTE:** You will need to update both instances of the Docker version in this command if there is a more recent version of Docker CE on the Docker Download page: https://download.docker.com/linux/centos/7/x86_64/stable/Packages/.

9. Create the Docker group:

```
sudo groupadd docker
```

10. Add your user to the Docker group:

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

11. Log out and log back in to ensure that your group membership is re-evaluated.
12. Run the following commands for the 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
```

13. Run the following 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
```

> **TIP:** To view a list of all ports, run the following command: `firewall-cmd --list-all`

14. Copy the Integration Service RPM to the AWS instance of installation and install the RPM:

```
sudo yum install sl1-integration-services
systemctl restart docker
```

15. Create a password for the Integration Service:

```
sudo printf <password> > /etc/iservices/is_pass
```

where `<password>` is a new, secure password.

16. Pull and start iservices to start the Integration Service:

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

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

NOTE: For a clustered Integration Service environment, you must install the Integration Service RPM on every server that you plan to cluster into the Integration Service. You can load the Docker images for the services onto each server locally by running `/opt/iservices/scripts/pull_start_iservices.sh`. Installing the RPM onto each server ensures that the Integration Service containers and necessary data are available on all servers in the cluster.

NOTE: After installation, you must license your Integration Service system to enable all of the features. Licensing is required for production systems only, not for test systems. For more information, see *Licensing the Integration Service*.

Upgrading the Integration Service

The process for upgrading to version 2.0.0 or later of the Integration Service includes the following required steps:

1. Upgrade the host packages and Python 3.6 (previous versions of the Integration Service used Python 2.6).
2. Upgrade to Oracle 7.3 or later.
3. Upgrade to Docker version 18.09.2 or later.

NOTE: Integration Service version 2.0.0 or later requires the **docker-ce 18.09.2** or later version of Docker. The Integration Service ISO installs the **docker-ce 19.03.5** version of Docker by default, but if you are upgrading to this version from the RPM, you must upgrade Docker *before* you upgrade the Integration Service with the RPM.

4. Install the Integration Service 2.0.*x* upgrade RPM.
5. Update the Integration Service system from Basic Authentication to OAuth 2.0. For more information, see ***Configuring Authentication with the Integration Service***.
6. Set up licensing for the Integration Service. You must license your Integration Service system to enable all of the features. If you are not deploying the Integration Service on a production or pre-production environment, you can skip licensing. "For more information, see ***Licensing the Integration Service***.

Please note that upgrading to version 2.0.0 or later *will* involve some downtime of the Integration Service. Before upgrading your version of the Integration Service, ScienceLogic recommends that you make a backup of your Integration Service. For more information, see ***Backing up Data***.

You can upgrade the Integration Service from any 1.8.x version to the 2.0.0 or later version. If you upgrade from 1.8.3 or later, you can simply follow the steps below. However, if you are upgrading from a version before 1.8.3, be sure to review the release notes for the older version for any relevant update considerations before upgrading. For example, there is a small port change that you might need to apply if you are upgrading a customized cluster from a version of the Integration Service before version 1.8.3.

WARNING: If you have made any customizations to default integration applications or steps that shipped with previous versions of the Integration Service, you will need to make those customizations compatible with Python 3.6 or later *before* upgrading to version 2.0.0 or later of the Integration Service. Also, if you made any modifications to the nginx configuration or to other service configuration files outside of the **docker-compose.yml** file, you will need to modify or back up those custom configurations before upgrading, or contact ScienceLogic Support to prevent the loss of those modifications.

Locating and Running the Upgrade Script

You can perform the upgrade steps manually, or you can run the **is_upgrade_to_2.0.x.sh** script to perform the upgrade steps automatically. The script upgrades the Integration Service system from 1.8.x to 2.0.*x*.

To locate the upgrade script:

1. Go to the ScienceLogic Support site at https://support.sciencelogic.com/s/.
2. Click the **Product Downloads** tab and select *Integration Service*. The **Integration Service Release** page appears.
3. Click the "Integration Service 2.0" link. The **Integration Service 2.0 Release Version** page appears.
4. In the **Release Files** section, click the "1.8.X to 2.0.*x* Upgrade" link for the script. A **Release File** page appears.
5. Click **[Download File]** at the bottom of the **Release File** page. The **is_upgrade_to_2.0.x.sh** script is in the **is_upgrade_tools.zip** file.

The upgrade script runs the following steps:

1. Checks to see if the Oracle version is greater or equal to 7.3. If not, the script stops and displays a message that you need to update to Oracle 7.3 or later.
2. Sets the requirements location and either mounts the ISO or verifies that the RPM exists. For this step, the script asks if the installation will be offline or online, and it also asks you for the location of the RPM.
3. Installs Python 3.6.
4. Installs Docker 19.03.5.
5. Installs the Integration Service RPM.
6. Runs the **pull_start_iservices.sh** script to deploy and initialize the Integration Service.
7. If the upgrading process was offline, cleans the changes made for the upgrading process, unmounts the ISO, and removes the **localiso.repo** file.

To run the upgrade script:

1. Download the **is_upgrade_to_2.0.x.sh** script and add it to a directory on the Integration Service system.
2. Download the **sl1-integration-services-2.0.x-1.x86_64.rpm** file or the ISO file and add it to a directory on the Integration Service system. Make a note of this directory, because you will need it for Step 2 in the script.

TIP: Alternately, instead of downloading the RPM file, you can specify an online location for the RPM file.

3. If needed, run the following command on the Integration Service system to give the script execution permissions:

```
sudo chmod +x is_upgrade_to_2.0.0.sh
```

NOTE: If you are installing a version of the Integration Service later than 2.0.0, you will need to update the version number in the command, above.

4. Change directory to the directory containing the **is_upgrade_to_2.0.x.sh** script, such as **/home/isadmin/**, and then run the following command to execute the upgrade script:

```
sudo ./is_upgrade_to_2.0.0.sh
```

NOTE: If you are installing a version of the Integration Service later than 2.0.0, you will need to update the version number in the command, above.

5. For Step 2 of the script, you will need to specify if you want to run the upgrade online, or run it offline. Type "1" if you have Internet access, or "2" if you want to run the update offline.

6. For Step 2 of the script, you will need to specify the location of the RPM or ISO file for 2.0.x. You can use a location on the Integration Service system for the RPM or ISO file, or an online location for the RPM. For example: **/home/isadmin/sl1-integration-services-2.0.0-1.x86_64.rpm**
7. When the script completes, you can view updates to each service and the service versions for services throughout the whole stack, type the following at the command line:

   ```
   docker service ls
   ```

8. As needed, update the Integration Service system from Basic Authentication to OAuth 2.0. For more information, see ***Configuring Authentication with the Integration Service***.
9. As needed, set up licensing for the Integration Service. For more information, see ***Licensing the Integration Service***.

Upgrading Manually

Instead of running the upgrade script, you can perform the upgrade steps manually, following the detailed instructions below.

Step 1. Upgrading Host Packages and Python 3.6

To access the host packages online:

1. To make sure that all repositories can access the required host-level packages, enable the necessary repositories by running the following commands on the Integration Service system:

   ```
     yum install yum-utils
     yum-config-manager --enable ol7_latest
     yum-config-manager --enable ol7_optional_latest
   ```

2. Run the following commands to update and install the host-level packages, and to upgrade to Python 3.6:

   ```
     yum remove python34-pip python34-setuptools python3
     yum --setopt=obsoletes=0 install python36-pip python36 python36-setuptools
     python36-devel openssl-devel gcc make kernel
     yum update
   ```

3. Continue the upgrade process by upgrading to Oracle 7.3 or later.

If you need to upgrade the host packages offline, without Internet access, you can mount the latest Integration Service ISO file onto the system and create a yum repository configuration that points to the local mount point in **/etc/yum.repos.d**. After the ISO is mounted, you can import the latest GNU Privacy Guard (GPG) key used by the repository.

To access the host packages offline:

1. Mount the Integration Service ISO onto the system:

   ```
   mount -o loop /dev/cdrom /mnt/tmpISMount
   ```

2. After you mount the ISO, add a new repository file to access the ISO as if it were a yum repository. Create a **/etc/yum.repos.d/localiso.repo** file with the following contents:

   ```
   [localISISOMount]
   name=Locally mounted IS ISO for packages
   enabled=1
   baseurl=file:///mnt/tmpISMount
   gpgcheck=0
   ```

   After you create and save this file, the Linux system can install packages from the Integration Service ISO.

3. Optionally, you can import the latest GNU Privacy Guard (GPG) key to verify the packages by running the following command:

   ```
   rpm --import /mnt/repo_keys/RPM-GPG-KEY-Oracle
   rpm --import /mnt/tmpISMount/repo_keys/RPM-GPG-KEY-Docker-ce
   rpm --import /mnt/tmpISMount/repo_keys/RPM-GPG-KEY-EPEL-7
   ```

4. If you cannot install Docker or Python offline, delete the other repository references by running the following command (ScienceLogic recommends that you back up those file first):

   ```
   rm -rf /etc/yum.repos.d/epel.repo /etc/yum.repos.d/epel-testing.repo
   /etc/yum.repos.d/public-yum-ol7.repo
   ```

5. Run the following commands to update and install the host-level packages, and to upgrade to Python 3.6:

   ```
   yum remove python34-pip python34-setuptools python3
   yum --setopt=obsoletes=0 install python36-pip python36 python36-setuptools
   python36-devel openssl-devel gcc make kernel
   yum update
   ```

6. Continue the upgrade process by upgrading to Oracle 7.3 or later.

Step 2. Upgrading to Oracle 7.3 or Later

ScienceLogic recommends that you update the version of Oracle Linux running on the Integration Service to 64-bit version 7.3 or later.

NOTE: If you want to upgrade to Oracle Linux to 7.6, see https://docs.oracle.com/en/operating-systems/oracle-linux/7/relnotes7.6/.

To upgrade to Oracle 7.3 or later:

1. To check the current version of Oracle Linux on your Integration Service system, run the following command on the Integration Service system:

   ```
   cat /etc/oracle-release
   ```

2. To install Oracle, choose one of the following procedures:

  - Using a public Oracle Linux repository, run the `yum update` command.
  - By mounting the latest Integration Service ISO to the system and installing the latest packages from the ISO.

3. Continue the upgrade process by updating Docker.

Step 3. Upgrading to Docker 18.09.2 or later

Integration Service systems before version 2.0.0 included Docker 18.06. If you are running a version of the Integration Service before version 2.0.0, you will need to update Docker 18.09.2 or later to be able to upgrade to Integration Service version 2.0.0 or later. For more information about the security updates included in Docker 18.09.2, see https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-5736.

Before upgrading Docker, ScienceLogic recommends that you review the following information from the Docker product manual: https://docs.docker.com/ee/upgrade/.

NOTE: Run the following process on Docker Swarm node, starting with the manager nodes.

WARNING: For clustered configurations, see the information in ***Installing Docker in Clustered Configurations*** before running the upgrade steps below.

To upgrade to Docker 18.09.2 or later:

1. Review the steps in the Docker product manual: https://docs.docker.com/ee/docker-ee/oracle/#install-with-a-package.
2. To install docker-io, run the following command on the Integration Service instance:

```
sudo yum install -y https://download.docker.com/linux/centos/7/x86_
64/stable/Packages/containerd.io-1.2.6-3.3.el7.x86_64.rpm
https://download.docker.com/linux/centos/7/x86_64/stable/Packages/docker-ce-
19.03.5-3.el7.x86_64.rpm
https://download.docker.com/linux/centos/7/x86_64/stable/Packages/docker-ce-cli-
19.03.5-3.el7.x86_64.rpm
```

3. To install docker-ce offline using the IS 2.0.*x* ISO, run the following commands:

```
rpm -e --nodeps docker-ce
yum install -y /mnt/tmpISMount/Packages/containerd.io-1.2.6-3.3.el7.x86_64.rpm
yum install -y /mnt/tmpISMount/Packages/docker-ce-cli-19.03.5-3.el7.x86_64.rpm
yum install -y /mnt/tmpISMount/Packages/docker-ce-19.03.5-3.el7.x86_64.rpm
systemctl enable docker
systemctl start docker
```

NOTE: If the node is a member of a cluster, wait for a few minutes, and the node will automatically rejoin the swarm cluster, and re-deploy the services running on that node. Wait until all services are operational, then proceed to upgrade the next node. For more information, see ***Installing Docker in Clustered Configurations.***

3. Continue the upgrade process by ***installing the Integration Service RPM***.

Installing Docker in Clustered Configurations

Follow the best practices for upgrading a cluster described in the Docker product manual: https://docs.docker.com/ee/upgrade/#cluster-upgrade-best-practices.

NOTE: You should upgrade all manager nodes before upgrading worker nodes. Upgrading manager nodes sequentially is recommended if live workloads are running in the cluster during the upgrade. After you upgrade the manager nodes, you should upgrade worker nodes, and then the Swarm cluster upgrade is complete.

Docker recommends that you drain manager nodes of any services running on those nodes If a live migration is expected, all workloads must be running on swarm workers, not swarm managers, or the manager under upgrade needs to be completely drained.

Also, a new Python Package Index (PyPI) service was added to the Integration Service stack. When deploying the Integration Service in a cluster setup, and not using network-aware volumes, the PyPI server must be "pinned" to a specific node with constraints. Pinning the PyPI server to a single node ensures that its persistent volume containing the Synchronization PowerPacks will always be available to the Integration Service.

Step 4. Installing the Integration Service RPM

To update the Integration Service:

1. Download the Integration Service RPM and copy the RPM file to the Integration Service system.
2. Log in as *isadmin* with the appropriate (root) password. You must be root to upgrade the RPM file.
3. 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.

4. Run the pull start script to deploy and initialize 2.0.*x*:

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

5. To view updates to each service and the service versions for services throughout the whole stack, type the following at the command line:

```
docker service ls
```

6. Verify that each service now uses the new version of the Integration Service.
7. As needed, update the Integration Service system from Basic Authentication to OAuth 2.0. For more information, see *Configuring Authentication with the Integration Service*.
8. As needed, set up licensing for the Integration Service. For more information, see *Licensing the Integration Service*.

Troubleshooting Upgrade Issues

The following topics describe issues that might occur after the upgrade to version 2.0.0 or later, and how to address those issues.

To roll back to a version before Integration Service 2.0.0 or later

After a schedule is accessed or modified on the 2.0.0 or later Integration Service API or scheduler, that schedule will not be accessible again in a 1.x version. If you upgrade to 2.0.0 from 1.x and you want to go back to the 1.x release, you must delete the schedule and recreate the schedule in 1.x (if the schedule was modified in 2.0.0).

Cannot access the Integration Service or an Internal Server Error occurs

Integration Service version 2.0.0 and later use a new type of authentication session. This change might cause problems if your browser attempts to load the Integration Service user interface using a "stale" cache from version 1.8.4. If you have issues accessing the user interface, or if you see an "Internal server error" message when you log in, be sure to clear the local cache of your browser.

After upgrading, the syncpack_steprunner fails to run

This error flow tends to happen when the syncpack_steprunner is deployed, but the database is not yet updated with the indexes necessary for the Synchronization PowerPack processes to query the database. In most deployments, the index should be automatically created. If the index is not automatically created, which it might do in a clusterd configuration, you can resolve this issue by manually creating the indexes.

In this situation, if you check the logs, you will most likely see the following message:

```
couchbase.exceptions.HTTPError: <RC=0x3B[HTTP Operation failed. Inspect status code
for details], HTTP Request failed. Examine 'objextra' for full result, Results=1, C
Source=(src/http.c,144), OBJ=ViewResult<rc=0x3B[HTTP Operation failed. Inspect status
code for details], value={'requestID': '57ad959d-bafb-46a1-9ede-f80f692b0dd7',
'errors': [{'code': 4000, 'msg': 'No index available on keyspace content that matches
your query. Use CREATE INDEX or CREATE PRIMARY INDEX to create an index, or check that
your expected index is online.'}], 'status': 'fatal', 'metrics': {'elapsedTime':
'5.423085ms', 'executionTime': '5.344487ms', 'resultCount': 0, 'resultSize': 0,
'errorCount': 1}}, http_status=404, tracing_context=0, tracing_output=None>, Tracing
Output={":nokey:0": null}>
```

To address this issue, wait a few minutes for the index to be populated. If you are still getting an error after the database has been running for a few minutes, you can manually update the indexes by running the following command:

```
initialize_couchbase -s
```

Licensing for the Integration Service

Before users can access all of the features of version 2.0.0 or later of the Integration Service, the *Administrator* user must license the Integration Service instance through the ScienceLogic Support site. For more information about accessing Integration Service files at the ScienceLogic Support site, see the following Knowledge Base article: SL1 Integration Service Download and Licensing.

NOTE: The administrator and all users cannot access certain production-level capabilities until the administrator licenses the instance. For example, users cannot create schedules or upload integration applications and steps that are not part of a Synchronization PowerPack until the Integration Service has been licensed.

TIP: If you are not deploying the Integration Service on a production or pre-production environment, you can skip the licensing process.

NOTE: If you are licensing an Integration Service High Availability cluster, you can run the following licensing process on *any* node in the cluster. The node does not have to be the leader, and the licensing process does not have to be run on all nodes in the Swarm.

To license an Integration Service system:

1. Run the following command on your Integration Service system to generate the **.iskey** license file:

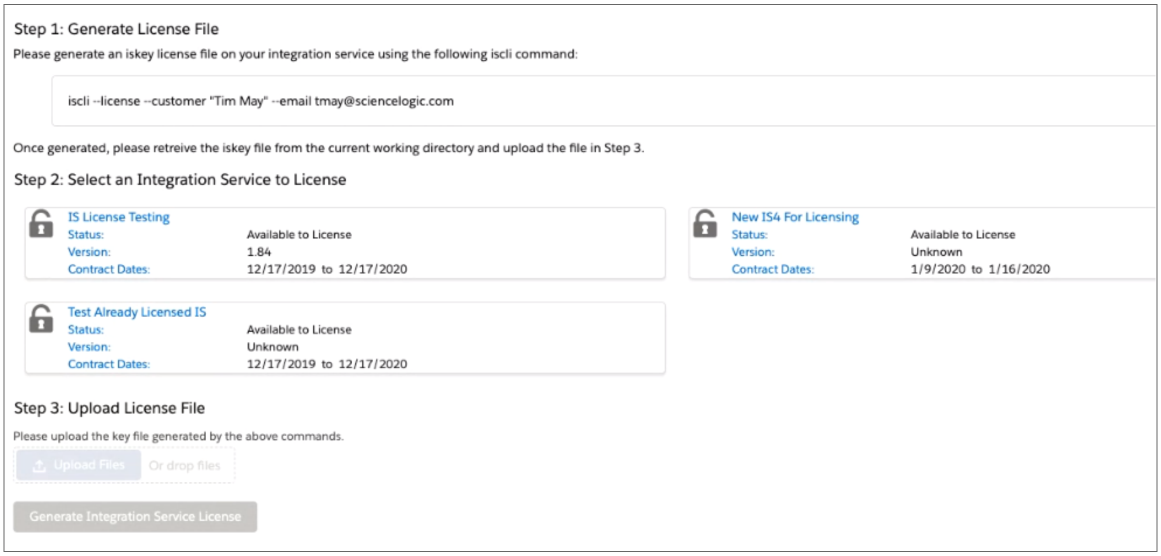
   ```
   iscli --license --customer "<user_name>" --email <user_email>
   ```

   where `<user_name>` is the first and last name of the user, and `<user_email>` is the user's email address. For example:

   ```
   iscli --license --customer "John Doe" --email jdoe@sciencelogic.com
   ```

2. Run an `ls` command to locate the new license file: **customer_key.iskey**.
3. Using WinSCP or another utility, copy the **.iskey** license file to your local machine.
4. Go to the **Integration Service License Request** page at the ScienceLogic Support site: https://support.sciencelogic.com/s/integration-service-license-request.

5. Click the relevant license link. The **Generate License File** page appears:



Step 1: Generate License File

Please generate an iskey license file on your integration service using the following iscli command:

```
iscli --license --customer "Tim May" --email tmay@sciencelogic.com
```

Once generated, please retreive the iskey file from the current working directory and upload the file in Step 3.

Step 2: Select an Integration Service to License

IS License Testing
Status: Available to License
Version: 1.84
Contract Dates: 12/17/2019 to 12/17/2020

New IS4 For Licensing
Status: Available to License
Version: Unknown
Contract Dates: 1/9/2020 to 1/16/2020

Test Already Licensed IS
Status: Available to License
Version: Unknown
Contract Dates: 12/17/2019 to 12/17/2020

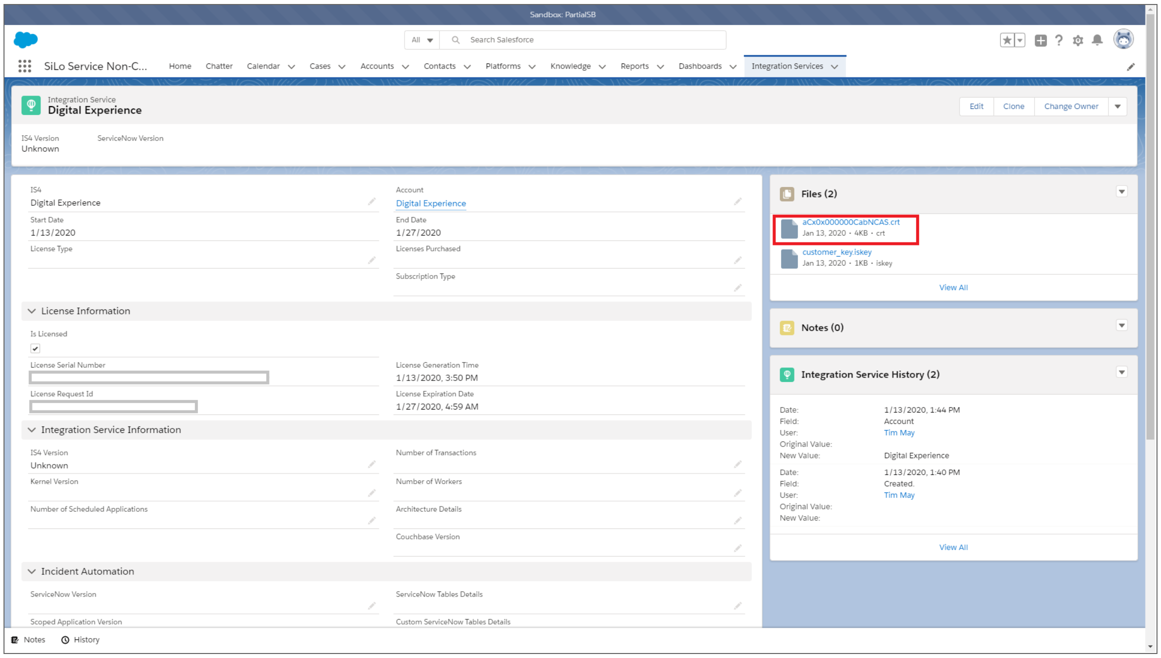
Step 3: Upload License File

Please upload the key file generated by the above commands.

TIP: You already covered Step 1 of the "Generate License File" process in steps 1-3 of this procedure.

6. For Step 2 of the "Generate License File" process, select the type of Integration Service you are using.
7. For Step 3 of the "Generate License File" process, upload the **.iskey** license file you created in steps 1-3 of this procedure and click **[Upload File]**.

8. After uploading the license file, click **[Generate Integration Service License]**. A new **Licensing** page appears:



Integration Service
Digital Experience

IS4 Version: Unknown
ServiceNow Version:

IS4: Digital Experience
Account: Digital Experience
Start Date: 1/13/2020
End Date: 1/27/2020
License Type:
Licenses Purchased:
Subscription Type:

License Information

Is Licensed: ☑
License Serial Number:
License Generation Time: 1/13/2020, 3:50 PM
License Request Id:
License Expiration Date: 1/27/2020, 4:59 AM

Integration Service Information

IS4 Version: Unknown
Number of Transactions:
Kernel Version:
Number of Workers:
Number of Scheduled Applications:
Architecture Details:
Couchbase Version:

Incident Automation

ServiceNow Version:
ServiceNow Tables Details:
Scoped Application Version:
Custom ServiceNow Tables Details:

Files (2)

aCx0x000000CabNCAS.crt
Jan 13, 2020 • 4KB • crt

customer_key.iskey
Jan 13, 2020 • 1KB • iskey

View All

Notes (0)

Integration Service History (2)

Date: 1/13/2020, 1:44 PM
Field: Account
User: Tim May
Original Value:
New Value: Digital Experience

Date: 1/13/2020, 1:40 PM
Field: Created.
User: Tim May
Original Value:
New Value:

View All

9. Click the **.crt** file in the **Files** pane to download the new **.crt** license file.

10. Using WinSCP or another file-transfer utility, copy the **.crt** license file to your Integration Service system.

11. Upload the **.crt** license file to the Integration Service server by running the following command on that server:

```
iscli -l -u -f ./<license_name>.crt -H <IP_address> -U <user_name> -p <user_
password>
```

where `<license_name>` is the system-generated name for the **.crt** file, `<IP_address>` is the IP address of the Integration Service system, `<user_name>` is the user name, and `<user_password>` is the user password. For example:

```
iscli -l -u -f ./aCx0x000000CabNCAS.crt -H 10.2.33.1 -U isadmin -p passw0rd
```

NOTE: ScienceLogic determines the duration of the license key, not the customer.

TIP: If you have any issues licensing your Integration Service system, please contact your ScienceLogic Customer Success Manager (CSM) or open a new Service Request case under the "Integration Service" category.

Configuring Additional Elements of 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'
```

Changing the Integration Service Password

The Integration Service uses two primary passwords. For consistency, both passwords are the same after you install the Integration Service, but you can change them to separate passwords as needed.

The Integration Service uses the following passwords:

- ***The Integration Service Administrator (isadmin) user password***. This is the password that you set during the Integration Service ***ISO installation process***, and it is only used by the default local Administrator user (*isadmin*). You use this password to log into the Integration Service user interface and to verify API requests and database actions. This password is set as both the "Linux host isadmin" user and in the **/etc/iservices/is_pass** file that is mounted into the Integration Service stack as a "Docker secret". Because it is mounted as a secret, all necessary containers are aware of this password in a secure manner. Alternatively, you can enable third-party authentication, such as LDAP or AD, and authenticate with credentials other than isadmin. However, you will need to set the user policies for those LDAP users first with the default isadmin user. For more information, see ***Managing Users in the Integration Service***.
- ***The Linux Host OS SSH password***. This is the password you use to SSH and to log in to isadmin. You can change this password using the standard Linux `passwd` command or another credential management application to manage this user. You can also disable this Linux user and add your own user if you want. The Integration Service containers and integration applications do not use or know this Linux login password, and this password does not need to be the same between nodes in a cluster. This is a standard Linux Host OS password.

To change the Integration Service Administrator (*isadmin*) user password:

1. You can change the mounted isadmin password secret (which is used to authenticate via API by default) and the Couchbase credentials on the Integration Service stack by running the `ispasswd` script on any node running Integration Service in the stack:

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

2. Follow the prompts to reset the password. The password must be at least six characters and no more than 24 characters, and all special characters are supported.

NOTE: Running the `ispasswd` script automatically changes the password for all Integration Service application actions that require credentials for the isadmin user.

3. If you have multiple nodes, copy **/etc/iservices/is_pass** file, which was just updated by the `ispasswd` script, to all other manager nodes in the cluster. You need to copy this password file across all nodes in case you deploy from a different node than the node where you changed the password. The need to manually copy the password to all nodes will be removed in a future release of the Integration Service.

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**.

> **NOTE:** The Integration Service uses a **docker-compose-override.yml** file to persistently store user-specific configurations for containers, such as proxy settings, replica settings, additional node settings, and deploy constraints. The user-specific changes are kept in this file so that they can be re-applied when the **/opt/iservices/docker-compose.yml** file is completely replaced on an RPM upgrade, ensuring that no user-specific configurations are lost.

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: ".isnet" (no *)
```

5. In the **environment** section of the **syncpack_steprunner** service, add the following lines:

```
services:
  syncpack_steprunner:
    environment:
       https_proxy: "<proxy_host>"
       http_proxy: "<proxy_host>"
       no_proxy: ".isnet" (no *)
```

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

> **NOTE:** The **compose_override.sh** script validates that the configured **docker-compose.yml** and **docker-compose-override.yml** files are syntactically correct. If the settings are correct, the script applies the settings to your existing **docker-compose.yml** file that is used to actually deploy.

7. 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**

NOTE: If you are using a clustered configuration for the Integration Service, you will need to copy the key and certificate to the same location on each of the nodes.

To specify the HTTPS certificate to use in your environment:

1. Copy 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 code is an example of the volume specification:

```
volumes:
  - "<path to IS key>:/etc/iservices/is_key.pem"
  - "<path to IS certificate>:/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.

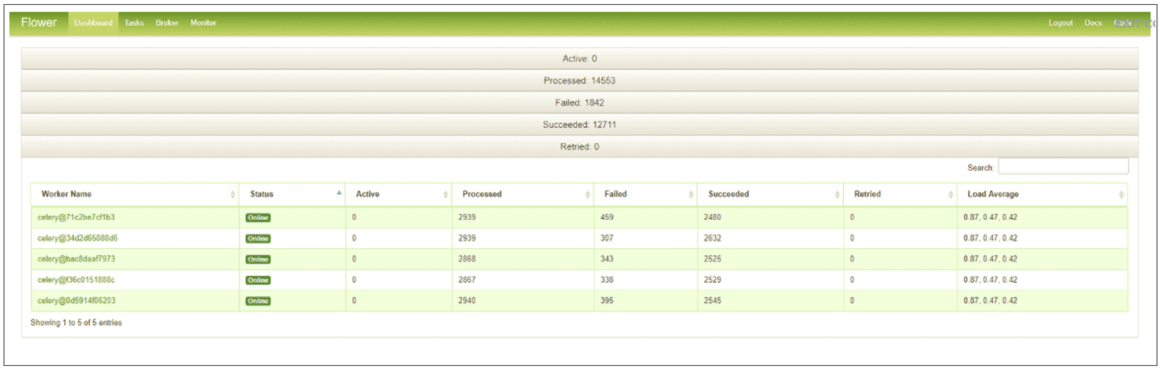
TIP: The Integration Service supports all special characters in passwords.

Integration Service Management Endpoints

This section provides technical details about managing the Integration Service. The following information is also available in the PowerPacks in ***Using SL1 to Monitor the Integration Service***.

Flower API

Celery Flower is a web-based tool for monitoring Integration Service tasks and workers. Flower lets you see task progress, details, and worker status:



Active: 0

Processed: 14553

Failed: 1842

Succeeded: 12711

Retried: 0

Worker Name	Status	Active	Processed	Failed	Succeeded	Retried	Load Average
celery@71c2be7cf1b3	Online	0	2939	459	2480	0	0.87, 0.47, 0.42
celery@34d2d65888d6	Online	0	2939	307	2632	0	0.87, 0.47, 0.42
celery@bac8daaf7973	Online	0	2868	343	2525	0	0.87, 0.47, 0.42
celery@f36c0151888c	Online	0	2867	338	2529	0	0.87, 0.47, 0.42
celery@0d5914f06203	Online	0	2940	395	2545	0	0.87, 0.47, 0.42

Showing 1 to 5 of 5 entries

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`.

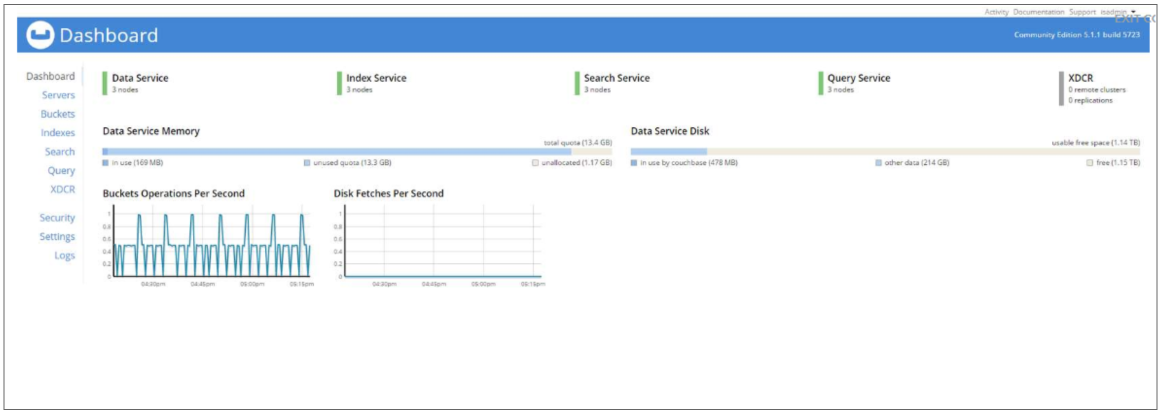
For more information, see the Flower API Reference.

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. For more information, see ***Using SL1 to Monitor the Integration Service***.

Couchbase API

The Couchbase Server is an open-source database software that can be used for building scalable, interactive, and high-performance applications. Built using NoSQL technology, Couchbase Server can be used in either a standalone or cluster configuration.

The following image shows the CouchBase user interface, which you can access at port 8091:



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`.

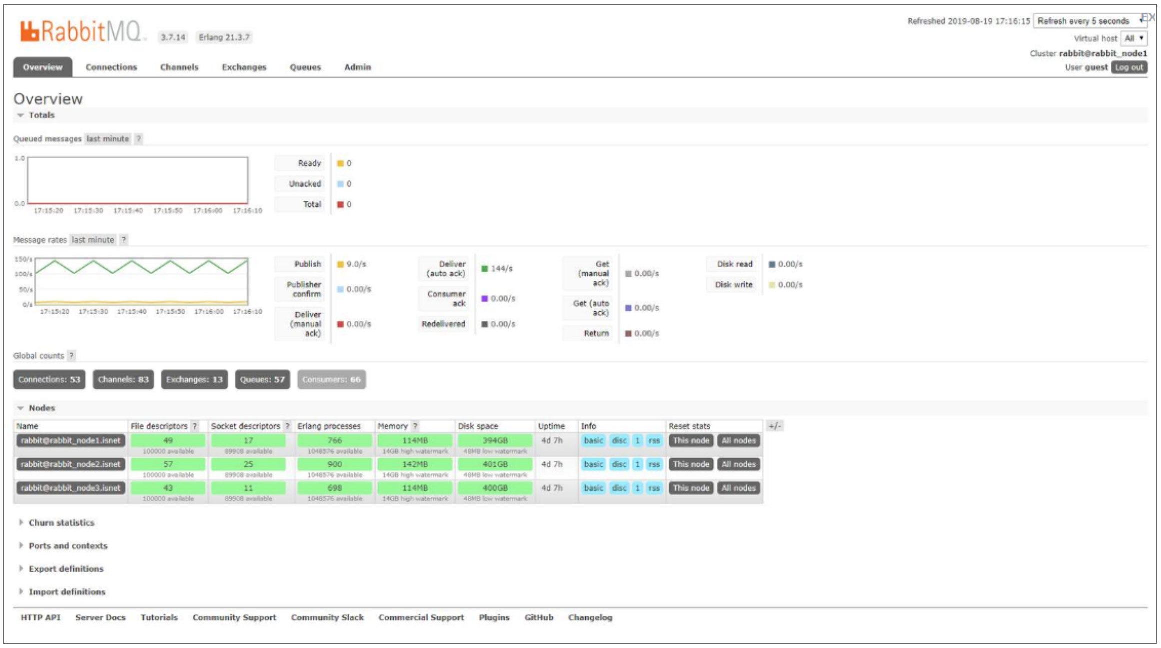
For more information, see the Couchbase API Reference.

> **NOTE:** You can also use the Couchbase PowerPack to collect this information. For more information, see *Using SL1 to Monitor the Integration Service*.

RabbitMQ

RabbitMQ is an open-source message-broker software that originally implemented the Advanced Message Queuing Protocol and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol, Message Queuing Telemetry Transport, and other protocols.

The following image shows the RabbitMQ user interface, which you can access at port 15672:



Refreshed 2019-08-19 17:16:15 Refresh every 5 seconds

Virtual host All

Cluster **rabbit@rabbit_node1**

User **guest**

Overview

Totals

Queued messages last minute ?

Ready 0
Unacked 0
Total 0

Message rates last minute ?

Publish 9.0/s
Publisher confirm 0.00/s
Deliver (manual ack) 0.00/s
Deliver (auto ack) 144/s
Consumer ack 0.00/s
Redelivered 0.00/s
Get (manual ack) 0.00/s
Get (auto ack) 0.00/s
Return 0.00/s
Disk read 0.00/s
Disk write 0.00/s

Global counts ?

Connections: 53 Channels: 83 Exchanges: 13 Queues: 57 Consumers: 66

Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats
rabbit@rabbit_node1.isnet	49 100000 available	17 89908 available	766 1048576 available	114MB 14GB high watermark	394GB 48MB low watermark	4d 7h	basic disc 1 rss	This node All nodes
rabbit@rabbit_node2.isnet	57 100000 available	25 89908 available	900 1048576 available	142MB 14GB high watermark	401GB 48MB low watermark	4d 7h	basic disc 1 rss	This node All nodes
rabbit@rabbit_node3.isnet	43 100000 available	11 89908 available	698 1048576 available	114MB 14GB high watermark	400GB 48MB low watermark	4d 7h	basic disc 1 rss	This node All nodes

Churn statistics

Ports and contexts

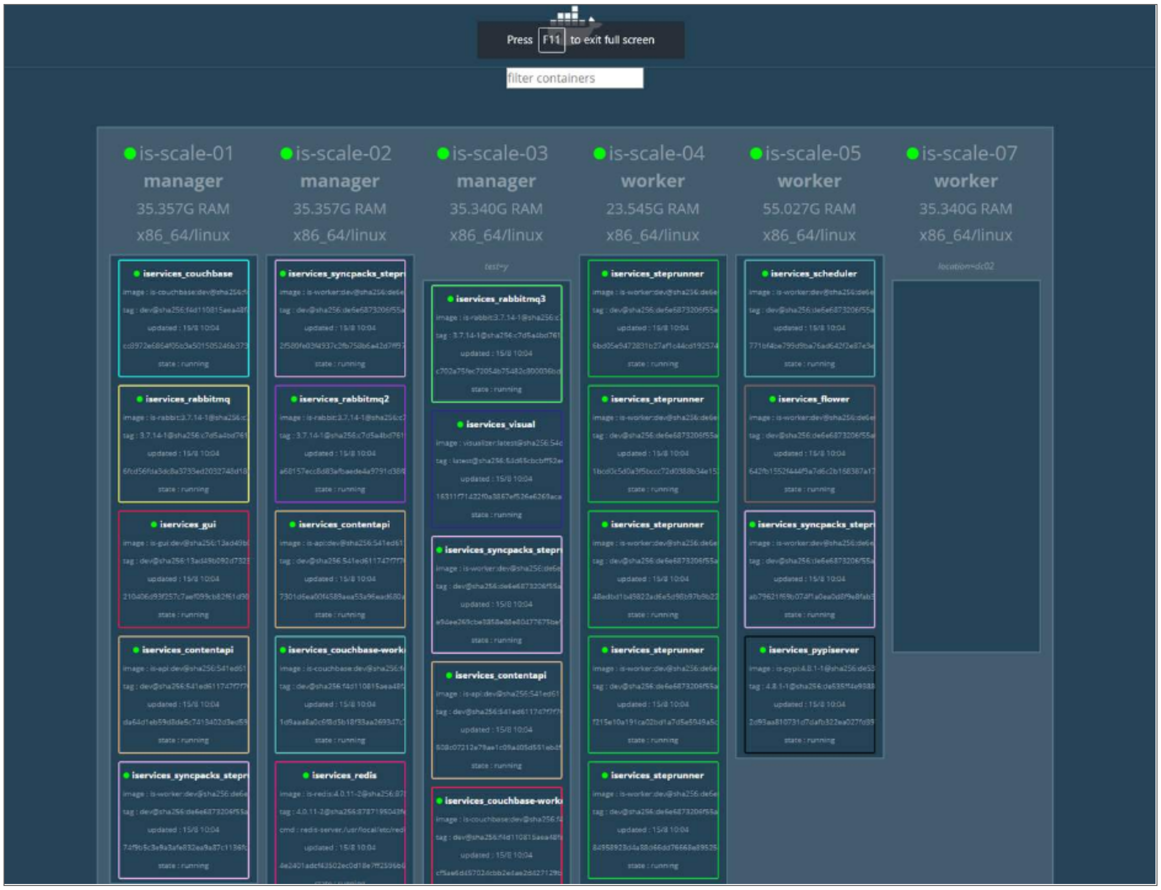
Export definitions

Import definitions

Docker Swarm Visualizer

Docker Swarm Visualizer is a visualizer for Docker Swarm using the Docker Remote API, Node.JS, and D3. Each node in the swarm shows all tasks running on it. When a service goes down, the service is be removed. When a node goes down it will not be removed; instead the circle icon in Visualizer turns red to indicate it went down.

The following image shows the Visualizer user interface, which you can access at port 8081:



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. For more information, see ***Using SL1 to Monitor the Integration Service***.

Chapter 3

Configuring the Integration Service for High Availability

Overview

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

Types of High Availability Deployments for the Integration Service

The following table contains a set of ratings that depict the level of resiliency enabled by various Integration Service deployment types. The higher the rating, the more resilient the Integration Service system, not just from a node failure perspective, but also from a throughput and load-balancing regard.

Deployment Type	Resiliency Rating	Typical Audience
Single-node deployment	F	Users who want the Integration Service running, but do not care about failover.
Three-node cluster	B+	Users who want the Integration Service running, and also want support for automatic failover for one-node failure.
3+ Node Cluster with separate workers (at least 4 nodes)	A-	Users who want automatic failover for one-node failure, and intend to have very CPU- or memory-intensive tasks executing on the workers constantly.
3+ Node cluster with separate workers, and drained manager nodes (at least 6 nodes)	A	Users who want automatic failover for one-node failure, intend to have very CPU- or memory-intensive tasks executing on the workers, and want to completely mitigate risks of resource contention between services.

NOTE: You can start with any deployment type, and at a later time scale up to any other deployment type as needed. For example, a you can start with a single-node deployment, then at a later date add three more nodes to enable a 3+ node cluster with separate workers.

The deployments listed in the table are just the standards for deployment. For very high-scale customers, a more advanced deployment might be necessary. For deployment requirements like this, please contact ScienceLogic Support.

The standard deployments are listed below in the following topics:

- *Standard Single-node Deployment (1 Node)*
- *Standard Three-node Cluster (3 Nodes)*
- *3+ Node Cluster with Separate Workers (4 or More Nodes)*
- *3+ Node Cluster with Separate Workers and Drained Manager Nodes (6 or More Nodes)*

NOTE: You can use a command-line utility called **iservicecontrol** that performs multiple administrator-level actions on either the node or the cluster. You can use this script to automate the configuration of a three-node cluster. For more information, see *Automating the Configuration of a Three-Node Cluster*.

Standard Single-node Deployment (1 Node)

Single-node deployment is the standard deployment that comes along with the ISO and RPM installation. This is the default deployment if you install the ISO and run the **pull_start_iservices.sh** script.

This deployment provides a single node running the Integration Service system. If this node fails, the system will not be operational.

Requirements

One node, 8 CPU, 24 GB memory minimum, preferably 34 GB to 56 GB memory, depending on workload sizes. For more information, see *System Requirements*.

Risks

A single node supports no data replication, no queue mirroring, and no failover capabilities.

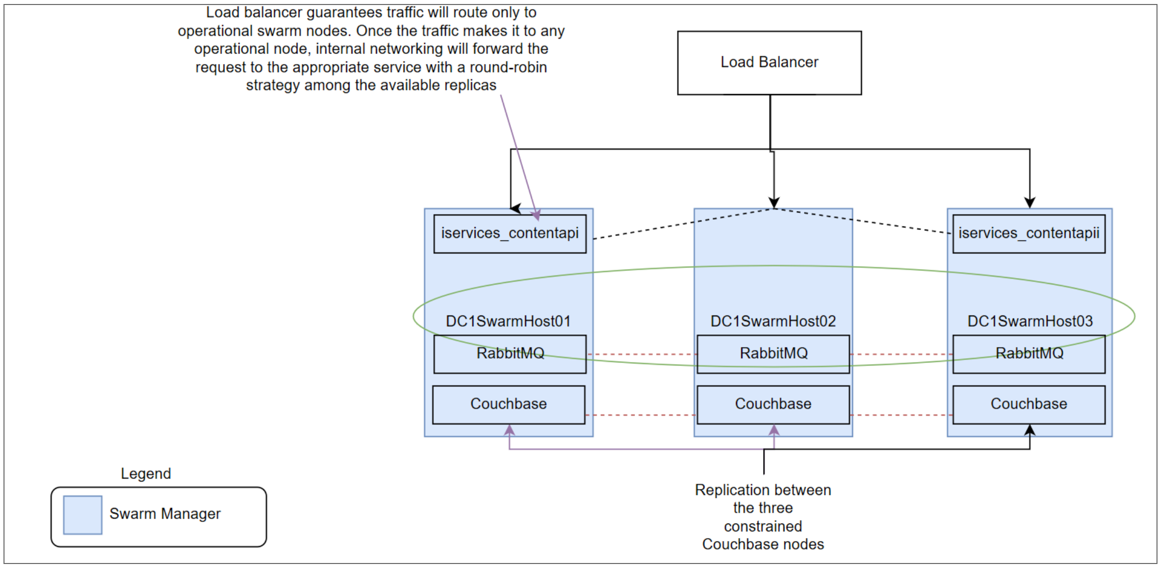
Configuration

This configuration is available as a default deployment with the docker-compose included in the Integration Service 2.0.0 or later ISO or RPM.

Standard Three-node Cluster (3 Nodes)

The following High Availability deployment is an example of a three-node cluster:

- Each node in the Swarm is a Swarm Manager.
- All Swarm nodes are located within the same data center.


Load balancer guarantees traffic will route only to operational swarm nodes. Once the traffic makes it to any operational node, internal networking will forward the request to the appropriate service with a round-robin strategy among the available replicas
Load Balancer
iservices_contentapi
iservices_contentapii
DC1SwarmHost01
DC1SwarmHost02
DC1SwarmHost03
RabbitMQ
RabbitMQ
RabbitMQ
Couchbase
Couchbase
Couchbase
Legend
Swarm Manager
Replication between the three constrained Couchbase nodes

The three-node cluster is the most basic option providing full High Availability and data replication support among three nodes. In this deployment, each of the three nodes are running the same services in a clustered environment, which provides failover and data loss prevention capabilities. This deployment option will satisfy most High Availability needs, but it does not mitigate risks with the potential for worker operations to affect and degrade the database and queue services, because all services are running on the same nodes.

This deployment provides:

- **Automatic failover for one out of three node failure**: If one node in the cluster fails, automatic failover occurs, and the Integration Service system will continue to be operational running on two out of three of the nodes.
- **Full data replication between all three nodes**. All nodes have a copy of the same data replicated across all three nodes. If one or two nodes fail, you will not experience data loss in the database or in the queues.
- **Full queue mirroring between all three nodes**. All nodes have a mirror of the queues defined in the Integration Service environment. If one or two nodes fail, the system still retains messages in queues using the autoheal policy by default. For more information about autoheal behavior in RabbitMQ, see ***The RabbitMQ Split-brain Handling Strategy***.

Requirements

Three nodes, 8 CPU, 24 GB memory minimum, preferably 34 GB to 56 GB memory, depending on workload sizes. For more information, see ***System Requirements***.

Risks

When only three nodes are allocated used for High Availability, the following risks are present:

- **Over-utilization of nodes causing clustering issues**. In a three node cluster, worker containers, and Docker Swarm Managers are running on the same node as the database and queue services. As a result, if the node is not provisioned correctly, there could be some resource contention. If a node reaches 100% CPU, Docker Swarm cluster operations might fail, causing a node to completely restart, and causing a failover or other unexpected behavior.
- **Over-utilization of workers nodes causing database or queue issues**. Since all services are sharing the same nodes in this configuration, if worker operations become extremely CPU- or memory-intensive, the system might try to use resources needed from the database or queue. If this happens, you might encounter failures when querying the database or using the queues.

Mitigating Risks

The above risks can be mitigated by ensuring that the node is deployed with adequate CPU and memory for the workloads that you plan to run on the node. Memory limits are placed on containers by default. If needed, you could also add CPU limits to worker containers to further prevent resource contention.

Configuration

The Integration Service uses a **docker-compose-override.yml** file to persistently store user-specific configurations for containers, such as proxy settings, replica settings, additional node settings, and deploy constraints. The user-specific changes are kept in this file so that they can be re-applied when the **/opt/iservices/docker-compose.yml** file is completely replaced on an RPM upgrade, ensuring that no user-specific configurations are lost.

Below is an example **docker-compose-override.yml** file for Integration Service version 2.0.0:

```
services:
  contentapi:
    environment:
      db_host: "couchbase.isnet,couchbase-worker.isnet,couchbase-worker2.isnet"
  couchbase:
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname1>"
    environment:
      db_host: couchbase.isnet
    hostname: couchbase.isnet
    networks:
      isnet:
        aliases:
          - couchbase
          - couchbase.isnet
  couchbase-worker:
    container_name: couchbase-worker
    depends_on:
      - couchbase
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname2>"
      replicas: 0
    environment:
      AUTO_REBALANCE: "true"
      TYPE: WORKER
      db_host: couchbase
    hostname: couchbase-worker.isnet
    image: "sciencelogic/is-couchbase:1.7.0"
    networks:
      isnet:
        aliases:
          - couchbase-worker
          - couchbase-worker.isnet
    ports:
      - "8100:8091"
    secrets:
      - is_pass
      - encryption_key
    volumes:
      - "/var/data/couchbase:/opt/couchbase/var"
  couchbase-worker2:
    container_name: couchbase-worker2
    depends_on:
      - couchbase
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname3>"
      replicas: 0
    environment:
      AUTO_REBALANCE: "true"
      TYPE: WORKER
      db_host: couchbase
    hostname: couchbase-worker2.isnet
    image: "sciencelogic/is-couchbase:1.7.0"
    networks:
      isnet:
        aliases:
          - couchbase-worker2
          - couchbase-worker2.isnet
    ports:
      - "8101:8091"
    secrets:
      - is_pass
      - encryption_key
    volumes:
      - "/var/data/couchbase:/opt/couchbase/var"
  dexserver:
    deploy:
      replicas: 2
    environment:
      db_host: "couchbase.isnet,couchbase-worker.isnet,couchbase-worker2.isnet"
  pypiserver:
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname1>"
  rabbitmq:
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname1>"
    hostname: rabbit_node1.isnet
    image: "sciencelogic/is-rabbit:3.7.14-3"
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node1.isnet
    volumes:
      - "rabbitdb:/var/lib/rabbitmq"
  rabbitmq2:
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname2>"
    hostname: rabbit_node2.isnet
    image: "sciencelogic/is-rabbit:3.7.14-3"
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node2.isnet
    volumes:
      - "rabbitdb2:/var/lib/rabbitmq"
  rabbitmq3:
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname3>"
    hostname: rabbit_node3.isnet
    image: "sciencelogic/is-rabbit:3.7.14-3"
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node3.isnet
    volumes:
      - "rabbitdb3:/var/lib/rabbitmq"
  scheduler:
    environment:
      db_host: "couchbase.isnet,couchbase-worker2.isnet,couchbase-worker.isnet"
  steprunner:
    environment:
      db_host: "couchbase.isnet,couchbase-worker2.isnet,couchbase-worker.isnet"
version: "3.4"
volumes:
  rabbitdb2:
  rabbitdb3:
```

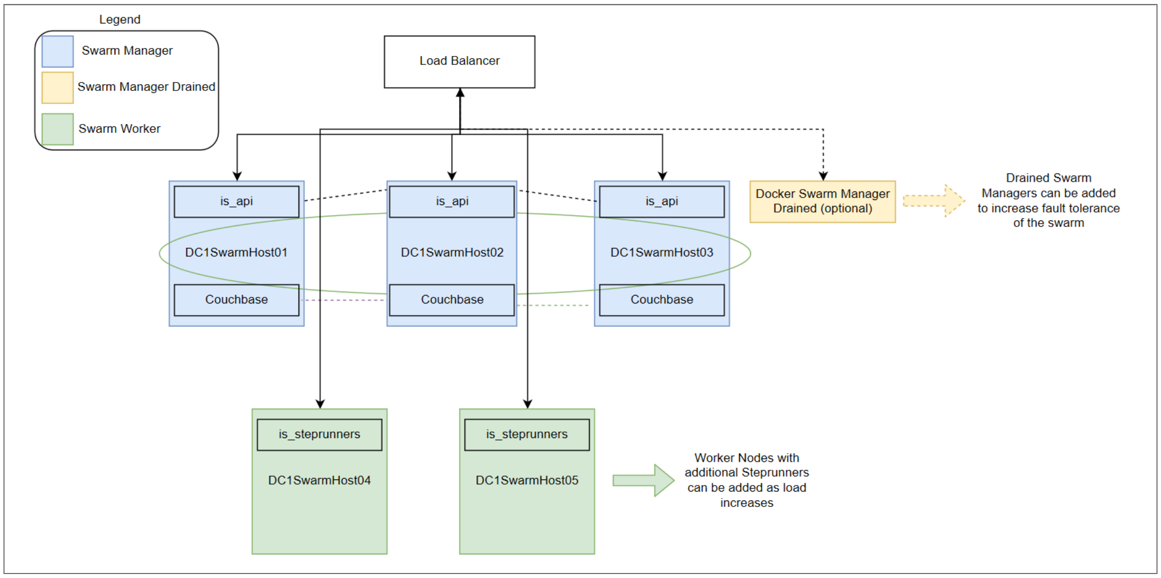
3+ Node Cluster with Separate Workers (4 or More Nodes)

The three-node cluster with separate workers is a slight variation of the standard three-node cluster. With this deployment strategy, all worker operation load is run by a separate independent node. This is preferable over the standard three-node deployment, because it completely prevents worker operations from stealing resources from the databases or queues.

Since steprunner workload is entirely on dedicated servers, you have greater ability to scale up to more workers, or even add additional nodes of workers to the system, without affecting critical database or queue operations.

This deployment provides a complete separation of worker processing from the database and queue processing, which is very helpful for users which have very CPU-intensive tasks that execute frequently.

The following High Availability deployment adds Docker Swarm worker nodes where steprunners can be constrained. This lets you continue to scale out new worker nodes as the load increases. This also lets you distribute steprunners based on workloads. Core services include ContentAPI, RabbitMQ, and Couchbase.



You can add drained Docker Swarm Manager nodes to increase fault tolerance of the Swarm, and to ensure that the orchestration of the Swarm is not impeded by large workloads on the core nodes.

The maximum Couchbase cluster with fully replicated nodes is four. Anything greater than four will not have a full replica set and will auto-shard data across additional nodes. There is no way as of this version of Couchbase to set the placement of the replicas. Redis replication and clustering is not currently supported in this version of the Integration Service

Requirements

Three nodes, 8 CPU, 24 GB memory minimum, preferably 34 GB to 56 GB memory, depending on workload sizes. For more information, see *System Requirements*.

One or more worker node with your choice of sizing.

Worker Node Sizing

Worker nodes can be sized to any CPU or memory constraints, though the greater the memory and CPU, more workers the node can run. The minimum size of a worker node is 2 CPU, 4 GB memory.

Risks

Core Node over-utilization could cause Swarm clustering problems. Because the Swarms are the same nodes as the core managers, there is a possibility for heavily loaded databases and queues to contend with the Swarm hosts for resources. In this case the Swarm may restart itself and the services running on that node. This is not as likely to occur with workers running on their own dedicated nodes.

Mitigating Risks

The above risks can easily be mitigated by ensuring the node is deployed with adequate CPU and memory for the workloads it's expected to run. Additionally, you can apply CPU and memory limits to the database or queue containers so that there will always be enough resources allocated to the host to prevent this scenario. For more information, see *Configuring Additional Elements of the Integration Service*.

Configuration

Using this configuration consists of:

- Joining the standard three-node Swarm cluster with one or more nodes as a Swarm worker.
- Labeling each additional "worker" node with a Swarm label "worker". For more information, see *Creating a Node Label*.
- In addition to the standard three-node deployment, steprunners should be updated to run on a dedicated node in the **docker-compose-override** file:

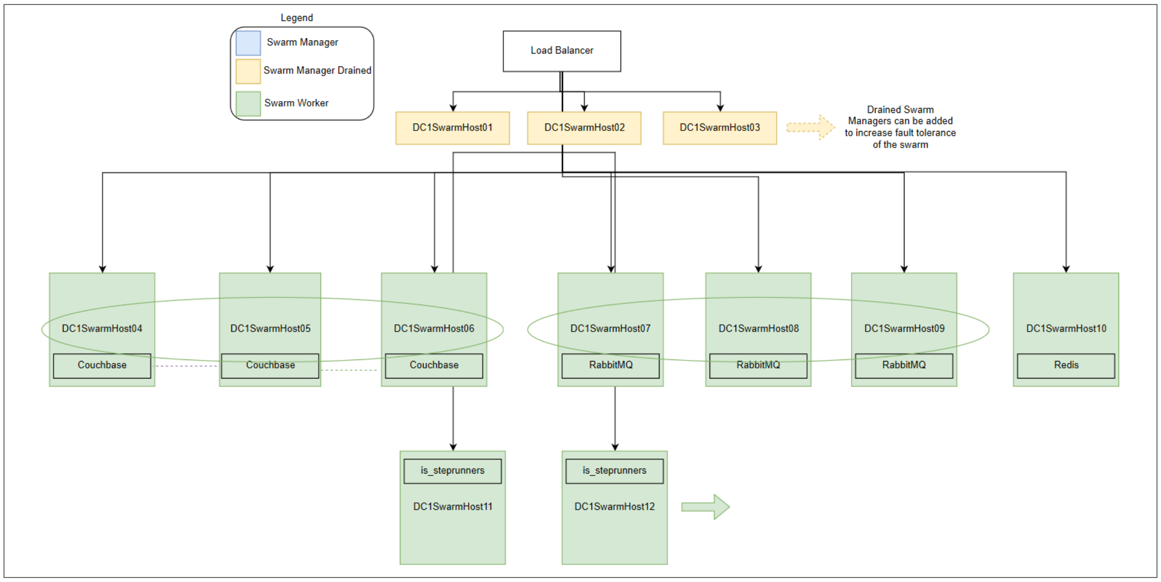
```
steprunner3:
  deploy:
    placement:
      constraints:
      - node.labels.types == worker
```

3+ Node Cluster with Separate Workers and Drained Manager Nodes (6 or More Nodes)

This deployment option is the most robust of the one-node auto-failover deployments, and completely mitigates known risks for resource contention in clusters.

This configuration provides everything that the 3+ node cluster with dedicated workers provides, with the addition of drained Swarm Managers. The drained Swarm Managers mitigate the risk of database or queue processing causing contention of resources for the Swarm clustering operations at the host level.

This deployment should only be used for large deployments of the Integration Service. This deployment separates out all the core services onto their own dedicated worker node and lets you distribute steprunners based on workloads:



You can add drained Docker Swarm Manager nodes to increase fault tolerance of the Swarm, and to ensure that the orchestration of the Swarm is not impeded by large workloads on the core nodes.

The maximum Couchbase cluster with fully replicated nodes is four. Anything greater than four will not have a full replica set and will auto-shard data across additional nodes. There is no way as of this version of Couchbase to set the placement of the replicas. Redis replication and clustering is not currently supported in this version of the Integration Service

Requirements

Three nodes, 8 CPU, 24 GB memory minimum, preferably 34 GB to 56 GB memory, depending on workload sizes. For more information, see *System Requirements*.

Also, three nodes, 2 CPU, 4 GB memory for the Swarm Manager.

Risks

None.

Configuration

Use the same **docker-compose-override.yml** file found in *Standard Three-node Cluster (3 Nodes)*.

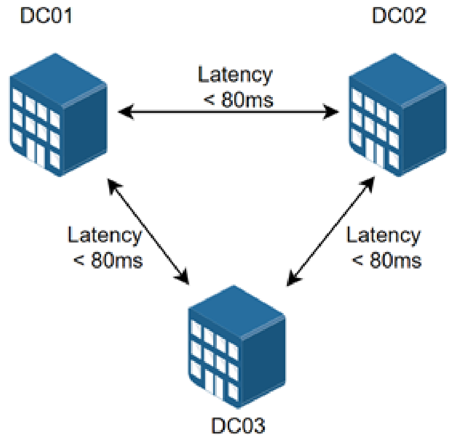
Next, add the additional three nodes to the cluster as managers, and drain them of all services (see *Using Drained Managers to Maintain Swarm Health*). Promote the drained nodes to Swarm Managers, and make all other nodes workers.

Additional Deployment Options

The following diagrams show additional High Availability deployment architectures that are supported for the Integration Service.

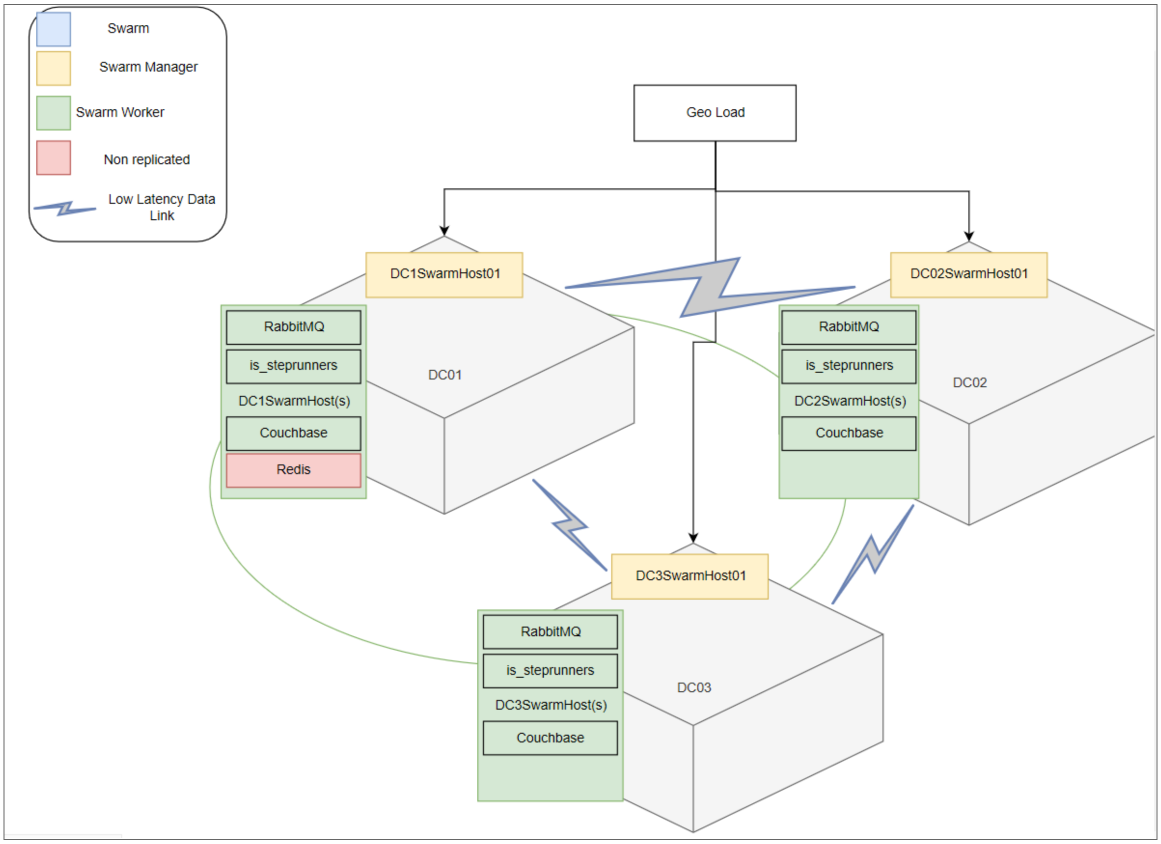
Cross-Data Center Swarm Configuration

Docker Swarm requires three data centers to maintain quorum of the swarm in the event of a full data center outage. Each data center must have a low-latency connection between the data centers.



The cross-data center configuration has the following limitation: the Redis service cannot be deployed in High Availability. As a result, all task results saved by any steprunner will have to be saved within that data center. Upon a failure of that data center, a new Redis service will be created, but an application in the middle of its run would have to retry.

The following High Availability deployment shows a cross-data center swarm configuration:



Additional Notes

Tagging and constraints in the Docker compose file should be used to ensure proper placement. Example compose files are not available at this time.

Configuration management solutions such as Ansible should be used to update and manage large swarm environments.

For easy upgrade of the Integration Service, use Docker Hub to pull the latest images or use an internal Docker registry.

Requirements Overview

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.

IMPORTANT: To support automatic failover of the Couchbase database without manual intervention, you must set up at least **three nodes** for automatic failover of a single node, **five nodes** for automatic failover of two nodes, and so on.

NOTE: For a clustered Integration Service environment, you must install the Integration Service RPM on every server that you plan to cluster into the Integration Service. You can load the Docker images for the services onto each server locally by running **/opt/iservices/scripts/pull_start_iservices.sh**. Installing the RPM onto each server ensures that the Integration Service containers and necessary data are available on all servers in the cluster. For more information, see ***Installing the Integration Service via RPM***.

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 ports 443:HTTPS, 3141:Devpi and 5556:Dex.

An example of why a load balancer is needed in front of the virtual IP is the ServiceNow ticketing workflow. If you're only directing the request to a single node and that node goes down, your ticketing will stop even if the other Integration Service nodes are still up and functional. The load balancer will account for the downed node and automatically route to the other nodes in the cluster.

For more information, see the Docker High Availability Documentation.

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 from the Integration Service 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.

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.

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 docker-compose Definition

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

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

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

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

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

Potential Risks and Mitigations

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

Requirements/Setup

For a Docker Swarm configuration with multiple independent nodes:

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

Example docker-compose Definition of Two Clustered Rabbit Services

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

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

Checking the Status of a RabbitMQ Cluster

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

To check the status of your clustered RabbitMQ environment:

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

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

You can run the following commands for more information:

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

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

Preparing the Integration Service System for High Availability

You need to prepare your Integration Service in the following ways before configuring the High Availability solution:

1. Make sure 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
```

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
```

TIP: To view a list of all ports, run the following command: `firewall-cmd --list-all`

3. Make sure that the **/etc/iservices/is_pass** and **/etc/iservices/encryption_key** are identical on all clustered nodes.
4. Make sure that NTP is properly configured on all nodes:
   - Edit the **/etc/chrony.conf file** to add NTP servers. If you want to use the pool.ntp.org NTP servers, remove the **.ol.** from the domain names.
   - Enable chrony by running the following commands:

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

Configuring Clustering and High Availability

This section describes how to configure clustering and High Availability with Docker Swarm and the Couchbase database, using 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.

Automating the Configuration of a Three-Node Cluster

Integration Service 2.0.0 includes a command-line utility called ***iservicecontrol*** that performs multiple administrator-level actions on either the node or the cluster. You can use this script to automate the configuration of a three-node cluster.

To automate the configuration of a three-node cluster, run the following command:

```
iservicecontrol autocluster --host <is_host1>  <username>:<password> --host <is_host2>
 <username>:<password> --host <is_host3>  <username>:<password>
```

For example:

```
iservicecontrol autocluster --host 192.11.1.1 isadmin:passw0rd --host 192.11.1.2
isadmin:passw0rd --host 192.11.1.3 isadmin:passw0rd
```

Running this command will configure your Integration Service three-node cluster without any additional manual steps required.

For more information on the **iservicecontrol** utility, run the following command:

```
# iservicecontrol --help
```

Configuring Docker Swarm

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

> **NOTE:** Two-Node High Availability is not possible because Docker Swarm requires an odd number of nodes (3+) for quorum and consensus.

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 ***Preparing the Integration Service System for High Availability***.
3. SSH to the Docker Swarm manager (leader) 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 in step 4, below:

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

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

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

   where `<join token>` is the value from step 3. For example:

   ```
   docker swarm join --token SWMTKN-1-
   5e8skxby61cthkfkv6gzhhil89v0og2m7lx014tvvv42n7m0rz-an0fdam5zj0v7d471co57d09h
   10.7.3.21:2377
   ```

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 $(ls -1 /opt/iservices/images/); do docker load -i
   /opt/iservices/images/$i; done
   ```

Configuring the Couchbase Database

To add a Couchbase worker node:

1. In the **docker-compose-override.yml** file, 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 db_host variables in docker-compose should be in the following format:

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

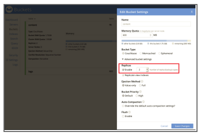
5. If you are using the override file, run the **/opt/iservices/compose_override.sh** script to validate and update the **docker-compose.yml** file with your changes.
6. Deploy the stack with only the Couchbase node by editing the replicas on couchbase-worker to 1 and running the following command:

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

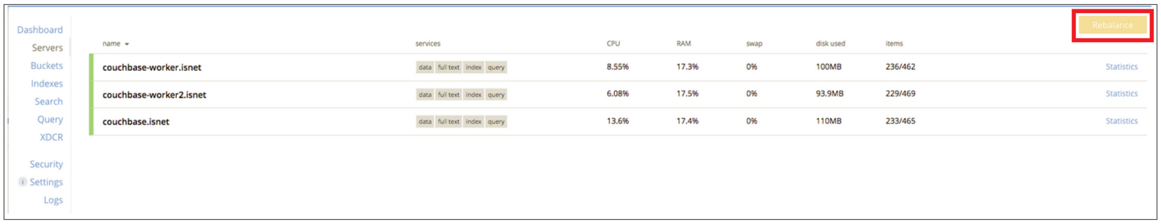
7. After the two-node Couchbase cluster has been successfully deployed and the secondary indexes are successfully added, edit the replicas on couchbase-worker2 to 1 and run the following command:

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

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



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



name	services	CPU	RAM	swap	disk used	items	
couchbase-worker.isnet	data full text index query	8.55%	17.3%	0%	100MB	236/462	Statistics
couchbase-worker2.isnet	data full text index query	6.08%	17.5%	0%	93.9MB	229/469	Statistics
couchbase.isnet	data full text index query	13.6%	17.4%	0%	110MB	233/465	Statistics

Code Example: docker-compose-override.yml

The Integration Service uses a **docker-compose-override.yml** file to persistently store user-specific configurations for containers, such as proxy settings, replica settings, additional node settings, and deploy constraints. The user-specific changes are kept in this file so that they can be re-applied when the **/opt/iservices/docker-compose.yml** file is completely replaced on an RPM upgrade, ensuring that no user-specific configurations are lost.

If you are running the Integration Service in a cluster, these files should always be the same between all manager nodes. With this in place, if any manager node dies, you can re-deploy with the same settings from any other manager node.

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"
    ports:
      - "8100:8091"
    deploy:
      replicas: 0
      placement:
        constraints:
          - node.hostname == <swarm node hostname>
    networks:
      isnet:
        aliases:
          - couchbase-worker
          - couchbase-worker.isnet
    hostname: couchbase-worker.isnet
    secrets:
      - is_pass
      - encryption_key
    environment:
       TYPE: 'WORKER'
       AUTO_REBALANCE: 'true'
       db_host: 'couchbase'
    depends_on:
      - couchbase

  couchbase-worker2:
    image: sciencelogic/is-couchbase:1.7.0
    container_name: couchbase-worker2
    ports:
```

```
    - "8101:8091"
  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:3.7.7-1
   hostname: rabbit_node2.isnet
   volumes:
     - "rabbitdb2:/var/lib/rabbitmq"

   networks:
     isnet:
       aliases:
         - rabbit
         - rabbit_node2.isnet
   deploy:
     placement:
       constraints:
         - node.hostname == <swarm node hostname>
rabbitmq3:
   image: sciencelogic/is-rabbit:3.7.7-1
   hostname: rabbit_node3.isnet
   volumes:
       - "rabbitdb3:/var/lib/rabbitmq"

     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'

  pypiserver:
     image: sciencelogic/is-pypi:4.8.0-1
     hostname: devpi
     container_name: devpi
     volumes:
        - "devpi:/data"
     networks:
        isnet:
           aliases:
              - pypiserver
     secrets:
        - is_pass
  dexserver:
    image: scr.sl1.io/is-dex:2.18.0-1
    ports:
      - "5556:5556"
      - "5558:5558"
    command: ["serve", "/dexConfiguration.yaml"]
    networks:
      isnet:
        aliases:
          - dexserver
    configs:
      - source: dex_config
        target: /dexConfiguration.yaml
volumes:
  rabbitdb2:
  rabbitdb3:
  devpi:
configs:
  dex_config:
    file: /etc/iservices/dexConfiguration.yaml
```

Scaling iservices-contentapi

To scale out the iservices-contentapi 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

Optimization Settings to Improve Performance of Large-Scale Clusters

In large-scale clusters, one of the root causes of abnormal memory and CPU usage is from inter-worker communication overhead, or overly "chatty" workers, and their event queues. You can completely disable inter-worker eventing to significantly reduce overhead on the queuing system and prevent the symptoms associated with abnormal memory usage.

Also, to improve the performance of large-scale clusters by default, the following optimization settings were added to the **docker-compose.yml** file for all workers in version 2.0.1 of the Integration Service platform:

```
steprunner-<worker-x>:
  environment:
    additional_worker_args: "--max-tasks-per-child 1 --without-gossip --without-
mingle"
```

In addition to the default optimization settings above, you can further reduce system overhead by setting the `--without-heartbeat` environment variable in `additional_worker_args`. Please note that this setting will reduce the memory and CPU utilization of the system, but it will come at the cost of preventing the Flower service from getting an accurate depiction of current worker states.

If you want to disable these new configuration settings, set the environment variable "disable_default_optimizations" to "True" for all workers.

> **NOTE:** Workers will continue to generate events for consumption from monitoring tools like Flower even with the new default configuration settings. In some extremely large clusters, you might want to completely disable eventing of workers completely, especially if Flower is not in use. To completely disable worker eventing, set the environment variable "disable_events" to "True".

For more information, see https://docs.celeryproject.org/en/latest/reference/celery.bin.worker.html#cmdoption-celery-worker-without-gossip

Additional suggestions for improving performance in large-scale clusters:

- Assess the impact of using Flower before keeping it enabled for a long period of time. Running Flower can cause increased overhead on the RabbitMQ nodes, but the overhead is not significant initially. However, the overhead generated by Flower will continue to increase as more workers are added to the stack, and those workers send events to Flower.
- ScienceLogic recommends that you monitor memory and queue utilization before and after running Flower with your current environment size to determine whether the extra overhead provided is worth the task information it provides.
- If a system event causes workers to restart, it is possible that all workers constantly restarting at the same time, every 0 seconds will generate increased load on the system, making it difficult for other services to start up. To prevent this, it is recommended to add a **restart_delay** to workers to prevent a "rush" of hundreds of workers trying to re-connect over the network all at once. For example:

```
steprunner-<worker-x>:
  deploy
    restart_policy:
      delay: 30s
```

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 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 = / {
  return 301 https://$host:8092/ui/index.html;
}

 location / {
  resolver 127.0.0.11 valid=5s;
  set $upstream couchbase-worker.isnet;
  proxy_pass http://$upstream:8092$request_uri;
  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:

```
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option                  redispatch
    retries                 3
    timeout http-request    1m
    timeout queue           1m
```

```
    timeout connect         1m
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 6000

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

frontend dex_front
  bind *:5556
  option tcplog
  mode tcp
  tcp-request inspect-delay 5s
  default_backend dex_back
frontend  devpi_front
  bind *:3141
  option tcplog
  mode tcp
  tcp-request inspect-delay 5s
  default_backend  devpi_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

backend dex_back
  mode tcp
  balance roundrobin
  server master1 <docker swarm node 1 ip>:5556 check
  server master2 <docker swarm node 2 ip>:5556 check
  server master3 <docker swarm node 3 ip>:5556 check


backend devpi_back
  mode tcp
  balance roundrobin
  server master1 <docker swarm node 1 ip>:3141 check
  server master2 <docker swarm node 2 ip>:3141 check
  server master3 <docker swarm node 3 ip>:3141 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 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.

Couchbase rebalance fails with "Rebalance exited" error

In this situation, you received the following error:

```
Rebalance exited with reason {service_rebalance_failed,index,
{linked_process_died,<12807.821.0>,
{no_connection,"index-service_api"}
}}
```

If the Couchbase rebalance fails on the initial rejoin of a failed node into a cluster, you should check the index states and wait until the indexes are no longer in a warmup state. After the indexes are created on that node, the rebalance should succeed.

When setting up a three-node High Availability Couchbase cluster, the second node does not appear

In this situation, if you have cloned any of the nodes, the nodes might think that there is a split-brain condition.

To address this issue, delete the Couchbase data on the newly added nodes by running the following command on each node:

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

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 - 2v4umws4pxag6kbxaelwfl3vf) 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 *Preparing the Integration Service System 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]
```

Chapter 4

Managing Users in the Integration Service

Overview

This chapter describes how to configure authentication for the Integration Service to allow access to multiple users with a variety of roles and permissions.

This chapter also describes how to use the **Admin Panel** page (👥) to manage user group access to the Integration Service. Only users with the *Administrator* role for the Integration Service system can edit this page.

This chapter covers the following topics:

Configuring Authentication with the Integration Service

Integration Service version 2.0.0 or later supports the following authentication methods:

- ***Local Authentication***. The same local Administrator user (*isadmin*) is supported by default with 2.0.0 or later installations. If you are migrating from a previous version of the Integration Service to version 2.0.0 or later, you can log in and authenticate with the same user and password.
- ***Basic Authentication***. Integration Service 2.0.0 or later continues to support Basic Authentication as well. Because the Integration ServicePowerPacks, diagnostic scripts, and the iscli tool continue to use Basic Authentication, ScienceLogic does not recommend disabling Basic Authentication with Integration Service version 2.0.0 or later.
- ***OAuth***. Lets Integration Service administrators use their own authentication providers to enforce user authentication and lockout policies. Authentication using a third-party provider, such as LDAP or Active Directory, requires additional configuration. For optimal security, ScienceLogic recommends that you disable the local Administrator user (*isadmin*) and exclusively use your own authentication provider.

Regardless of authentication strategy, authorization and role access is configured separately, based on user or user group. For more information, see ***Creating a User Group in the Integration Service***.

The following topics describe how to configure each authentication strategy for the Integration Service:

- ***User Interface Login Administrator User (default)***
- ***Basic Authentication Using a REST Administrator User (default)***
- ***User Interface Login Using a Third-party Authentication Provider***
- ***OAuth Client Authentication Using a Third-party Provider***

User Interface Login Administrator User (Default)

The local Administrator user is the default login user for the Integration Service. The username is "isadmin" by default, and the password gets set during the ISO installation process. The Integration Service administrator can change the default password or the Administrator user.

This authentication strategy allows authentication of the local Administrator user through the Integration Service user interface. The "isadmin" user is the local Administrator by default, but you can change the username of the local Administrator to something other than "isadmin" if needed.

> WARNING: If you disable this authentication strategy, you must first configure an alternative provider to appropriately authenticate to the Integration Service system and also configure a user or user group policy that has the *Administrator* role. If you disable this user without a second authentication provider configured, the Integration Service will not be able to authenticate any users.

To disable this user, set the following environment variable in the **/etc/iservices/isconfig.yml** file:

```
BASIC_AUTH: False
```

To change the local Administrator username, set the following environment variable in the **/etc/iservices/isconfig.yml** file:

```
BASIC_AUTH_USER: "username"
```

NOTE: Before changing the local Administrator username, make sure that the user has *Administrator* permissions in the Integration Service system. For more information, see ***User Groups, Roles, and Permissions***.

ScienceLogic recommends that you use the same system-wide password for the local Administrator user, which is located in **/etc/iservices/is_pass**. To change this password, run the **/opt/iservices/scripts/ispasswd** script.

Alternatively, you can set a different password for the local Administrator user by setting the following environment variable in **/etc/iservices/isconfig.yml**:

```
BASIC_AUTH_PASSWORD: "BASIC_AUTH_PASSWORD"
```

Basic Authentication Using a REST Administrator User (Default)

Basic Authentication through a REST administrator user enables the local Administrator user to access the Integration Service API through REST using Basic Authentication. This authentication strategy enables users to make queries through the API using `<isadmin:password>` basic authentication.

This Basic Authentication is enabled by default in Integration Service 2.0.0 or later. The REST administrator uses the same environment variables and configuration as the ***user interface login Administrator user***.

Basic Authentication is limited to only the local Administrator user. If your Integration Service system is configured to use a different authentication provider, you must authenticate using OAuth 2.0 and a bearer token. For more information, see ***OAuth Client Authentication Using an Internal Provider***.

WARNING: This Basic Authentication user is enabled by default to support backward compatibility with any scripts or queries running against the versions of the Integration Service before 2.0.0. If you disable this authentication strategy, PowerPacks and end-user scripts will not be able to query or access the API unless they are updated to use OAuth 2.0 with bearer a token.

User Interface Login Using a Third-party Authentication Provider

This authentication strategy lets you set up user authentication through a third-party authentication provider, such as LDAP or Active Directory (AD). You configure additional providers and their connectors in the **/etc/iservices/isconfig.yml** file, following the Dex connector configuration described below. This authentication strategy is automatically disabled when no connectors are present in the configuration.

After you configure the providers, users can select one of the defined providers or the local Administrator authentication, and authenticate using that provider.

When using this authentication strategy, the Integration Service system automatically retrieves the LDAP or AD groups to which the user belongs. You can use these groups can be used to apply specific role permissions. For more information, see ***Role-based Access Control (RBAC) Configuration***.

NOTE: By default, no third-party authentication providers are configured in the Integration Service.

Credentials for user authentication exist only with the third-party authentication provider, and the credentials are not imported into the Integration Service. The only information that the Integration Service retains for these users are the roles and permissions attached to the usernames.

To configure a third-party authentication provider:

1. Go to https://github.com/dexidp/dex#connectors and locate the connector type, such as LDAP or SAML 2.0, that you would like to use for authentication.
2. Click the link for the connector type to view example configuration options for the connector.
3. Update the **/etc/iservices/isconfig.yml** file and add a **DEX_CONNECTORS** section with the configuration for the connector you want to use. The **DEX_CONNECTORS** section in the **isconfig.yml** file is identical to the **CONNECTORS** section described in the Dex documentation.
4. Re-deploy the Integration Service stack and check for errors in docker service logs **iservices_dexserver**. If the dexserver starts successfully and the Integration Service is running, then the Integration Service has accepted the configuration.
5. Next, log in to the Integration Service system with a user provided by the newly configured authentication.
6. Look for errors with searching users or user groups in the user interface or the Docker service logs.

For reference, the following is an example **/etc/iservices/isconfig.yml** file with an Active Directory authentication provider configured to look up users and their groups:

```
HOST_ADDRESS: 10.2.11.222
CLIENT_ID: isproxy
CLIENT_SECRET: knivq7uDPVORdrSlWJ0I4YiiwuQgGDsf9rMWquoInYs
SESSION_SECRET: BDLO2xPrBs_s-YkqY-j4lN6VPeBzyrVsYt_P10oWbn0
DB_HOST: couchbase.isnet,localhost

DEX_CONNECTORS:
- type: ldap
  # Required field for connector id.
  name: ldap
  id: ldap
  # Required field for connector name.
  config:
    host: rstcsdc01.sciencelogic.local:636
       # Host and optional port of the LDAP server in the form "host:port".
       # If the port is not supplied, it will be guessed based on "insecureNoSSL",
       # and "startTLS" flags. 389 for insecure or StartTLS connections, 636
       # otherwise.

    insecureNoSSL: false
       # The insecureNoSSL field is required if the LDAP host is not using TLS (port 389).
       # Because this option inherently leaks passwords to anyone on the same network
       # as dex, THIS OPTION MAY BE REMOVED WITHOUT WARNING IN A FUTURE RELEASE.

    insecureSkipVerify: true
       # If a custom certificate isn't provide, this option can be used to turn on
       # TLS certificate checks. As noted, it is insecure and shouldn't be used outside
       # of explorative phases.

    bindDN: auser
    bindPW: password
       # The DN and password for an application service account. The connector uses
       # these credentials to search for users and groups. Not required if the LDAP
       # server provides access for anonymous auth.
       # Please note that if the bind password contains a `$`, it has to be saved in an
       # environment variable which should be given as the value to the bindPW field.

    usernamePrompt: silo credentials
       # The attribute to display in the provided password prompt. If unset, will
       # display "Username"

    userSearch:
       # User search maps a username and password entered by a user to a LDAP entry.
      baseDN: OU=Domain User Accounts,DC=ScienceLogic,DC=local
         # BaseDN to start the search from.
      filter: "(objectClass=user)"
         # Optional filter to apply when searching the directory.

      username: userPrincipalName
         # username attribute used for comparing user entries.
         # The following three fields are direct mappings of attributes on the user entry.
      idAttr: DN
         # String representation of the user.
      emailAttr: userPrincipalName
         # Required. Attribute to map to email.
      nameAttr: cn
         # The nameAttr field maps to the display name of users. No default value.

    groupSearch:
       # Group search queries for groups given a user entry. Group searches must match a
user
       # attribute to a group attribute.
      baseDN: OU=Domain Groups,DC=ScienceLogic,DC=local
         # BaseDN to start the search from.
      filter: "(objectClass=group)"
         # Optional filter to apply when searching the directory.

         # The following list contains field pairs that are used to match a user to a group.
         # It adds an additional requirement to the filter that an attribute in the group
         # must match the user's attribute value.
      userAttr: DN
         # The userAttr field is the attribute used from the user search query.
         # to match to an element of the group search.
      groupAttr: member
         # The groupAttr field is the attribute of the group results that should match the
userAttr value.
         nameAttr: cn
         #  This value must exactly match the group defined in the IS Group configuration.
```

OAuth Client Authentication Using a Third-party Provider

OAuth Client Authentication provides user authentication with a configured third-party provider, such as LDAP or AD. This allows users to use clients authenticating with OAuth 2.0 bearer tokens to authenticate against their configured provider.

In Integration Service 2.0.0 or later, the OAuth Client Authentication strategy is automatically enabled when a authentication provider is configured. You can configure this strategy using the same parameters as the ***User Interface Login Using a Third-party Authentication Provider***.

The Integration Service system provides discovery endpoints for all OAuth 2.0 required endpoints. The discovery address is: **https://IS-IP:5556/dex/.well-known/openid-configuration**.

Using these discovery endpoints, you can use an OAuth 2.0 client to generate a secure token using a third-party authentication provider. You can use the generated secure token as a bearer authentication token (specified in request headers) to authenticate and make requests.

The **client_secret** and **session_secret** are unique, randomly generated strings generated for each IS deployment. To obtain an OAuth 2.0 token, you will need these values, which you can find in the **/etc/iservices/isconfig.yml** file.

Basic Authentication Lockout Removal

Integration Service 2.0.0 supports OAuth 2.0 with OpenID Connect, which lets Integration Service administrators use third-party authentication providers, such as LDAP or Active Directory, to enforce user authentication and lockout policies.

Integration Service 2.0.0 continues to support Basic Authentication as well, but the Integration Service no longer enforces automatically locking out the *isadmin* user if that user has too many failed login attempts.

If you are concerned about removing the lockout functionality for the *isadmin* user, you can perform one of the following actions:

1. Change the default username from *isadmin* to a different name to prevent brute force type attacks on the *isadmin* user.
2. Disable Basic Authentication and use third-party authentication providers, such as your company's LDAP server, to enforce user authentication and lockout policies.

NOTE: Before disabling Basic Authentication, make sure that all scripts or tools that query the Integration Service API have been updated to use OAuth 2.0.

Role-based Access Control (RBAC) Configuration

Regardless of the authentication method you have chosen to use, the role-based permissions assigned to users is applied in the same way.

Assigning a Role to a Specific User

By applying a role permission to a single username in the admin panel, or through the API, permissions will be granted to any user who logs in through a provider matching that username.

Assigning Roles to a Specific User Group

By applying a role permission to a group name in the admin panel, or through the API, permissions will be granted to any user who logs in and is a member of the specified group.

For authentication to work properly, **group_search** must be configured in the authentication provider's connector information. When requesting authentication tokens, be sure to also request the **groups** claim.

NOTE: If a user belongs to multiple groups, with varying permissions defined to each group, the user will be permitted to do all actions provided by the group that provides the most roles.

Viewing User and Group Information

After configuring your third-party authentication connector in Dex, you can run the following command to view the search strategy and group results for any user that attempts to authenticate by looking at the Dexserver logs:

```
docker service logs -f iservices_dex
```

Changing Roles and Permissions

To change roles and permissions through the Integration Service user interface, go to the **Admin Panel** page (👥) to create and edit the roles and permissions of the user groups. For more information, see ***Creating a User Group in the Integration Service***.

To change roles and permissions through the API, refer to the **swagger.yml** file for API required parameters and endpoints to update roles and permissions.

Configuring Authentication Settings in the Integration Service

The following configuration settings can be configured and used with the Integration Service authentication and authorization strategies:

- **DEX_CONNECTORS**. This environment variable specifies which authentication providers Dex will use. For more information, see ***User Interface Login Using a Third-party Authentication Provider***.

- **BASIC_AUTH_USER**. This environment variable specifies the basic_auth and admin login username. For more information, see *User Interface Login Administrator User*.
- **BASIC_AUTH_PASSWORD**. This environment variable specifies the basic_auth and admin login password. For more information, see *User Interface Login Administrator User*.
- **BASIC_AUTH**. This environment variable specifies whether the local Administrator user will be enabled. For more information, see *User Interface Login Administrator User*.
- **CLIENT_ID**. The IS client_id in use by default is *isproxy*, and you should not change this value.
- **CLIENT_SECRET**. The client_secret is a randomly generated string unique to each Integration Service deployment. In most configurations, you do not need to change this secret.

User Groups, Roles, and Permissions

On the **Admin Panel** page (👥), you can edit and create ***user groups*** that define the different roles and permissions for your users. Depending on their assigned permissions, users have access to certain features, or they are blocked from certain features.

The available roles and permissions for the Integration Service include the following:

- ***View***. The user can view and get via the API the list of integration applications, the list of installed Synchronization PowerPacks, the dashboards, the reports, the configuration objects, and the results of application runs.
- ***Execute***. The user has all of the privileges of the **View** permission, but the user can also trigger application runs through the user interface or the API.
- ***Configuration***. The user has all of the privileges of the **Execute** permission, but the user can also add and edit configuration objects and modify the application variables.
- ***Developer***. The user has all of the privileges of the **Configure** permission, but the user can also add, copy, and edit step definitions; add, copy, and edit application definitions; and create Synchronization PowerPacks.
- ***Administrator***. The user has all of the privileges of the **Develop** permission, but the user can also add, install, activate, and delete Synchronization PowerPacks; delete applications, steps, and configuration objects; and access to (but not authorization to) the user interfaces for Couchbase, Flower, and RabbitMQ.

Additional information about roles and permissions in the Integration Service:

- Roles in the Integration Service are automatically assigned to a user based on the user's group with the external provider (such as LDAP or AD) .
- If an Integration Service system does not have an external provider configured, such as LDAP or Active Directory, that Integration Service system can only support a single *isadmin* Administrator user.
- The *only* password saved in the Integration Service system is the password for the *isadmin* Administrator user. All other user passwords are saved in the third-party authentication provider configured for the Integration Service.
- The authentication configuration used by the Integration Service is also supported in SL1.
- API queries made by users will also be checked for proper authorization.

- The Integration Service administrator can configure which users are in which roles.
- The Integration Service administrator can test and configure the LDAP and Active Directory settings to ensure that the settings work before proceeding.
- The Integration Service administrator can always log in with the *isadmin* Administrator user, even if the underlying LDAP provider is inaccessible.

Creating a User Group in the Integration Service

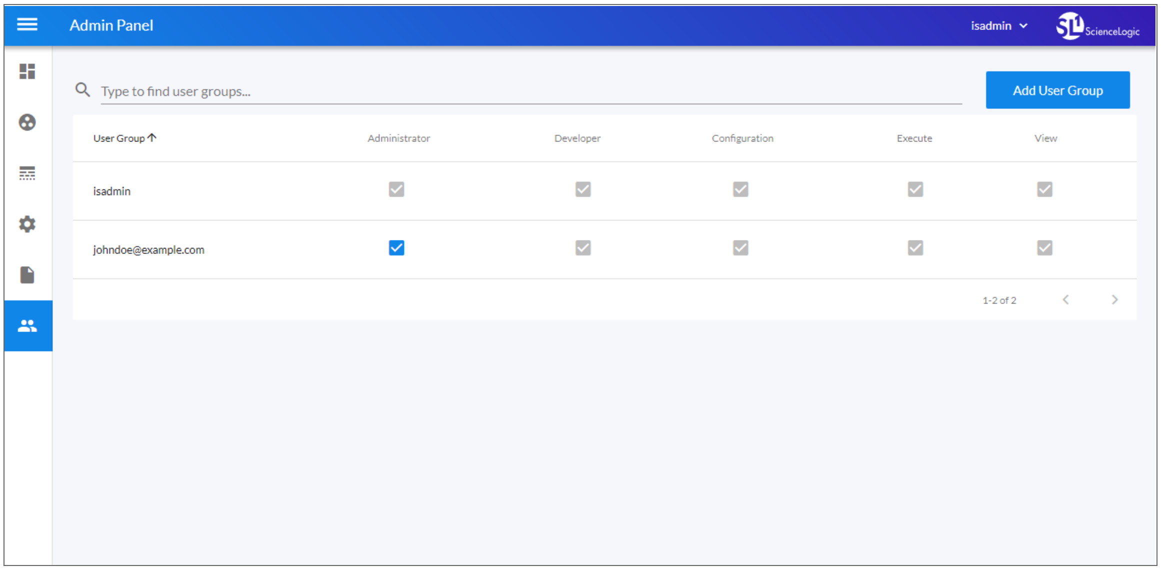
If you have the *Administrator* role, you can modify the permissions for each user group on the page. You cannot change the permissions for the user group to which you currently belong.

A user with the *Administrator* role can also create a user group and assign permissions to that group.

> **NOTE:** The *only* password saved in the Integration Service system is the password for the *isadmin* Administrator user. All other user passwords are saved in the third-party authentication provider configured for the Integration Service.

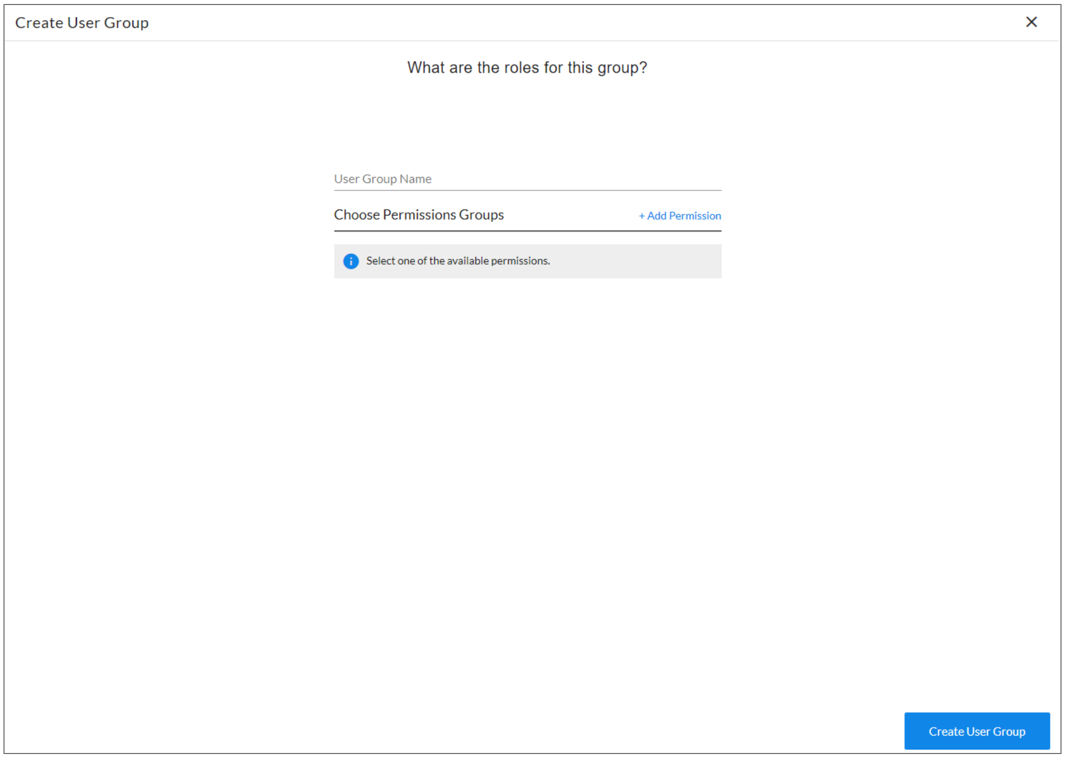
To create a user group:

1. In the Integration Service user interface, go to the **Admin Panel** page (👥):



User Group ↑	Administrator	Developer	Configuration	Execute	View
isadmin	☑	☑	☑	☑	☑
johndoe@example.com	☑	☑	☑	☑	☑

2. Click **[Add User Group]**. The **Create User Group** window appears.



Create User Group

What are the roles for this group?

User Group Name

Choose Permissions Groups

Select one of the available permissions.

3. Complete the following fields:
    - ***User Group Name***. Type a name for this user group, *without* spaces. This is the name that the users in this group will use when logging in to the Integration Service.
    - ***Choose Permissions Group***. Click **Add Permission** to select the permission to assign to this new user group. Your choices include *Developer*, *Configuration*, *View*, *Execute*, and *Administrator*, and these choices are defined in ***User Groups, Roles, and Permissions***.
4. Click **[Create User Group]**. The group is added to the **Admin Panel** page.
5. If you want to give a user group *more* permissions, select the empty checkbox (☐) for that role from the **Admin Panel** page. You must have the *Administrator* role to change these settings.
6. If you want to *remove* permissions from a user group, select the highest level role, which has a blue check mark (☑). The role to the right (with fewer permissions) becomes the new role for that user group.

Chapter 5

Managing Synchronization PowerPacks

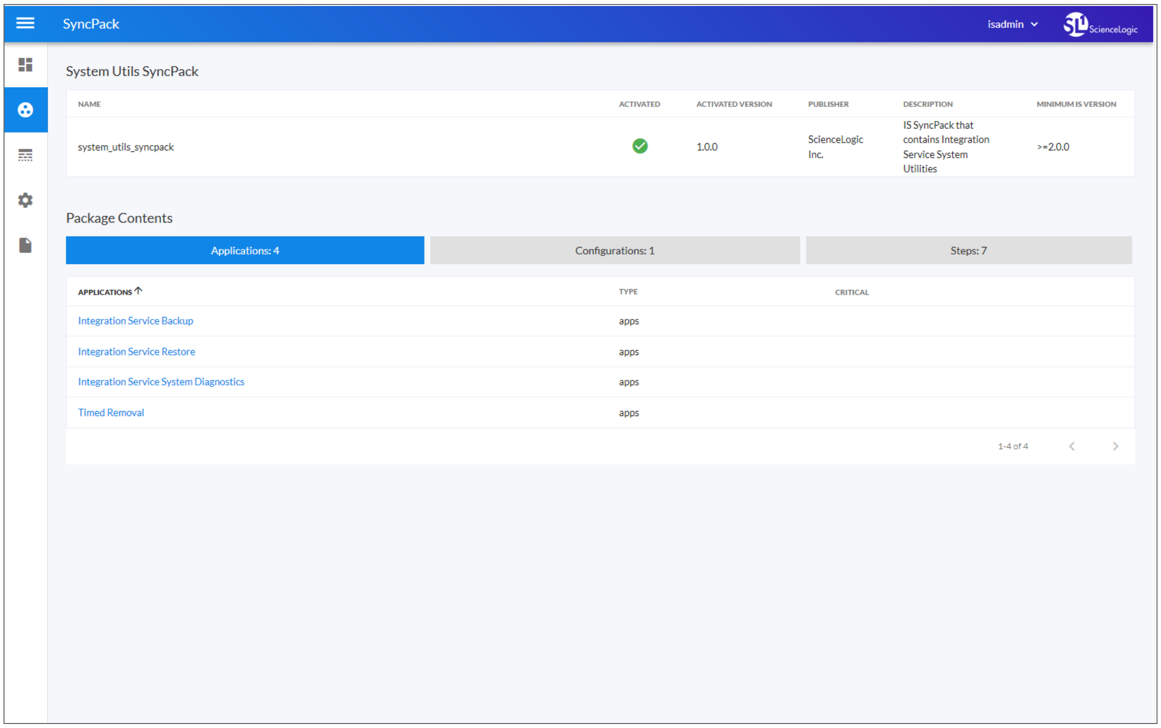
Overview

This chapter describes how to use the **SyncPacks** page ( ) to import, install, view, and edit Synchronization PowerPacks.

This chapter covers the following topics:

What is a Synchronization PowerPack?

A Synchronization PowerPack contains all of the code and logic needed to perform integrations on the Integration Service platform. A Synchronization PowerPack is saved as a Python **.whl** file, and it typically includes Python steps that are listed in integration applications, and then encapsulated within configurations:



System Utils SyncPack

NAME	ACTIVATED	ACTIVATED VERSION	PUBLISHER	DESCRIPTION	MINIMUM IS VERSION
system_utils_syncpack	✓	1.0.0	ScienceLogic Inc.	IS SyncPack that contains Integration Service System Utilities	>=2.0.0

Package Contents

APPLICATIONS ↑	TYPE	CRITICAL
Integration Service Backup	apps	
Integration Service Restore	apps	
Integration Service System Diagnostics	apps	
Timed Removal	apps	

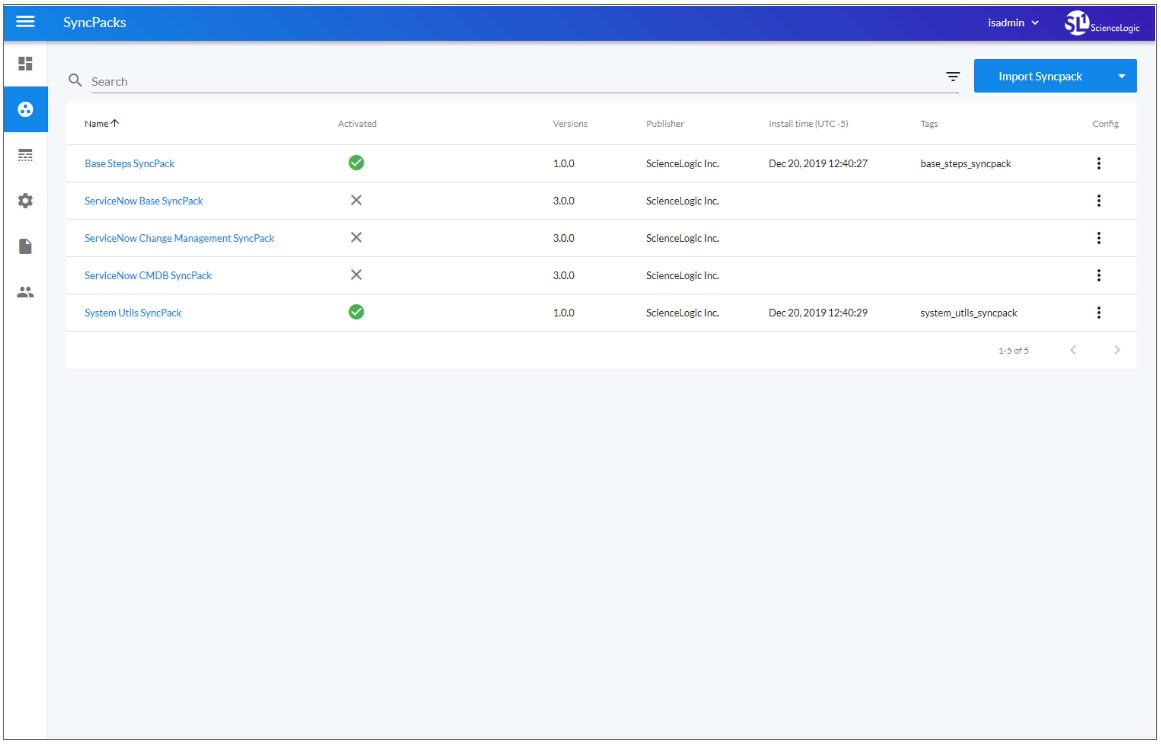
You can access the latest steps, integration applications, and configurations for the Integration Service or a third-party integration, such as ServiceNow, by downloading the most recent Synchronization PowerPack for that integration from ScienceLogic. You can access all Synchronization PowerPacks on the **SyncPacks** page.

A Synchronization PowerPack is highly customizable, and you can modify the contents of a Synchronization PowerPack to meet your business needs, instead of changing your SL1 configuration. You can categorize and segment your business integration solutions using a Synchronization PowerPack.

Synchronization PowerPacks let you distribute content to simplify installations, upgrades, and maintenance of integration solutions. Just like PowerPacks, you can also share Synchronization PowerPacks. If an Integration Service user creates a Synchronization PowerPack, that user can share the Synchronization PowerPack with other users through the ScienceLogic Customer Portal. In addition, Synchronization PowerPacks protect a company's intellectual property using licensing and encryption technologies.

Viewing the List of Synchronization PowerPacks

The **SyncPacks** page provides a list of the Synchronization PowerPacks on your Integration Service system. From this page you can import, install, and edit Synchronization PowerPacks:



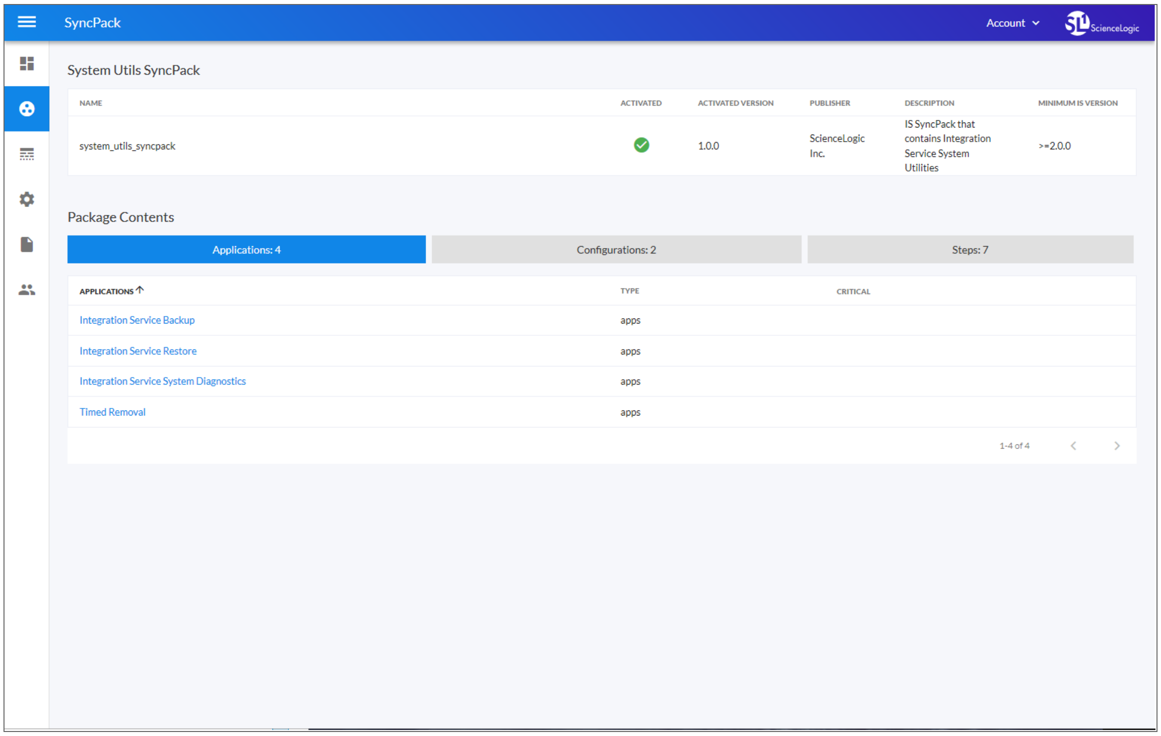
Name ↑	Activated	Versions	Publisher	Install time (UTC -5)	Tags	Config
Base Steps SyncPack	✓	1.0.0	ScienceLogic Inc.	Dec 20, 2019 12:40:27	base_steps_syncpack	⋮
ServiceNow Base SyncPack	✕	3.0.0	ScienceLogic Inc.			⋮
ServiceNow Change Management SyncPack	✕	3.0.0	ScienceLogic Inc.			⋮
ServiceNow CMDB SyncPack	✕	3.0.0	ScienceLogic Inc.			⋮
System Utils SyncPack	✓	1.0.0	ScienceLogic Inc.	Dec 20, 2019 12:40:29	system_utils_syncpack	⋮

You can search for a specific Synchronization PowerPack by typing the name of that Synchronization PowerPack in the ***Search*** field at the top of the **SyncPacks** page. The user interface filters the list as you type. You can also sort the list of Synchronization PowerPacks by clicking the headers of the columns.

TIP: Click the Filter icon (☰) at the top right of the **SyncPacks** page and click the ***Show all SyncPacks*** toggle to show all available Synchronization PowerPacks or to show only activated Synchronization PowerPacks. An *activated* Synchronization PowerPack has been installed and is ready to be used.

The **[Actions]** button (⋮) for a Synchronization PowerPack gives you the option to activate and install or uninstall that Synchronization PowerPack. Also, if you have older versions of the same Synchronization PowerPack, you can select *Change active version* to activate a different version other than the version that is currently running.

Click a Synchronization PowerPack from the list to view the **SyncPacks** page for the Synchronization PowerPack. From this page you can view the contents of the Synchronization PowerPack, including the applications, configuration objects, and steps included in the Synchronization PowerPack. You can also view additional metadata, including minimum Integration Service platform version and descriptions:



System Utils SyncPack

NAME	ACTIVATED	ACTIVATED VERSION	PUBLISHER	DESCRIPTION	MINIMUM IS VERSION
system_utils_syncpack	✓	1.0.0	ScienceLogic Inc.	IS SyncPack that contains Integration Service System Utilities	>=2.0.0

Package Contents

APPLICATIONS ↑	TYPE	CRITICAL
Integration Service Backup	apps	
Integration Service Restore	apps	
Integration Service System Diagnostics	apps	
Timed Removal	apps	

Importing and Installing a Synchronization PowerPack

The **SyncPacks** page in the Integration Service user interface lets you import and install the latest version of a Synchronization PowerPack. After you install a Synchronization PowerPack, the Integration Service considers that Synchronization PowerPack to be ***activated*** and ready to be used. By default, the **SyncPacks** page displays all activated Synchronization PowerPacks.

After you install a Synchronization PowerPack, that Synchronization PowerPack is available on the **SyncPacks** page of the Integration Service user interface. You can click the name of that Synchronization PowerPack to view the detail page for that Synchronization PowerPack. On the Synchronization PowerPack detail page, you can view metadata about the Synchronization PowerPack, and you can also view the integration applications, configuration objects, and steps for that Synchronization PowerPack. Also, the integration applications from that Synchronization PowerPack are added to the **Integrations** page of the user interface.

You can click the ***Show All SyncPacks*** toggle (☰) at the top of the **SyncPacks** page to view all versions of the available Synchronization PowerPacks. A Synchronization PowerPack must be a Python **.whl** file to upload and install in the Integration Service.

NOTE: You must have the *Develop* or *Administrator* role to install a Synchronization PowerPack. For more information, see ***Managing Users in the Integration Service***.

NOTE: If your Integration Service system uses self-signed certificates, you will need to manually accept the certificate before you can upload Synchronization PowerPacks. Go to **https://<*IP address of Integration Service*>:3141/isadmin**, accept the certificate, and then exit out of the tab. When you log into the Integration Service again, you will be able to upload Synchronization PowerPacks.

Importing a Synchronization PowerPack

WARNING: If you are *upgrading* to a newer version of a Synchronization PowerPack from a previous version, make a note of any settings you made on the **Configuration** tab for the various integration applications in that Synchronization PowerPack, as those settings are *not* retained when you upgrade.

NOTE: You must import and install the *ServiceNow Base* Synchronization PowerPack before uploading and installing any of the other ServiceNow Synchronization PowerPacks.

To import a Synchronization PowerPack in the Integration Service user interface:

1. On the **SyncPacks** page of the Integration Service user interface, click **[Import SyncPack]**. The **Import SyncPack** page appears.
2. Click **[Browse]** and select the **.whl** file for the Synchronization PowerPack you want to install.

TIP: You can also drag and drop a **.whl** file to the **SyncPacks** page.

3. Click **[Import]**. The Integration Service registers and uploads the Synchronization PowerPack. The Synchronization PowerPack is added to the **SyncPacks** page.

NOTE: You cannot edit the content package in a Synchronization PowerPack published by ScienceLogic. You must make a copy of a ScienceLogic Synchronization PowerPack and save your changes to the new Synchronization PowerPack to prevent overwriting any information in the original Synchronization PowerPack when upgrading.

Installing a Synchronization PowerPack

To install a Synchronization PowerPack in the Integration Service user interface:

1. On the **SyncPacks** page of the Integration Service user interface, click the **[Actions]** button (⋮) for the Synchronization PowerPack you want to install and select *Activate & Install*. The **Activate & Install SyncPack** modal appears.

> **TIP:** By default, the **SyncPacks** page displays only activated and installed PowerPacks. If you do not see the PowerPack that you want to install, click the toggle icon (≡) on the **SyncPacks** page and select *Show All SyncPacks* to see a list of the uninstalled PowerPacks.

2. Click **[Yes]** to confirm the activation and installation. When the Synchronization PowerPack is activated, the **SyncPacks** page displays a green check mark icon (✓) for that Synchronization PowerPack. If the activation or installation failed, then a red exclamation mark icon (❗) appears.
3. For more information about the activation and installation process, click the check mark icon (✓) or the exclamation mark icon (❗) in the **Activated** column for that Synchronization PowerPack. For a successful installation, the "Activate & Install SyncPack" integration application appears, and you can view the Step Log for the steps. For a failed installation, the **Error Logs** window appears.

Default Synchronization PowerPacks

When you install the Integration Service, the following Synchronization PowerPacks are added to the **SyncPacks** page by default:

- *Base Steps* Synchronization PowerPack
- *System Utils* Synchronization PowerPack

If you need to install these Synchronization PowerPacks, click the **[Actions]** button (⋮) and select *Activate & Install* for each Synchronization PowerPack.

Base Steps Synchronization PowerPack

The *Base Steps* Synchronization PowerPack version 1.0.0 contains the "Template App" integration application. You can use the "Template App" integration application as a template for building new integration applications.

The *Base Steps* Synchronization PowerPack also includes the following steps, which you can use in new and existing integration applications:

- Cache Delete
- Cache Read

- Cache Save
- Direct Cache Read
- MS-SQL Describe
- MS-SQL Insert
- MS-SQL Select
- MySQL Describe
- MySQL Insert
- MySQL Select
- Query GraphQL
- Query REST
- Run a command through an SSH tunnel
- Trigger Application

> TIP: To view the code for a step, select a Synchronization PowerPack from the **SyncPacks** page, click the **[Steps]** tab, and select the step you want to view.

System Utils Synchronization PowerPack

The *System Utils* Synchronization PowerPack version 1.0.0 is a Standard Synchronization PowerPack that contains integration applications, a configuration object, and steps.

> NOTE: You must install the *Base Steps* Synchronization PowerPack before you can install this Synchronization PowerPack.

The *System Utils* ... Synchronization PowerPackincludes the following integration applications:

- ***Integration Service Backup***. Creates a backup file of the Couchbase database used by the Integration Service and sends the file to a remote location. For more information, see ***Backing up Data***.
- ***Integration Service Restore***. Restores a Couchbase backup file that was created up with the "Integration Service Backup" integration application. For more information, see ***Restoring a Backup***.
- ***Integration Service System Diagnostics***. Displays a report of platform diagnostics for the services used by the Integration Service. For more information, see ***Viewing Integration Service System Diagnostics***.
- ***Timed Removal***. Removes logs from Couchbase on a regular schedule. For more information, see ***Removing Logs on a Regular Schedule***.

This Synchronization PowerPack includes the "IS - System Diagnostic Configuration Example" configuration object, which contains the structure needed for the "Integration Service System Diagnostics" integration application.

This Synchronization PowerPack also includes the following steps, which are used in the four integration applications listed above:

- Create Couchbase Backup
- Diagnostic reporter
- Collect IS Diagnostics
- Query Application Logs and Remove
- Query an IS Node
- Query IS Manager Node
- Restore Couchbase From Backup

Chapter 6

Managing Integration Applications

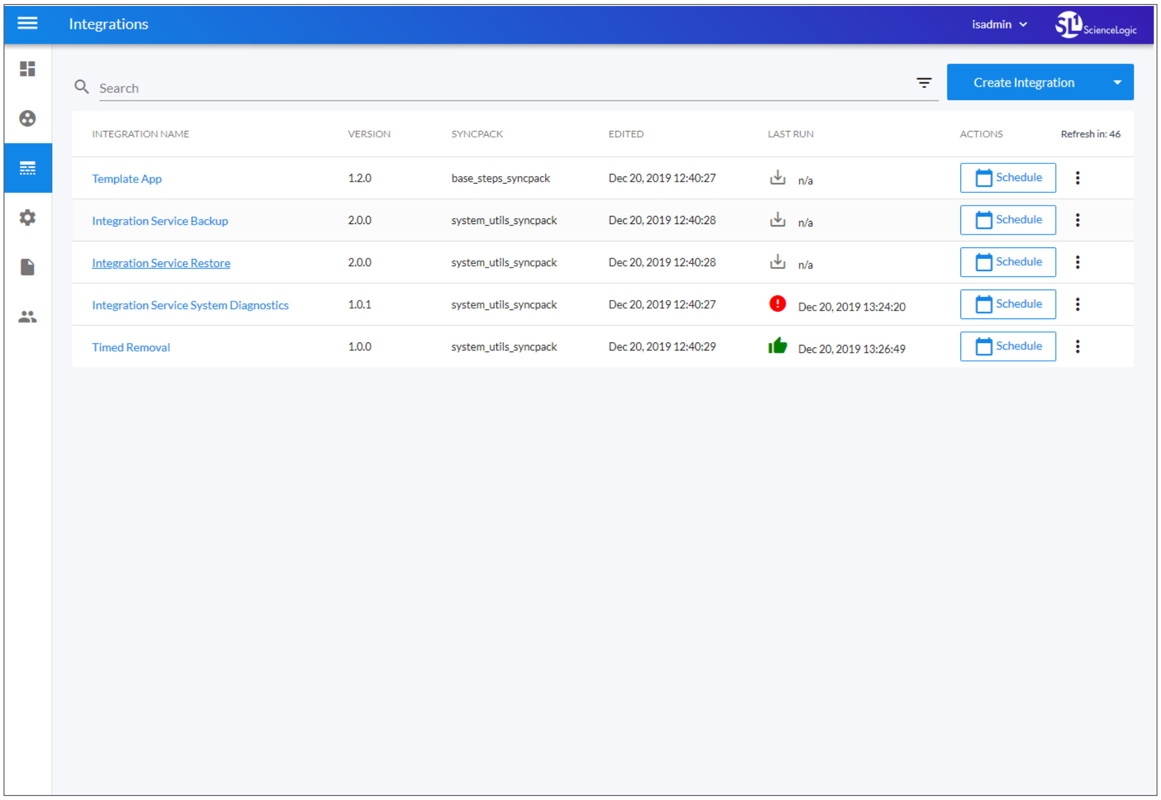
Overview

This chapter describes how to use the **Integrations** page ( ) of the Integration Service user interface to run, schedule, and create integration applications.

This chapter covers the following topics:

Viewing the List of Integration Applications

The **Integrations** page (☰) provides a list of the integration applications on your Integration Service system. From this page you can schedule, edit, view, and create integration applications and steps:



INTEGRATION NAME	VERSION	SYNCPACK	EDITED	LAST RUN	ACTIONS	Refresh in: 46
Template App	1.2.0	base_steps_syncpack	Dec 20, 2019 12:40:27	n/a	Schedule	⋮
Integration Service Backup	2.0.0	system_utils_syncpack	Dec 20, 2019 12:40:28	n/a	Schedule	⋮
Integration Service Restore	2.0.0	system_utils_syncpack	Dec 20, 2019 12:40:28	n/a	Schedule	⋮
Integration Service System Diagnostics	1.0.1	system_utils_syncpack	Dec 20, 2019 12:40:27	● Dec 20, 2019 13:24:20	Schedule	⋮
Timed Removal	1.0.0	system_utils_syncpack	Dec 20, 2019 12:40:29	✓ Dec 20, 2019 13:26:49	Schedule	⋮

You can search for a specific integration application by typing the name of that in the ***Search*** field at the top of the **Integrations** page. The user interface filters the list as you type. You can also sort the list of integration applications by clicking the headers of the columns.

The **SyncPack** column on the **Integrations** page lists the name of the Synchronization PowerPack to which a specific integration application belongs, where relevant. You can click the name of the Synchronization PowerPack to which an integration application belongs to go to the Synchronization PowerPack page for that pack.

Some of the integration applications on the **Integrations** page 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** page and select *Show Hidden Integrations*. Internal integration applications are hidden by default.

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.

TIP: You can click **[Run]** (▶) to run an integration application and then navigate to another page in the Integration Service, and the integration application will complete on its own.

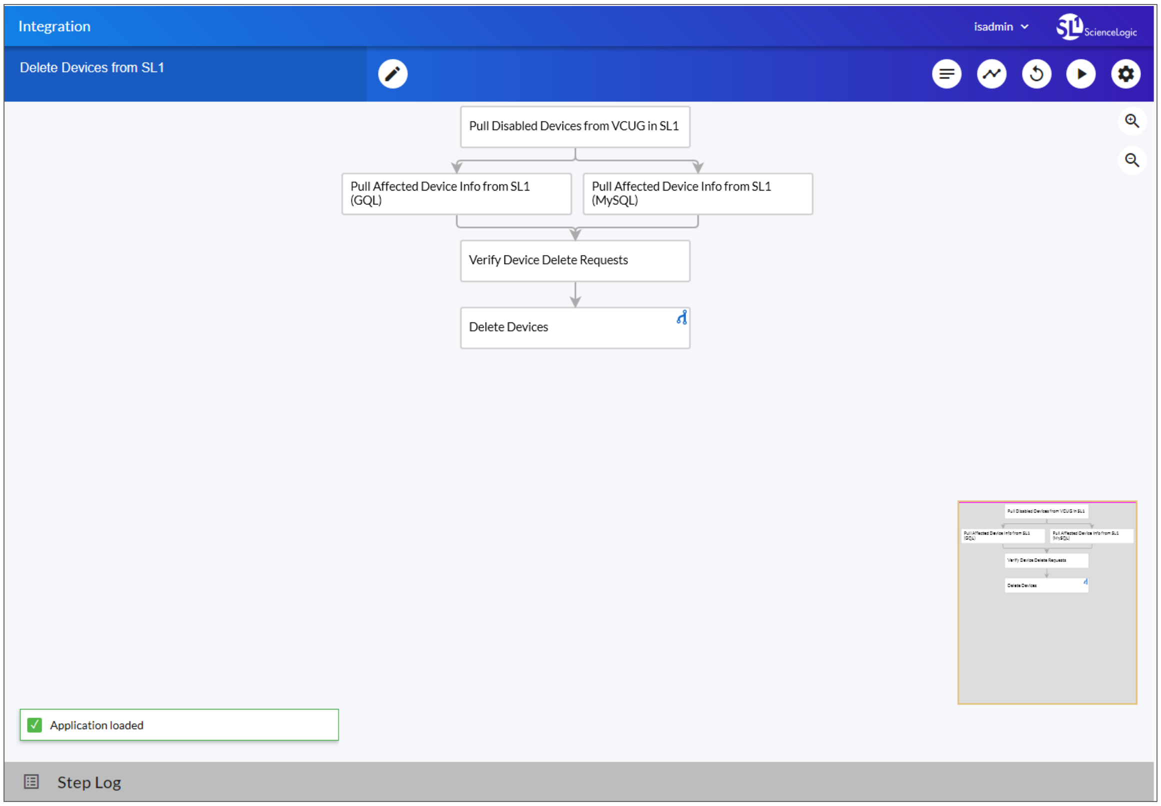
The **[Schedule]** button lets you use the Scheduler to define how often or at what time to run an integration application. A scheduled integration application displays a check mark in the **[Schedule]** button on this page. For more information, see ***Scheduling an Integration Application***.

The **[Actions]** button (⋮) for an integration application gives you the option to run, edit, or delete that integration application. You cannot run an integration application in Debug Mode or with run an application with custom attributes from this menu.

TIP: To open the **Notification Center** pane, which contains a list of all of the pop-up messages about integration applications that were run successfully or with warnings or failures, click your user name in the navigation bar in the top right of any window and select *Notifications*. The different notifications are color-coded: green for success, yellow for warning, and red for failure. The number of notifications displays as a badge in the menu. For more information about a notification, click the link for the page where the notification appeared and review the **Step Log** details for the application steps.

The Integration Application Window

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


Integration
isadmin
ScienceLogic
Delete Devices from SL1
Pull Disabled Devices from VCUG in SL1
Pull Affected Device Info from SL1 (GQL)
Pull Affected Device Info from SL1 (MySQL)
Verify Device Delete Requests
Delete Devices
Step Log

✓ Application loaded

The **Integration Application** page contains the logic for the integration application. In the main viewing pane, the steps for the application are organized as a flowchart. Each rectangular block is a step, and the arrows indicate the order in which the steps will execute when you run the application. The color of each step changes to show the progress of the integration application run as it runs: green for success, red for failure, and blue for running.

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 at the top right of the **Integration Application** page include the following:

- **[Open Editor]** (). Opens the **Steps Registry** pane for editing on the left side of the window. After you click the button, the following buttons appear:

    - **[Close Editor]** (ⓧ). Closes the **Steps Registry** pane.
    - **[Edit Metadata]** (🗎). Opens the **Integration Metadata** window for that integration application so you can update the description or version of the application.
    - **[Save]**. Lets you save any changes to the steps and integration application. You can also save the edited application as a new application with new metadata using the *Save as* option.

  For more information, see *Editing an Integration Application*.

- **[Reports]**(≡). Displays a report of the results of this application, where applicable. For more information, see *Viewing a Report for an Integration Application*.
- **[Timeline]** (〰). Opens the Timeline view at the top of the window where you can view past runs of this application, and whether the application failed or succeeded. For more information, see *Viewing Previous Runs of an Integration Application*.
- **[Replay]** (↻). Replays the last run of this integration application. For more information, see *Viewing Previous Runs of an Integration Application*.
- **[Run]** (▶). Runs the integration application if you click the button without hovering over it. If you hover over the **[Run]** button, you can choose from the following options:
    - *Debug Run*. Run the integration application in Debug Mode, which provides more log data in the Step Logs to help you with troubleshooting.
    - *Custom Run*. Run the integration application with custom parameters for testing or troubleshooting.

  For more information, see *Running an Integration Application*.

- **[Configure]** (⚙). Opens the **Configuration** pane for the integration application, where you can change the configuration object aligned with the integration application and edit other fields as needed. For more information, see *Aligning a Configuration Object with an Integration Application*.

TIP: Click the **Zoom** icons (🔍 , 🔍) to change the size of the flowchart of steps.

In the bottom left-hand corner of the page, pop-up messages appear temporarily with the status of the application or an action. In the example above, the status is "Application loaded".

The **Step Log** pane displays the time, the type of log, and the log messages from a step that you selected in the main pane. All times that are displayed in this pane are in seconds.

TIP: For longer log messages, click the down arrow icon (⌄) in the **Message** column of the **Step Log** pane to open the message. Also, you can triple-click a log message to highlight the entire text block so you can easily copy the log message to another text viewer.

To generate more detailed logs when you run an integration application, select *Debug Run* by hovering over **[Run]** (▶).

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.

Editing an Integration Application

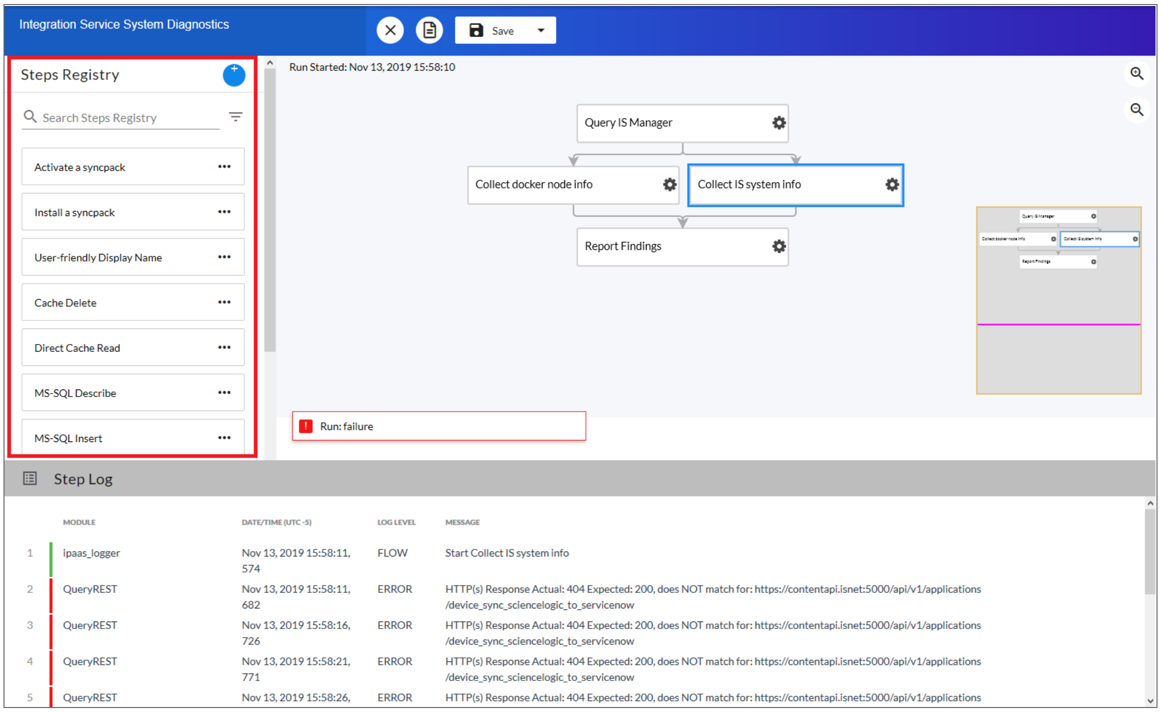
On the **Integrations** page, you can edit an existing integration application and its steps. You can also add and remove steps from that application.

> **NOTE:** You cannot overwrite integration applications where *ScienceLogic, Inc.* is listed as the Author, but you can edit a ScienceLogic application and save it with a different name.

To edit an integration application:

1. From the **Integrations** page, select the integration application that you want to edit. The **Integration Application** page for that application appears.

2. Click the **[Open Editor]** button (✎). The **Steps Registry** pane appears with a list of all of the steps that are available for that integration application:



Run Started: Nov 13, 2019 15:58:10

Run: failure

	MODULE	DATE/TIME (UTC -5)	LOG LEVEL	MESSAGE
1	ipaas_logger	Nov 13, 2019 15:58:11, 574	FLOW	Start Collect IS system info
2	QueryREST	Nov 13, 2019 15:58:11, 682	ERROR	HTTP(s) Response Actual: 404 Expected: 200, does NOT match for: https://contentapi.isnet:5000/api/v1/applications /device_sync_sciencelogic_to_servicenow
3	QueryREST	Nov 13, 2019 15:58:16, 726	ERROR	HTTP(s) Response Actual: 404 Expected: 200, does NOT match for: https://contentapi.isnet:5000/api/v1/applications /device_sync_sciencelogic_to_servicenow
4	QueryREST	Nov 13, 2019 15:58:21, 771	ERROR	HTTP(s) Response Actual: 404 Expected: 200, does NOT match for: https://contentapi.isnet:5000/api/v1/applications /device_sync_sciencelogic_to_servicenow
5	QueryREST	Nov 13, 2019 15:58:26,	ERROR	HTTP(s) Response Actual: 404 Expected: 200, does NOT match for: https://contentapi.isnet:5000/api/v1/applications

3. Scroll through the **Steps Registry** pane or use the **Search** field at the top of the pane to find the step you want to add.

> TIP: Click the **[Actions]** button (•••) on a step to view more information about that step, including the step ID, the Synchronization PowerPack for that step, and the step version author. You can also click **[Edit Step Code]** to edit the code for that step.

4. Click the steps you want to add from the **Steps Registry** pane and drag them to the main viewing pane of the **Integration Application** page to add them to the integration application.
5. 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.
6. To edit the configuration for a step, click the gear icon (⚙) on the step and update the fields as needed. You can add test data to the step and click **Custom Run** to run test data for that step.
7. To redirect the arrows connecting the steps, click an arrow and drag it to reposition it.
8. To remove a step, click the step to select it and press the **[Delete]** key on your keyboard.
9. To save the changes you made to the integration application, click **[Save]**. You can also click the *Save as* option to save the application with a new name.
10. To stop editing and close the **Search Steps Registry** panel, click the **[Close Editor]** button (⊗).

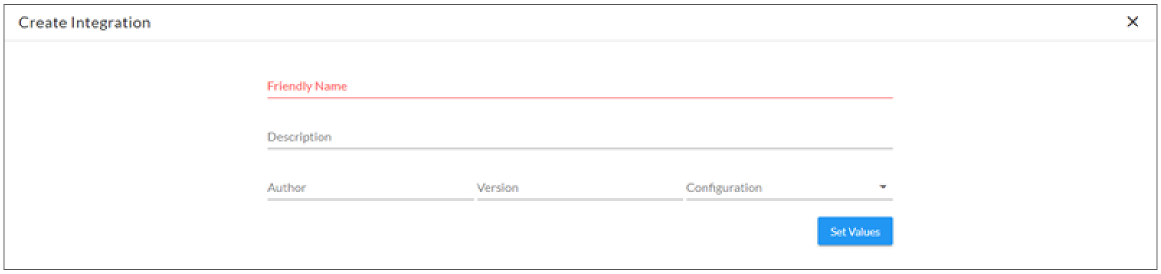
Creating an Integration Application

On the **Integrations** page, you can create new integration applications.

> **NOTE:** All Python step code should be Python 3.7 or later.

To create an integration application:

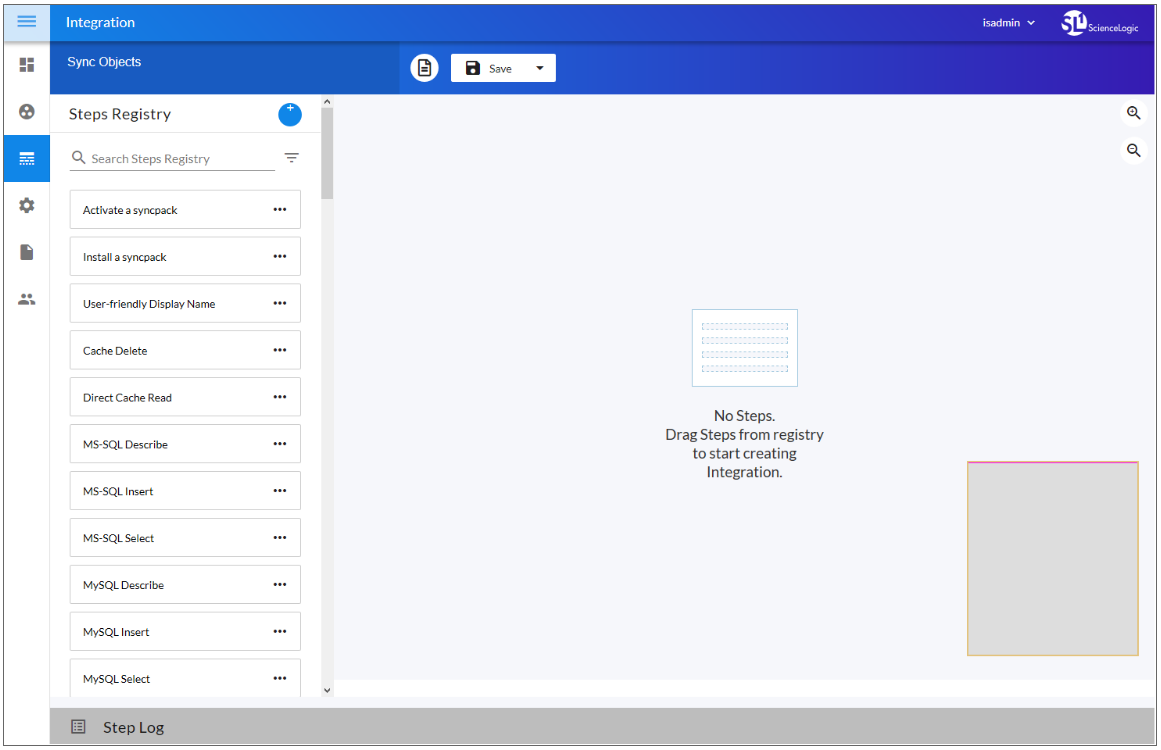
1. From the **Integrations** page ( ), click **[Create Integration]**. A **Create Integration** window appears:



2. Complete the following fields:

- ***Friendly Name***. The name that you want users to see for this integration application.
- ***Description***. A short description of what this integration application does.
- ***Author***. The name of the person or company that created this integration application. Use the same name for multiple integration applications.
- ***Version***. The version for this integration application.
- ***Configuration***. Select a configuration object to align with the new integration application.

3. Click **[Set Values]** to add the integration application to the Integration Service. The **Integration Application** page and the **Steps Registry** pane appear:



4. Click the steps you want to add from the **Steps Registry** pane and drag them to the main viewing pane of the **Integration Application** page to add them to the integration application.
5. 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.
6. To edit the configuration for a step, click the gear icon (⚙) on the step and update the fields as needed. You can add test data to the step and click **Custom Run** to run test data for that step.
7. To add an arrow from one step to another, click and drag the mouse from the first step to the second step. An arrow appears between the two steps, which you can click and drag to reposition.
8. When you are done adding steps to the new integration application, click **[Save]**. The integration application is added to the **Integrations** page.

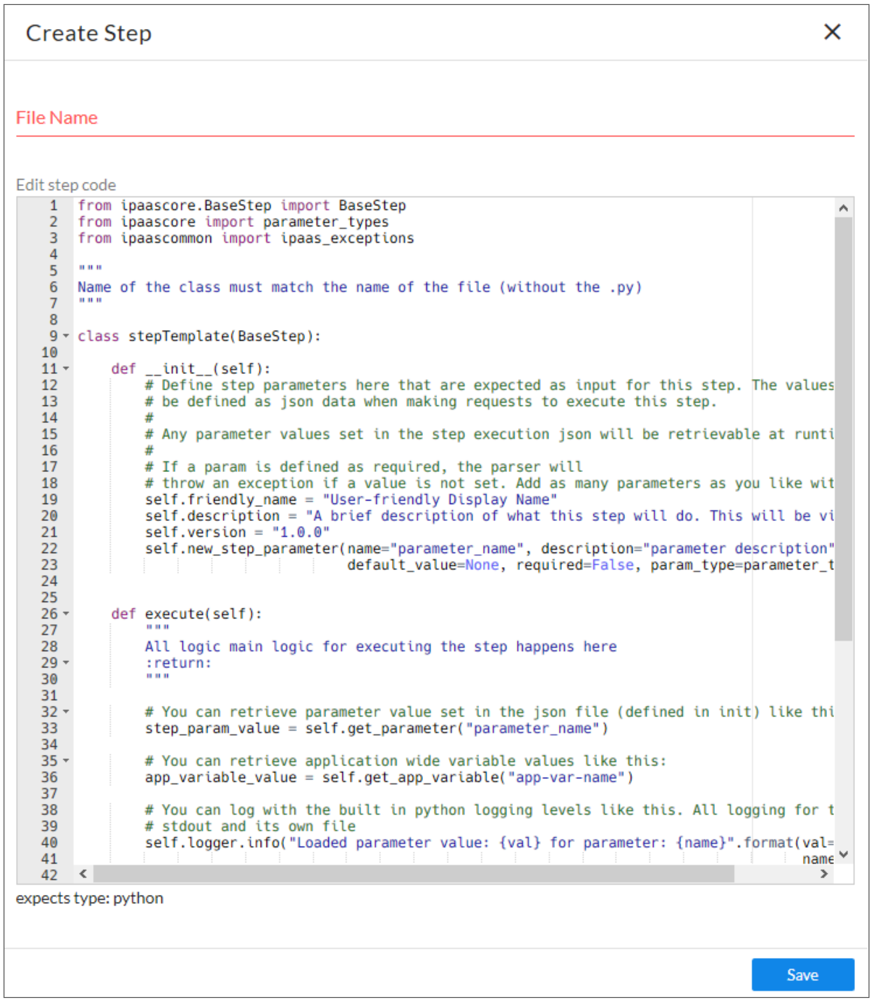
NOTE: The user interface will reject integration application names with special characters when you are creating or updating an integration application.

Creating a Step

On the **Integrations** page, you can create new steps that you can use in new or existing integration applications.

To create a step:

1. From the **Integrations** page ( ), click **[Create Integration]**. A **Create Step** window appears:



```python
from ipaascore.BaseStep import BaseStep
from ipaascore import parameter_types
from ipaascommon import ipaas_exceptions

"""
Name of the class must match the name of the file (without the .py)
"""

class stepTemplate(BaseStep):

    def __init__(self):
        # Define step parameters here that are expected as input for this step. The values
        # be defined as json data when making requests to execute this step.
        #
        # Any parameter values set in the step execution json will be retrievable at runti
        #
        # If a param is defined as required, the parser will
        # throw an exception if a value is not set. Add as many parameters as you like wit
        self.friendly_name = "User-friendly Display Name"
        self.description = "A brief description of what this step will do. This will be vi
        self.version = "1.0.0"
        self.new_step_parameter(name="parameter_name", description="parameter description"
                                default_value=None, required=False, param_type=parameter_t


    def execute(self):
        """
        All logic main logic for executing the step happens here
        :return:
        """

        # You can retrieve parameter value set in the json file (defined in init) like thi
        step_param_value = self.get_parameter("parameter_name")

        # You can retrieve application wide variable values like this:
        app_variable_value = self.get_app_variable("app-var-name")

        # You can log with the built in python logging levels like this. All logging for t
        # stdout and its own file
        self.logger.info("Loaded parameter value: {val} for parameter: {name}".format(val=
                                                                                      name
```

TIP: You can also create a step from an **Integration Application** page by clicking the **[Open Editor]** button ( ) and then clicking the **[Create Step]** button ( ) on the **Steps Registry** pane.

2. In the ***File Name*** field, type a name for the step. The name cannot include spaces.
3. In the ***Edit step code*** text box, update the Python code for the step as needed, including the metadata information for the new step in the `def __init__(self):` section. For more information, see the "Creating a Step" chapter in the ***Integration Service for Developers Manual***.
4. Click **[Save]**. The step is added to the **Steps Registry** pane.

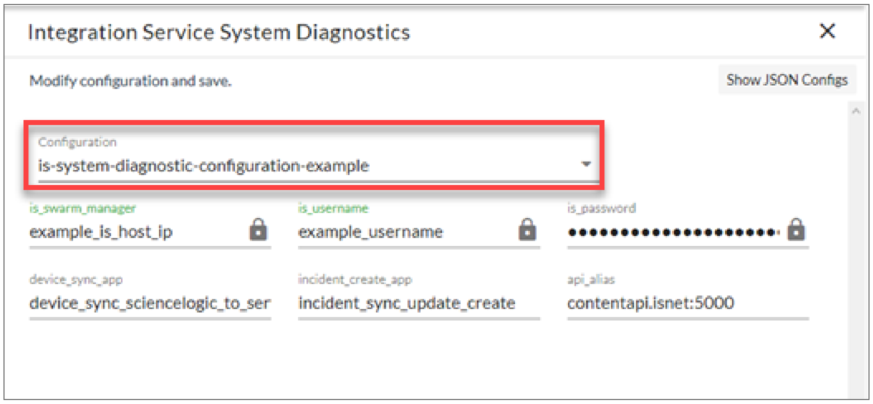
Aligning a Configuration Object with an Integration Application

Before you can run an integration application, you must align the application with a configuration object from the **Configurations** page. A ***configuration object*** defines global variables, such as endpoints and credentials, that can be used by multiple steps and integration applications. Each variable in a configuration object is set up as a name and value pair. You can also encrypt a variable to protect sensitive data like a password.

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

To align a configuration object with an integration application:

1. From the **Integrations** page ( ), select the integration application that you want to align with a configuration object. The **Integration Application** page for that integration application appears.
2. Click **[Configure]** ( ). The **Configuration** pane opens on the right side of the **Integration Application** page:



Modify configuration and save.

Configuration: is-system-diagnostic-configuration-example

is_swarm_manager: example_is_host_ip

is_username: example_username

is_password: ••••••••••••••••••••

device_sync_app: device_sync_sciencelogic_to_ser

incident_create_app: incident_sync_update_create

api_alias: contentapi.isnet:5000

> **TIP:** To view a pop-up description of a field on the **Configuration** pane for an integration application, hover over the label name for that field.

3. Select a configuration from the ***Configuration*** dropdown to "align" to this integration application. This step is required for all integration applications.

TIP: You can select *none* from the ***Configuration*** dropdown to clear or "un-align" the selected configuration object from an integration application. Also, if you did not select a configuration object when editing fields on the **Configuration** pane, the previously set configuration object will remain aligned (if there was a previously set configuration object).

4. Click **[Show JSON Configs]** to view the JSON configuration data for the configuration object. Click **[Hide JSON Editor]** again to view the fields instead.
5. 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, any fields that are encrypted and any configuration-specific fields containing a padlock icon (🔒) on the **Configuration** pane cannot be edited.

6. Click **[Save]** and wait for the "App & Config modifications saved" pop-up message to appear. The **Configuration** pane automatically closes after this message appears.

Running an Integration Application

You can run an integration application directly from the **Integrations** page (the list view) or from the **Integration Application** page (the detail view). If you run the integration application from the **Integration Application** page, you have the following additional options:

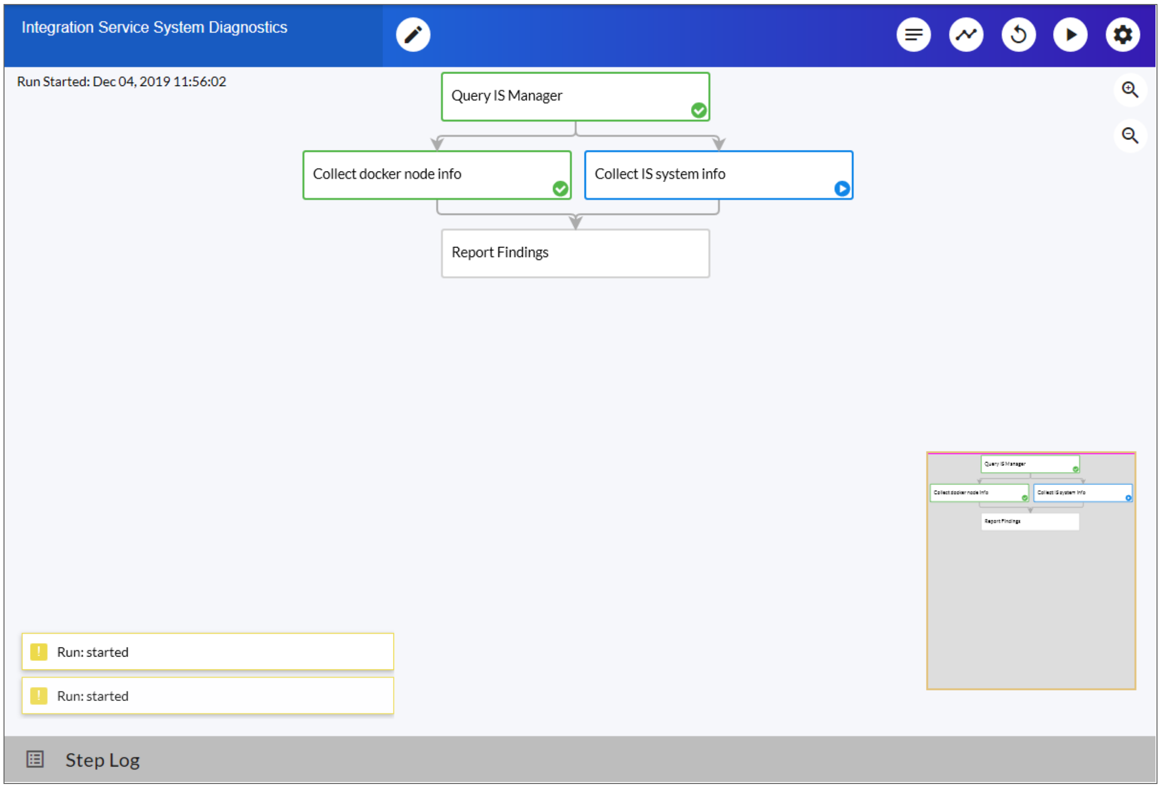
- ***Run***. Executes the integration application normally, with a log level of 1. This is the default, and it is the same as the *Run Now* option from the **Integrations** page.
- ***Debug Run***. Executes the integration application in Debug Mode with a log level of 10.
- ***Custom Run***. Executes the integration application using a logging level that you specify (Error, Warning, Info, or Debug). You can also add any customer parameters that you might want to use to test specific features in the integration application.

To run an integration application:

1. From the **Integrations** page ( ), click the **[Actions]** button (⋮) for the integration application you want to run and select *Run Now*.

TIP: You can also select an application from the **Integrations** page and click **[Run]** (▶) from the **Integration Application** page. If you hover over the **[Run]** button, you can select *Debug Run* or *Custom Run*.

2. As the application runs, the color of the border around each step represents whether it is running, is successful, or has failed:



Run Started: Dec 04, 2019 11:56:02

! Run: started

! Run: started

Step Color	Icon	State
Blue	▶	Running
Green	✓	Successful
Red	❗	Failed
Yellow	!	Warning

> **NOTE:** Pop-up status messages also appear in the bottom left-hand corner of the **Integration Application** page to update you on the progress of the application run.

3. 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".

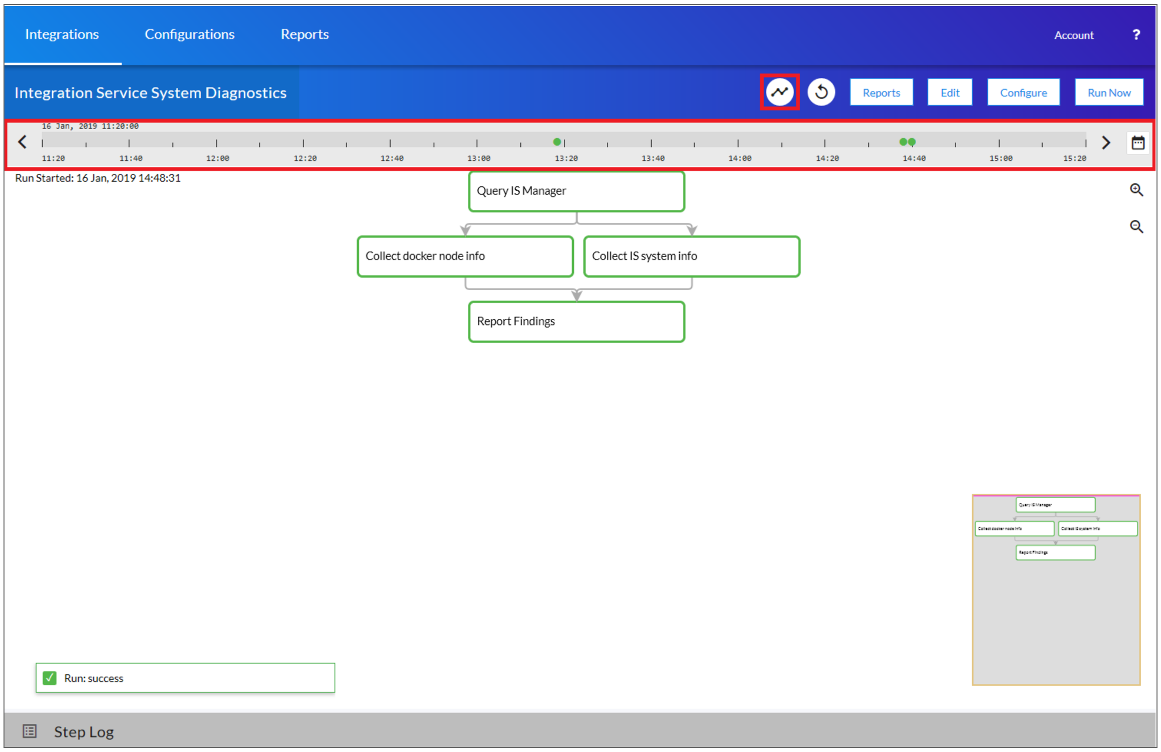
Viewing Previous Runs of an Integration Application

The **Integration Application** page for a selected integration application contains a Timeline feature that displays a history of previous runs of that application.

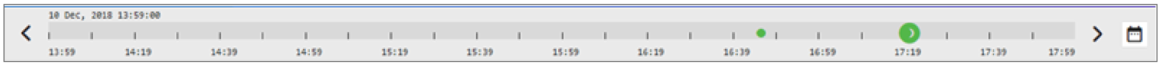
Also, you can click **[Replay]** (↻) to run that integration application again, such as when an integration application failed.

To view and filter the Timeline:

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

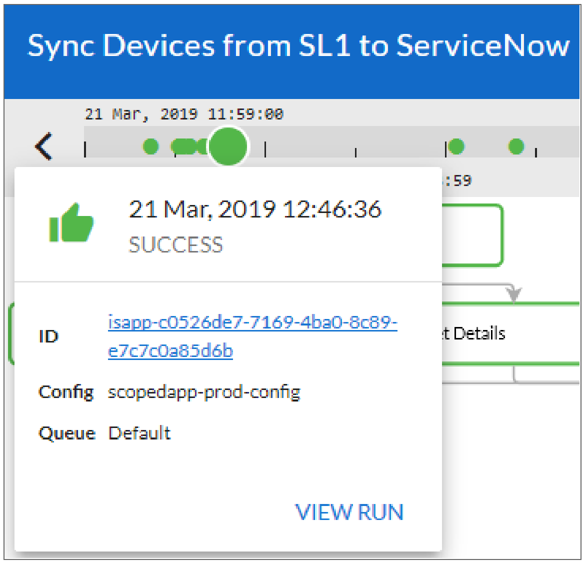


2. The default view for the Timeline shows the last two hours of runs for that integration application. The image above shows the last four hours of runs. Use the left arrow icon (‹) and the right arrow icon (›) to move through the Timeline in 15-minute increments:



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:



21 Mar, 2019 12:46:36
SUCCESS

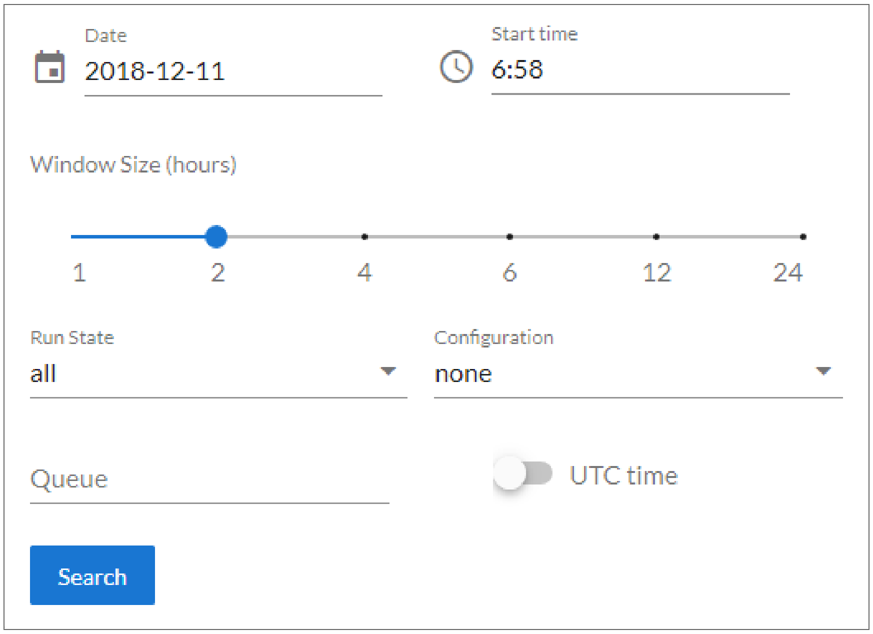
ID isapp-c0526de7-7169-4ba0-8c89-e7c7c0a85d6b

Config scopedapp-prod-config

Queue Default

TIP: Click the link for the run ID or click **View Run** to open the **Integration Application** page for that specific run of the integration application. The run ID also displays in the **Step Log** pane for the "triggering" step for that application, and it also appears at the end of the URL for that 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:



Date 2018-12-11

Start time 6:58

Window Size (hours) 2

Run State all

Configuration none

Queue

UTC time

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 **[Search]** to run the filter or search.

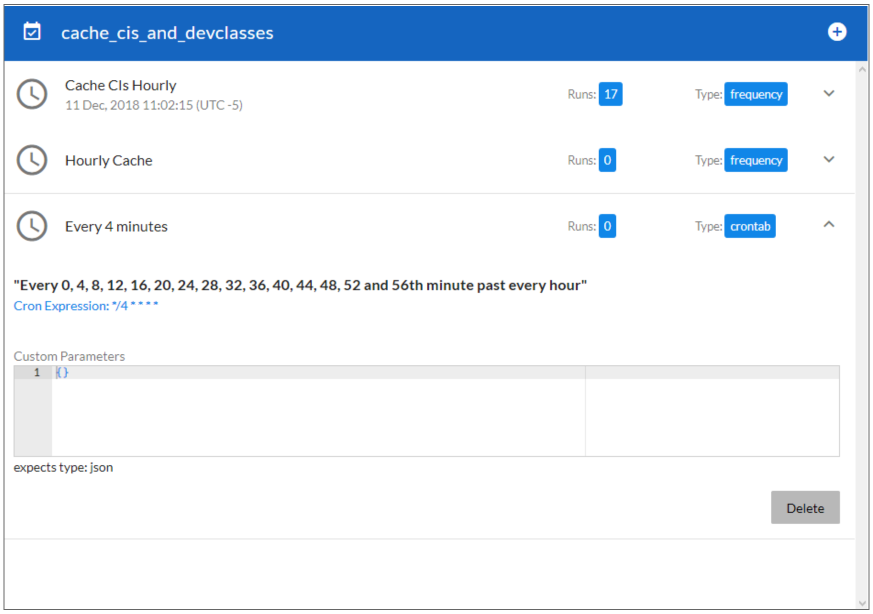
TIP: If the Timeline is open and you want to close it, click the **[Timeline]**button (📈).

Scheduling an Integration Application

You can create one or more schedules for a single integration application in the Integration Service user interface. When creating each schedule, you can specify the queue and the configuration file for that integration application.

To schedule an integration application:

1. On the **Integrations** page ( ), click **[Schedule]** for the integration application you want to schedule. The **Schedule** window appears, displaying any existing schedules for that application:



cache_cis_and_devclasses

Cache CIs Hourly
11 Dec, 2018 11:02:15 (UTC -5) Runs: 17 Type: frequency

Hourly Cache Runs: 0 Type: frequency

Every 4 minutes Runs: 0 Type: crontab

"Every 0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52 and 56th minute past every hour"
Cron Expression: */4 * * * *

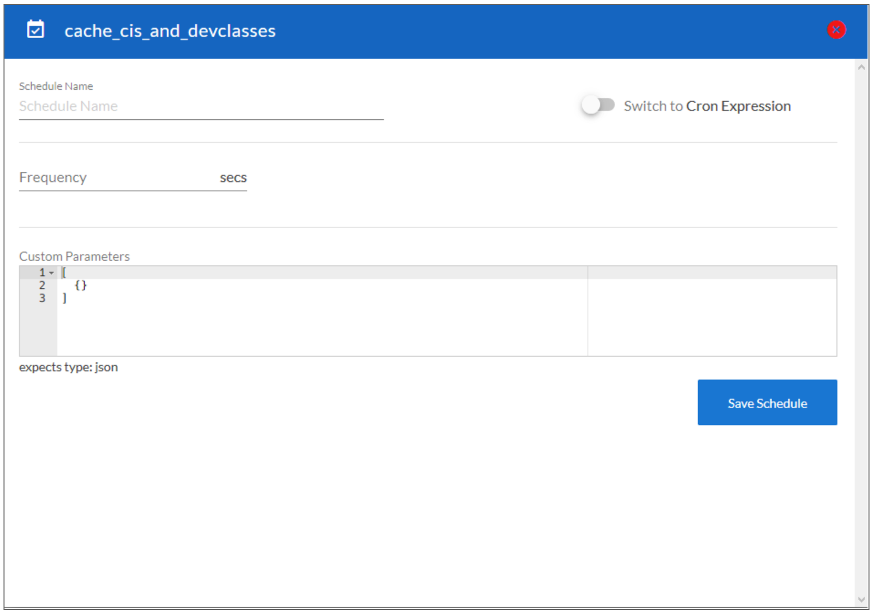
Custom Parameters

```
{}
```

expects type: json

> **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. Select a schedule from the list to view the details for that schedule.
3. Click the + icon to create a schedule. A blank **Schedule** window appears:



```
[
  {}
]
```

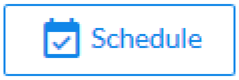
4. In the **Schedule** window, complete the following fields:

   - ***Schedule Name***. Type a name for the schedule.
   - ***Switch to***. Use this toggle to switch between a cron expression and setting the frequency in seconds.

     - ***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.

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

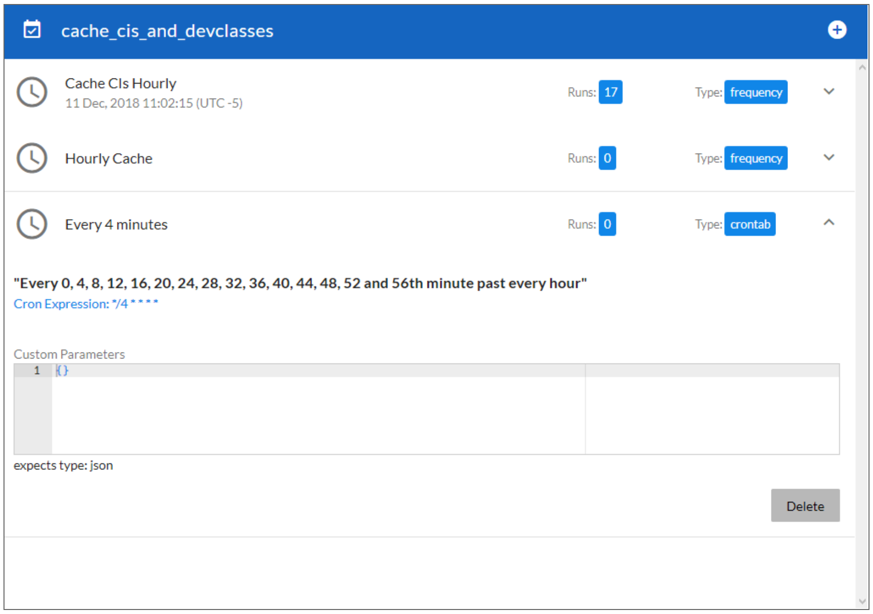


NOTE: After you create a schedule, it continues to run until you delete it. Also, you cannot edit an existing schedule, but you can delete it and create a similar schedule if needed.

To view or delete an existing schedule:

1. On the **Integrations** page , click **[Schedule]** for the integration application that contains a schedule you want to delete. The **Schedule** window appears.

2. Click the down arrow icon (⌄) to view the details of an existing schedule:



cache_cis_and_devclasses

Cache CIs Hourly
11 Dec, 2018 11:02:15 (UTC -5)
Runs: 17 Type: frequency

Hourly Cache
Runs: 0 Type: frequency

Every 4 minutes
Runs: 0 Type: crontab

"Every 0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52 and 56th minute past every hour"
Cron Expression: */4 * * * *

Custom Parameters

```
{}
```

expects type: json

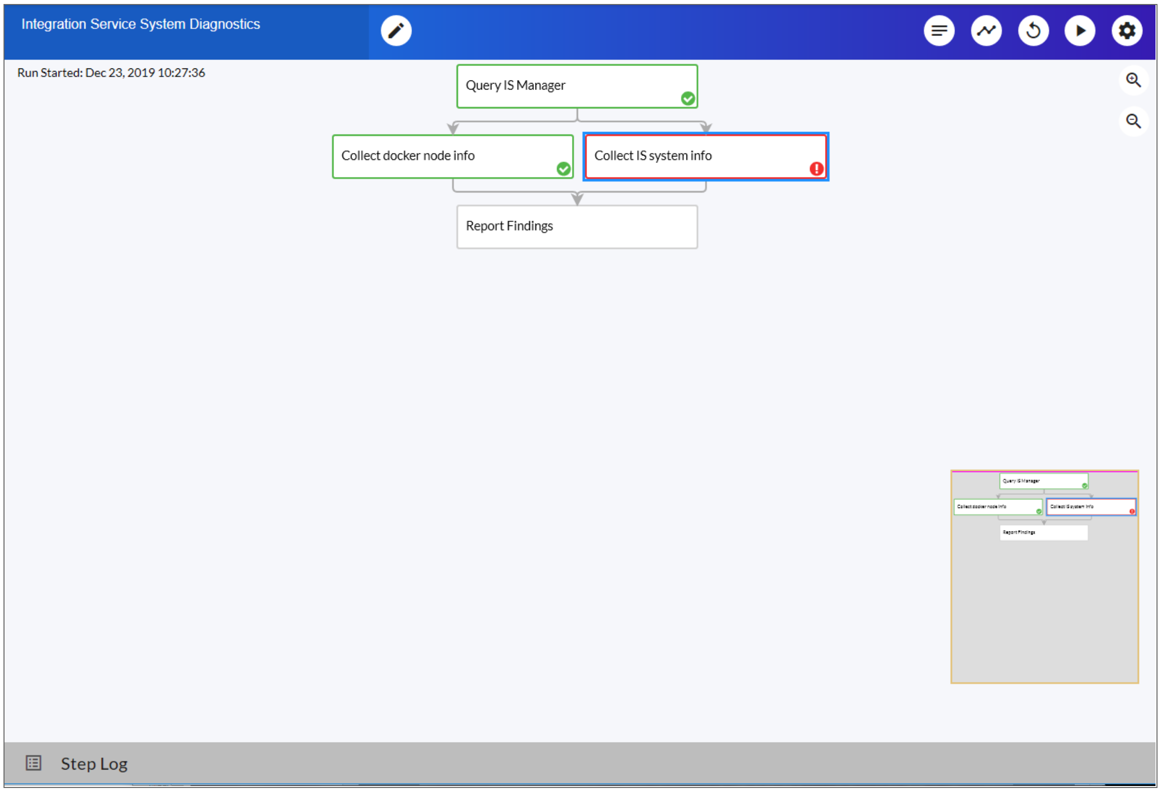
3. To delete the selected schedule, click **[Delete]**. The schedule is removed.

Viewing Integration Service System Diagnostics

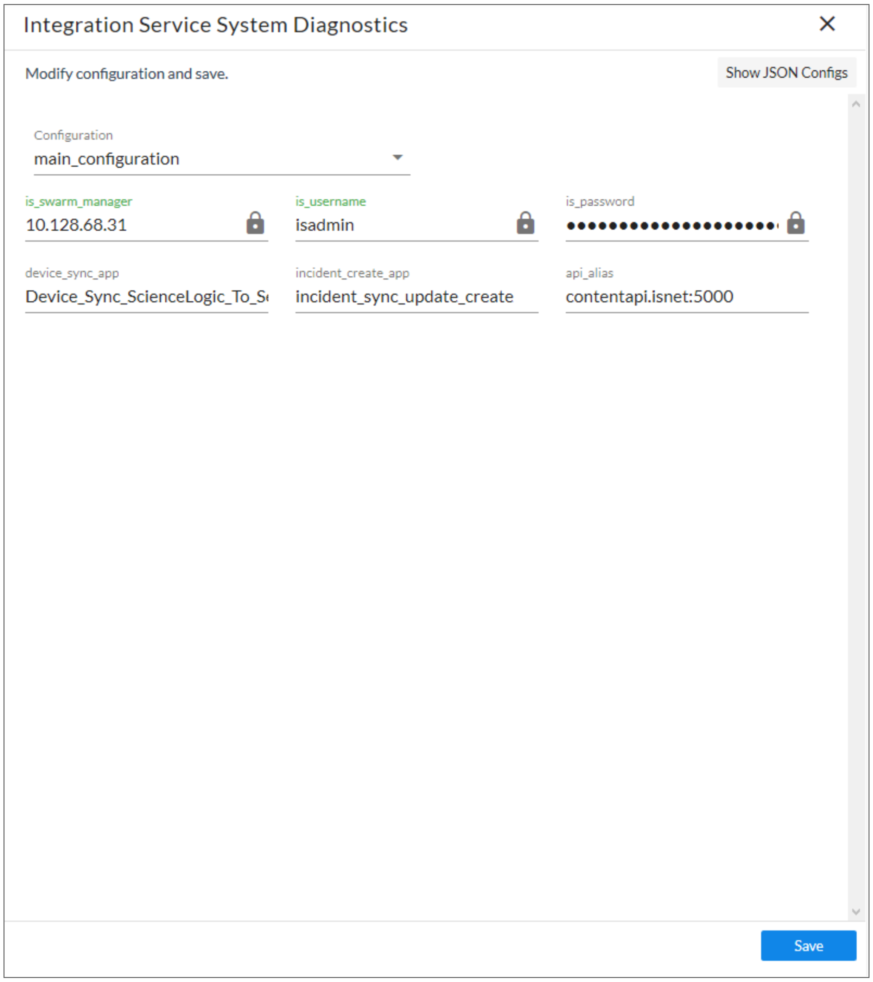
The "Integration Service System Diagnostics" integration application lets you view platform diagnostics for the Integration Service. You can use the information displayed in these diagnostics to help you troubleshoot issues with the different tools used by the Integration Service.

To view the diagnostics report:

1. From the **Integrations** page, select the "Integration Service System Diagnostics" integration application. The **Integration Application** page appears:



2. Click **[Configure]** (⚙). The **Configuration** pane appears:



Modify configuration and save.

Configuration: main_configuration

is_swarm_manager: 10.128.68.31

is_username: isadmin

is_password: ••••••••••••••••••••

device_sync_app: Device_Sync_ScienceLogic_To_S

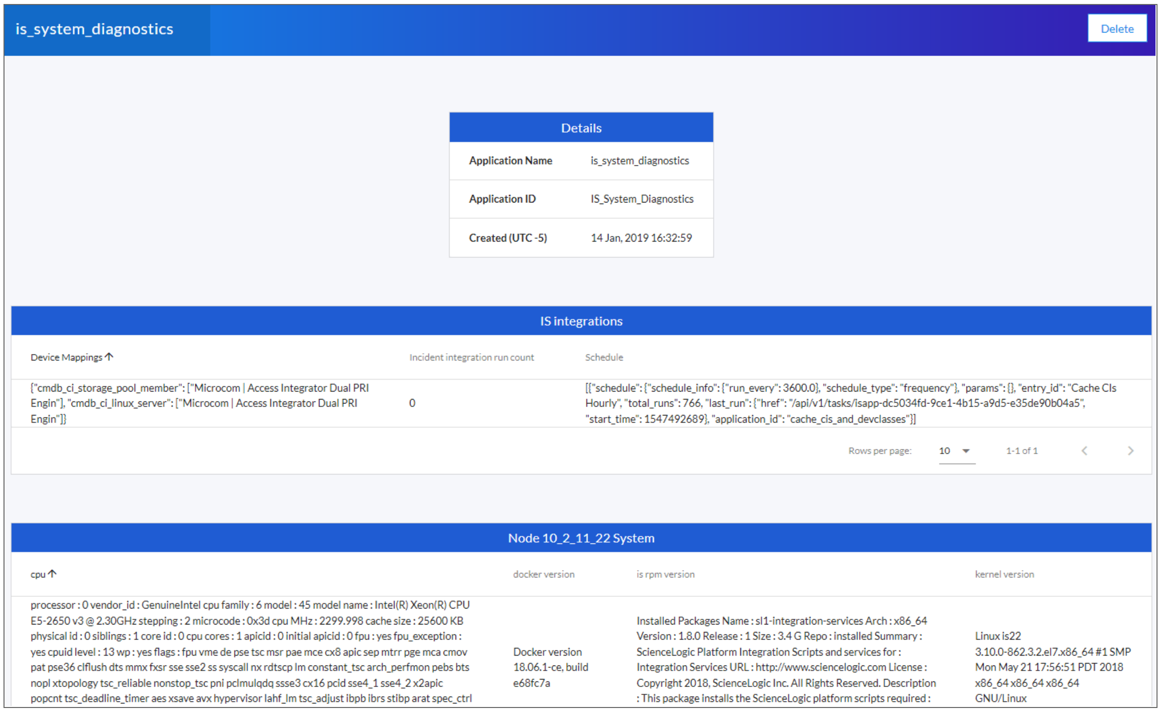
incident_create_app: incident_sync_update_create

api_alias: contentapi.isnet:5000

3. Complete the following fields:
   - ***Configuration***. Select a configuration to align with this integration application. The "IS - System Diagnostic Configuration Example" configuration object contains the structure needed for this integration application, and you can use that configuration object as a template. Be sure to update the configuration object with values for ***is_swarm_manager***, ***is_username***, and ***is_password***.
   - ***device_sync_app***. Specify the name of the Incident Creation integration application that contains the relevant device mappings.
   - ***incident_create_app***. Specify the name of the Device Sync integration application that contains information about the incidents that have been created.
   - **api_alias**. Specify the alias to reference the API internally to make calls.

4. Click **[Save]** and wait for the "App & Config modifications saved" pop-up message to appear. The **Configuration** pane automatically closes after this message appears.

5. On the **Integration Application** page, click **[Run]** (▶). The integration application generates a report that you can access on the **Reports** page (🗎):



Details

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

IS integrations

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 CIs Hourly", "total_runs": 766, "last_run": {"href": "/api/v1/tasks/isapp-dc5034fd-9ce1-4b15-a9d5-e35de90b04a5", "start_time": 1547492689}, "application_id": "cache_cis_and_devclasses"}]

Node 10_2_11_22 System

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 pclmulqdq ssse3 cx16 pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx hypervisor lahf_lm tsc_adjust ibpb 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.

Backing up Data

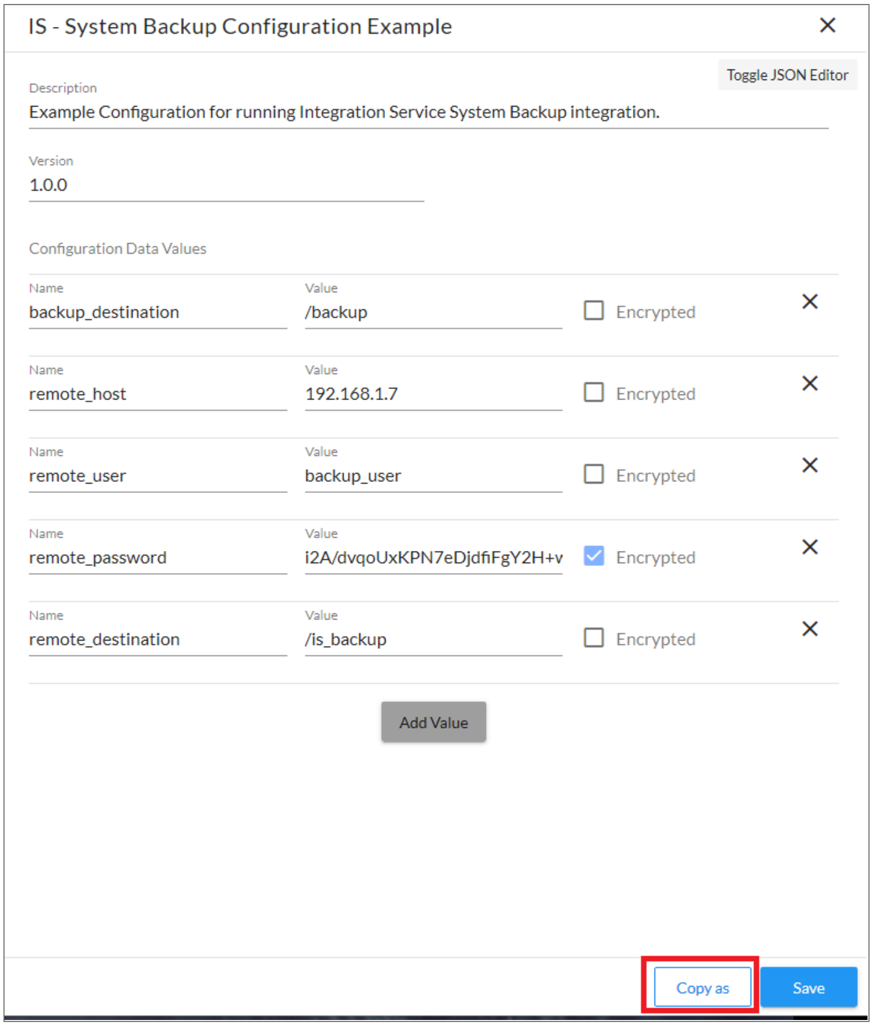
You can use the Integration Service to back up and recover data in the Couchbase database.

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** page (⚙), click the **[Actions]** button (⋮) for the "IS - System Backup Configuration Example", and select *Edit*. The **Configuration** pane for that configuration object appears:



IS - System Backup Configuration Example

Description
Example Configuration for running Integration Service System Backup integration.

Version
1.0.0

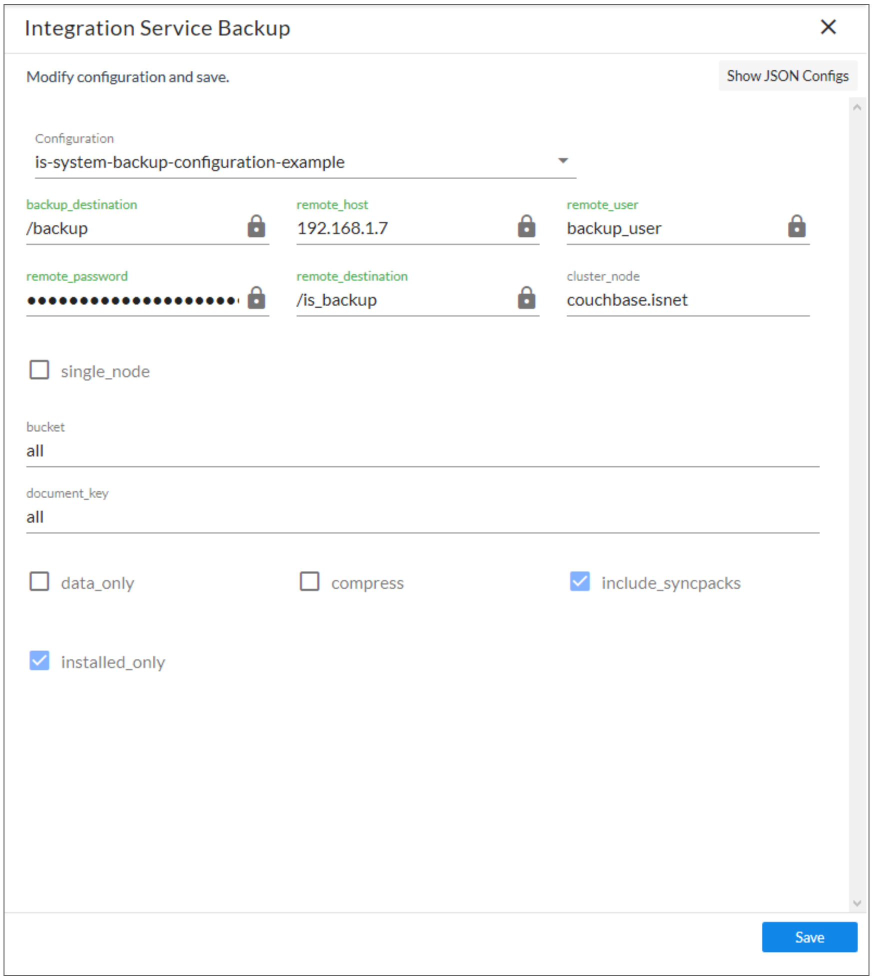
Configuration Data Values

Name	Value		
backup_destination	/backup	☐ Encrypted	✕
remote_host	192.168.1.7	☐ Encrypted	✕
remote_user	backup_user	☐ Encrypted	✕
remote_password	i2A/dvqoUxKPN7eDjdfiFgY2H+v	☑ Encrypted	✕
remote_destination	/is_backup	☐ Encrypted	✕

2. Click **[Copy as]** and 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 **[Save]**.
4. Go to the **Integrations** page and select the "Integration Service Backup" integration application. The **Integration Editor** page appears.
5. Click **[Configure]** (⚙). The **Configuration** pane appears:



Integration Service Backup

Modify configuration and save.

Configuration: is-system-backup-configuration-example

backup_destination: /backup

remote_host: 192.168.1.7

remote_user: backup_user

remote_password: ••••••••••••••••••••

remote_destination: /is_backup

cluster_node: couchbase.isnet

□ single_node

bucket: all

document_key: all

□ data_only □ compress ☑ include_syncpacks

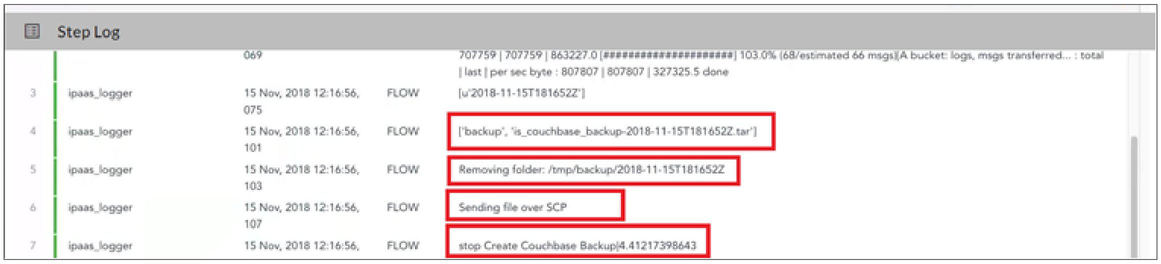
☑ installed_only

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

   - ***Configuration***. Select a configuration object you created in steps 1-3.
   - ***cluster_node***. Specify the node from which you want to make the backup (for a single-node backup only).
   - ***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.
   - ***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.
   - ***data_only***. Select this option if you only want to restore bucket data.
   - ***compress***. Select this option to compress backups using Gzip.
   - ***include_syncpacks***. Select this option if you want to back up the Synchronization PowerPacks on the Integration Service system.
   - ***installed_only***. Select this option if you want to back up only Synchronization PowerPacks that have been installed. If you do not select this option, the integration application will also back up Synchronization PowerPacks that have been uploaded to the Integration Service system.

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 **[Save]** and wait for the "App & Config modifications saved" pop-up message to appear. The **Configuration** pane automatically closes after this message appears.
8. On the **IS Backup Integration Editor** page, click **[Run Now]** (▶). 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, select the "Create IS Backup" step and open the ***Step Log*** section.
10. Look for entries related to backup and make a note of the of the backup file name, which you will need when you run the "Integration Service Restore" integration application:



Step Log

```
                    069                               707759 | 707759 | 863227.0 [####################] 103.0% (68/estimated 66 msgs)[A bucket: logs, msgs transferred... : total
                                                      | last | per sec byte : 807807 | 807807 | 327325.5 done
3   ipaas_logger    15 Nov, 2018 12:16:56,   FLOW     [u'2018-11-15T181652Z']
                    075
4   ipaas_logger    15 Nov, 2018 12:16:56,   FLOW     ['backup', 'is_couchbase_backup-2018-11-15T181652Z.tar']
                    101
5   ipaas_logger    15 Nov, 2018 12:16:56,   FLOW     Removing folder: /tmp/backup/2018-11-15T181652Z
                    103
6   ipaas_logger    15 Nov, 2018 12:16:56,   FLOW     Sending file over SCP
                    107
7   ipaas_logger    15 Nov, 2018 12:16:56,   FLOW     stop Create Couchbase Backup|4.41217398643
```

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** page. For more information, see *Scheduling an Integration Application*.

Restoring a Backup

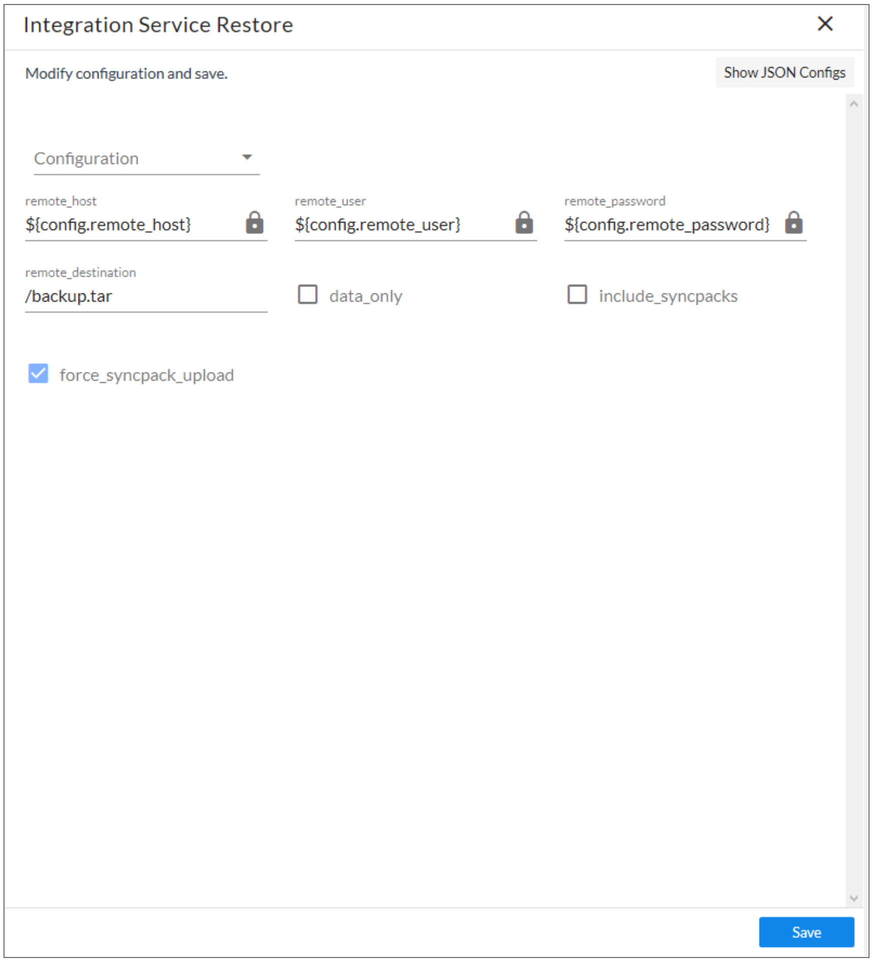
After you have created a backup using the "Integration Service Backup" integration application in the Integration Service user interface, you can use the "Integration Service Restore" integration application to restore that file.

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** page and select the **Integration Service Restore** integration application. The **Integration Service Restore** page appears.

2. Click **[Configure]** (⚙). The **Configuration** pane appears:



Integration Service Restore

Modify configuration and save.

Configuration

remote_host: ${config.remote_host}

remote_user: ${config.remote_user}

remote_password: ${config.remote_password}

remote_destination: /backup.tar

☐ data_only

☐ include_syncpacks

☑ force_syncpack_upload

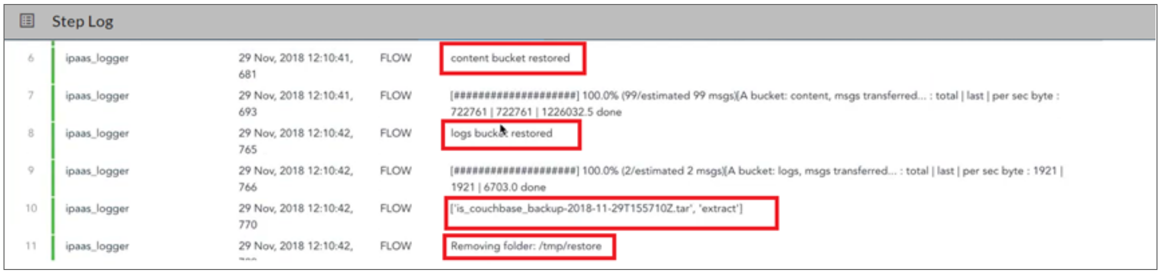
3. In the configuration file, provide values for the following fields:

   - ***Configuration***. Select the same configuration object you aligned with the "Integration Service Backup" integration application. Required.
   - ***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.
   - ***include_syncpacks***. Select this option to restore the Synchronization PowerPacks you backed up with the "Integration Service Backup" integration application.
   - ***force_syncpack_upload***. Select this option if you want to force upload Synchronization PowerPacks if the files already exist in the Integration Service system.

4. Click **[Save]** and wait for the "App & Config modifications saved" pop-up message to appear. The **Configuration** pane automatically closes after this message appears.

5. On the **Integration Service Restore** page, click **[Run Now]** ().

6. 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

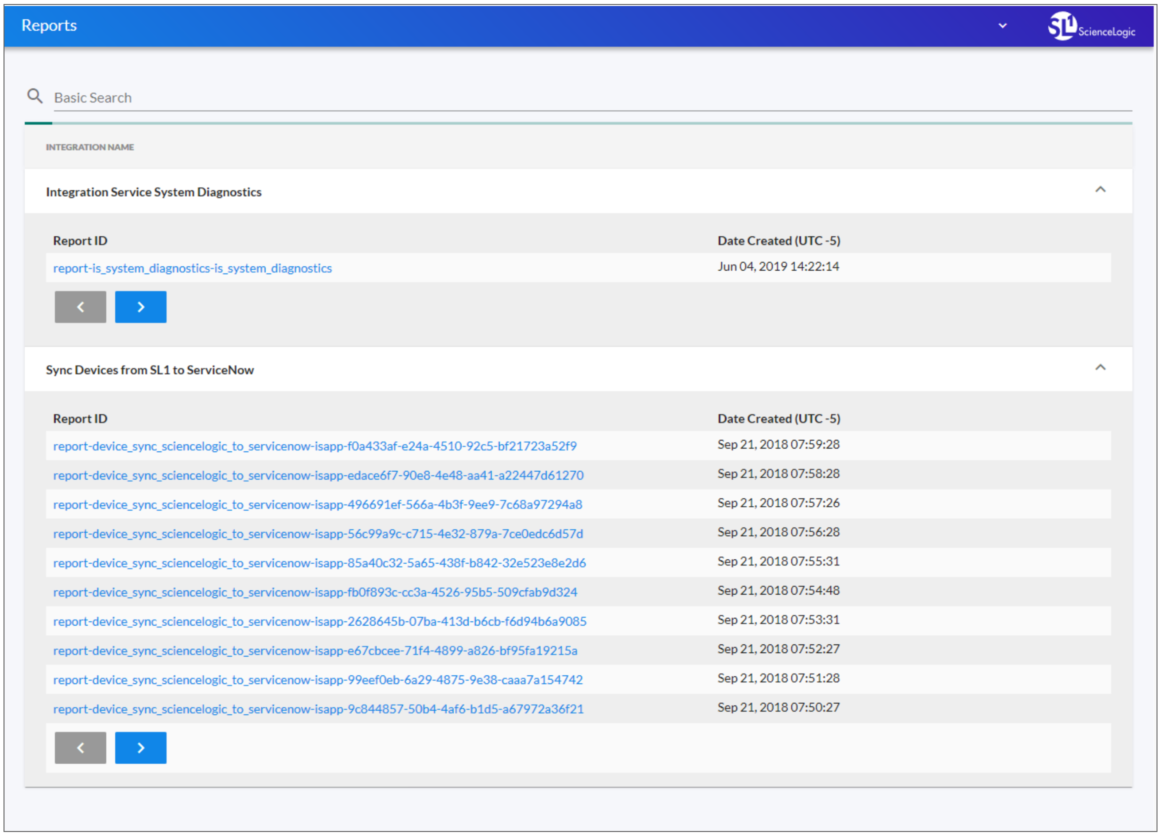
```
6   ipaas_logger   29 Nov, 2018 12:10:41,   FLOW   content bucket restored
                   681
7   ipaas_logger   29 Nov, 2018 12:10:41,   FLOW   [####################] 100.0% (99/estimated 99 msgs)[A bucket: content, msgs transferred... : total | last | per sec byte :
                   693 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

Viewing a Report for an Integration Application

In the Integration Service user interface, the **Reports** page (🗎) 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.



Integration Service System Diagnostics

Report ID	Date Created (UTC -5)
report-is_system_diagnostics-is_system_diagnostics	Jun 04, 2019 14:22:14

Sync Devices from SL1 to ServiceNow

Report ID	Date Created (UTC -5)
report-device_sync_sciencelogic_to_servicenow-isapp-f0a433af-e24a-4510-92c5-bf21723a52f9	Sep 21, 2018 07:59:28
report-device_sync_sciencelogic_to_servicenow-isapp-edace6f7-90e8-4e48-aa41-a22447d61270	Sep 21, 2018 07:58:28
report-device_sync_sciencelogic_to_servicenow-isapp-496691ef-566a-4b3f-9ee9-7c68a97294a8	Sep 21, 2018 07:57:26
report-device_sync_sciencelogic_to_servicenow-isapp-56c99a9c-c715-4e32-879a-7ce0edc6d57d	Sep 21, 2018 07:56:28
report-device_sync_sciencelogic_to_servicenow-isapp-85a40c32-5a65-438f-b842-32e523e8e2d6	Sep 21, 2018 07:55:31
report-device_sync_sciencelogic_to_servicenow-isapp-fb0f893c-cc3a-4526-95b5-509cfab9d324	Sep 21, 2018 07:54:48
report-device_sync_sciencelogic_to_servicenow-isapp-2628645b-07ba-413d-b6cb-f6d94b6a9085	Sep 21, 2018 07:53:31
report-device_sync_sciencelogic_to_servicenow-isapp-e67cbcee-71f4-4899-a826-bf95fa19215a	Sep 21, 2018 07:52:27
report-device_sync_sciencelogic_to_servicenow-isapp-99eef0eb-6a29-4875-9e38-caaa7a154742	Sep 21, 2018 07:51:28
report-device_sync_sciencelogic_to_servicenow-isapp-9c844857-50b4-4af6-b1d5-a67972a36f21	Sep 21, 2018 07:50:27

You can search for a specific report by typing the name of that report in the ***Search*** field at the top of the **Reports** page. The user interface filters the list as you type.

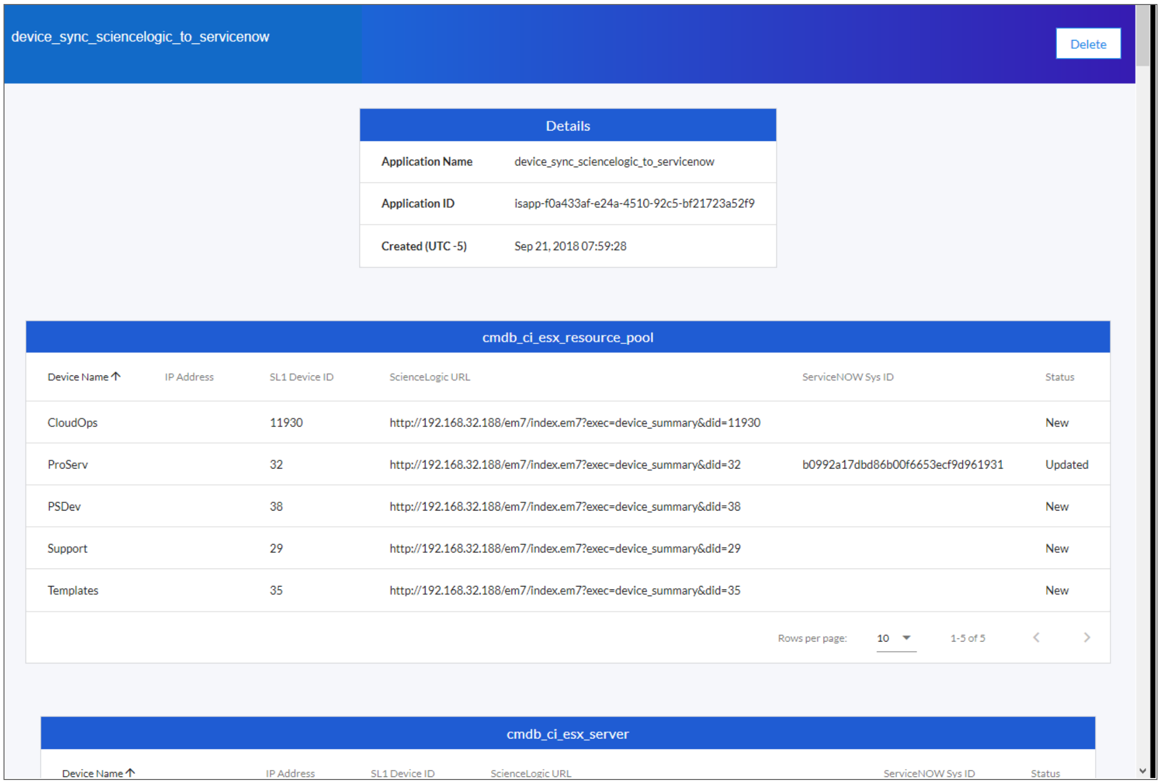
NOTE: Not all integration applications generate reports. Currently, only the "Integration Service System Diagnostics" integration applications support the generation of reports.

To view details for an integration application report:

1. On the **Reports** page ( ), 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:



Details	
Application Name	device_sync_sciencelogic_to_servicenow
Application ID	isapp-f0a433af-e24a-4510-92c5-bf21723a52f9
Created (UTC -5)	Sep 21, 2018 07:59:28

cmdb_ci_esx_resource_pool

Device Name ↑	IP Address	SL1 Device ID	ScienceLogic URL	ServiceNOW Sys ID	Status
CloudOps		11930	http://192.168.32.188/em7/index.em7?exec=device_summary&did=11930		New
ProServ		32	http://192.168.32.188/em7/index.em7?exec=device_summary&did=32	b0992a17dbd86b00f6653ecf9d961931	Updated
PSDev		38	http://192.168.32.188/em7/index.em7?exec=device_summary&did=38		New
Support		29	http://192.168.32.188/em7/index.em7?exec=device_summary&did=29		New
Templates		35	http://192.168.32.188/em7/index.em7?exec=device_summary&did=35		New

cmdb_ci_esx_server

Device Name ↑	IP Address	SL1 Device ID	ScienceLogic URL	ServiceNOW Sys ID	Status

TIP: The ***Status*** field in a report displays the current state of the synced item, which can include *New*, *Removed*, *Updated*, or *Unchanged*.

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

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

4. To delete a report, open the report and click **[Delete]**. Click **[OK]** to delete the report.

Chapter 7

Managing Configuration Objects

Overview

This chapter describes how to use the **Configurations** page (⚙) to create or use a ***configuration object*** that contains a set of variables that all steps and integration applications can use.

This chapter covers the following topics:

What is a Configuration Object?

A ***configuration object*** is a stand-alone JSON file that lives on the Integration Service system. A configuration object supplies the login credentials and other global variables that can be used by all steps and integration applications. Configuration objects allow the same integration application to be deployed in multiple Integration Service instances, with different configurations.

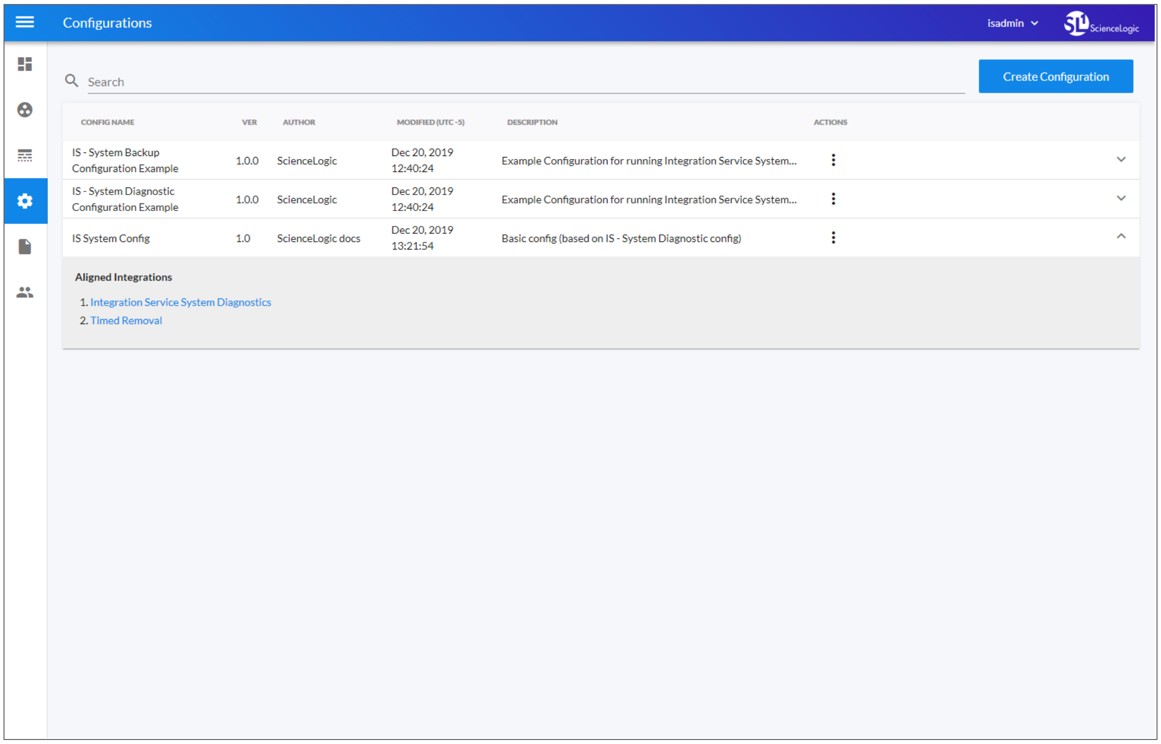
Configuration objects can map variables from SL1 to a third-party platform. For instance, SL1 has device classes and various third-party platforms have CI classes; the configuration object would map these two variables.

You can create and edit configuration objects on the **Configurations** page of the Integration Service user interface. After you create the configuration object, it appears in the ***Configuration*** dropdown on the **Configuration** pane of the **Integrations** page. Before you can run an integration application, you must select a configuration object and "align" that configuration object with the integration application.

TIP: You can select *none* from the ***Configuration*** dropdown to clear or "un-align" the selected configuration object from an integration application. Also, if you did not select a configuration object when editing fields on the **Configuration** pane, the previously set configuration object will remain aligned (if there was a previously set configuration object).

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** page (⚙) displays a list of available configurations. From this page you can create and edit configuration objects:



CONFIG NAME	VER	AUTHOR	MODIFIED (UTC -5)	DESCRIPTION	ACTIONS
IS - System Backup Configuration Example	1.0.0	ScienceLogic	Dec 20, 2019 12:40:24	Example Configuration for running Integration Service System...	⋮
IS - System Diagnostic Configuration Example	1.0.0	ScienceLogic	Dec 20, 2019 12:40:24	Example Configuration for running Integration Service System...	⋮
IS System Config	1.0	ScienceLogic docs	Dec 20, 2019 13:21:54	Basic config (based on IS - System Diagnostic config)	⋮

Aligned Integrations

1. Integration Service System Diagnostics
2. Timed Removal

You can search for a specific configuration object by typing the name of that configuration in the ***Search*** field at the top of the **Configurations** page. The user interface filters the list as you type. Click the **[Actions]** button (⋮) for a configuration object to edit or delete that object.

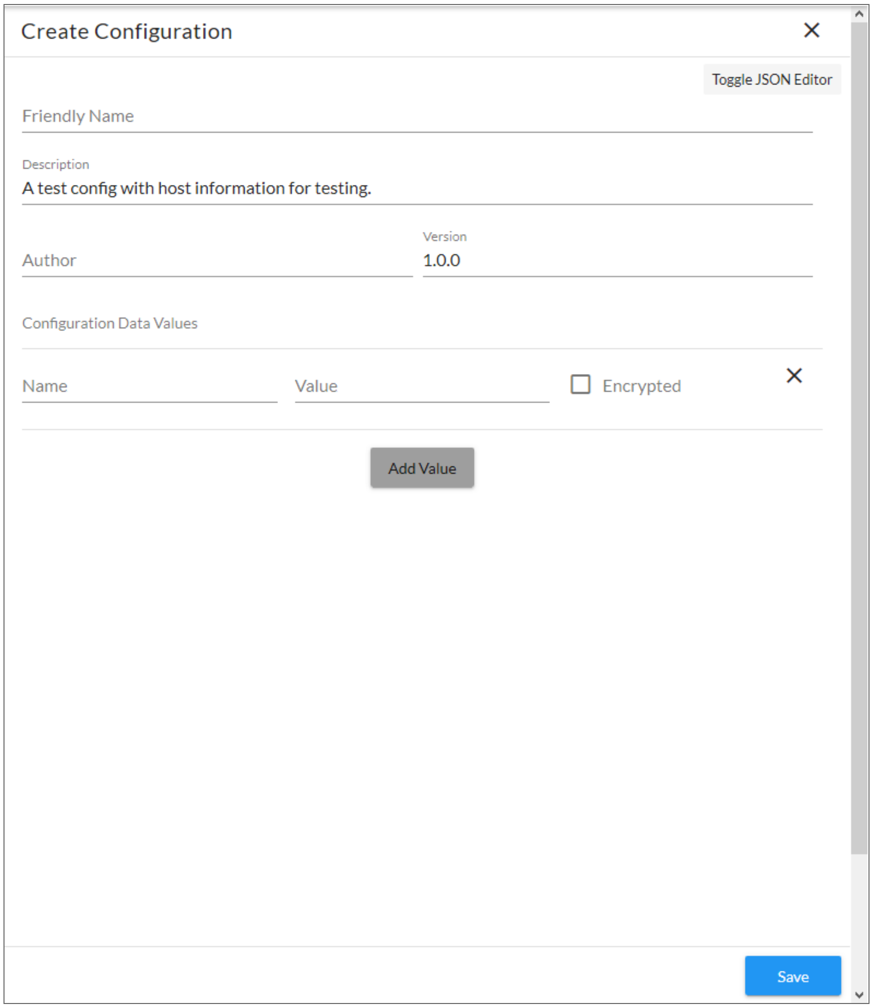
TIP: Click the down arrow icon (⌄) for a configuration object to see which integration applications are currently "aligned" with that configuration object.

Creating a Configuration Object

When creating or editing a configuration object on the **Configurations** page, the name-value pairs in the **Configuration Data** section display in fields by default instead of a block of JSON code. For more complex configuration objects, you can click **[Toggle JSON Editor]** to switch between text fields and JSON code for the configuration data.

To create a new configuration object:

1. From the **Configurations** page (⚙), click **[Create Configuration]**. The **Create Configuration** pane appears:



Description
A test config with host information for testing.

Version
1.0.0

TIP: Instead of creating a complete new configuration object, you can also edit an existing configuration object that has some of the configuration data that you want to use and click **[Copy as]** from the **Configuration** pane to create a copy of that configuration object.

2. Complete the following fields:

   - ***Friendly Name***. Name of the configuration object that will display in the user interface.
   - ***Description***. A brief description of the configuration object.
   - ***Author***. User or organization that created the configuration object.
   - ***Version***. Version of the configuration object.
   - ***Configuration Data Values***. To add configuration values in the form of name-value pairs, click **[Add Value]**. Complete the ***Name*** and ***Value*** fields, and select ***Encrypted*** if needed.

TIP: Click **[Toggle JSON Editor]** to view the JSON configuration data for the configuration object at the bottom of the pane. Click **[Toggle JSON Editor]** again to view the ***Configuration Data Values*** section with the **[Add Value]** button instead.

3. In the **Configuration Data** section, include the required block of code to ensure that the integration applications aligned to this configuration object do not fail:

```
{
  "encrypted": false,
  "name": "sl1_db_host",
  "value": "${config.sl1_host}"
}
```

NOTE: If you are using SL1 with an External Database (SL1 Extended architecture or a cloud-based architecture), update the `"value"` of that block of code to be the host of your database. This field accepts IP addresses. For example: `"value": "db.sciencelogic.com"`. If you are *not* using the SL1 Extended architecture or a cloud-based architecture, you do not need to make any changes to the block of code other than pasting the code into the configuration object.

4. Click **[Toggle JSON Editor]** to view the JSON configuration data. In the ***Configuration Data*** field, update the default variable definitions to match your Integration Service configuration. For example, you could add the following JSON code to the ***Configuration Data*** field:

```
[
  {
    "encrypted": false,
    "name": "sl1_db_host",
    "value": "10.2.11.42"
  },
  {
    "encrypted": false,
    "name": "em7_user",
    "value": "em7admin"
  },
  {
    "encrypted": true,
    "name": "em7_password",
    "value": "+dqGJe1NwTyvdaO2EizTWjJ2uj2C1wzBzgNqVhpdTHA="
  }
]
```

5. When creating a new 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. Click **[Save]**, or click **[Copy as]** to save the configuration object as a new configuration object with a different name.
8. Align this configuration object with the integration applications that you want to run by clicking the **Configure** button from the detail page for each integration application and selecting this configuration object from the ***Configuration*** drop-down.

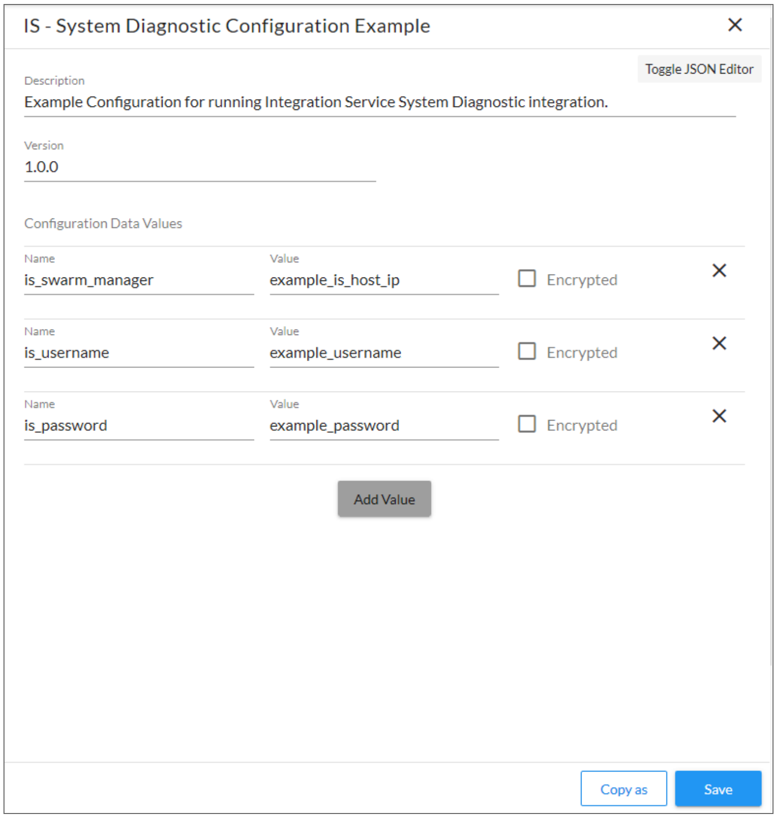
> **NOTE**: In a step, you can include the **config.** prefix with a variable to tell the Integration Service system to look in a configuration object to resolve the variable.

Editing a Configuration Object

To edit an existing configuration object:

1. Go to the **Configurations** page.

2. Click the **[Actions]** button (⋮) for the configuration object you want to edit and select *Edit*. The **Configuration** pane appears:



IS - System Diagnostic Configuration Example

Description
Example Configuration for running Integration Service System Diagnostic integration.

Version
1.0.0

Configuration Data Values

Name	Value		
is_swarm_manager	example_is_host_ip	□ Encrypted	✕
is_username	example_username	□ Encrypted	✕
is_password	example_password	□ Encrypted	✕

3. Edit the values in the following fields as needed:

   - ***Description***. A brief description of the configuration object.
   - ***Version***. Version of the configuration object.
   - ***Configuration Data Values***. To add configuration values in the form of name-value pairs, click **[Add Value]**. Complete the ***Name*** and ***Value*** fields, and select ***Encrypted*** if needed.

TIP: Click **[Toggle JSON Editor]** to view the JSON configuration data for the configuration object at the bottom of the pane. Click **[Toggle JSON Editor]** again to view the ***Configuration Data Values*** section with the **[Add Value]** button instead.

4. To make a copy of this configuration object, click **[Copy As]** and update the relevant fields in the **Create Configuration** pane.

5. Click **[Save]** to save your changes.

Chapter 8

Viewing Logs in the Integration Service

Overview

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

This chapter covers the following topics:

Logging Data in the Integration Service

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

Local Logging

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

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

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

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
```

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.

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

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/iservices** 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.

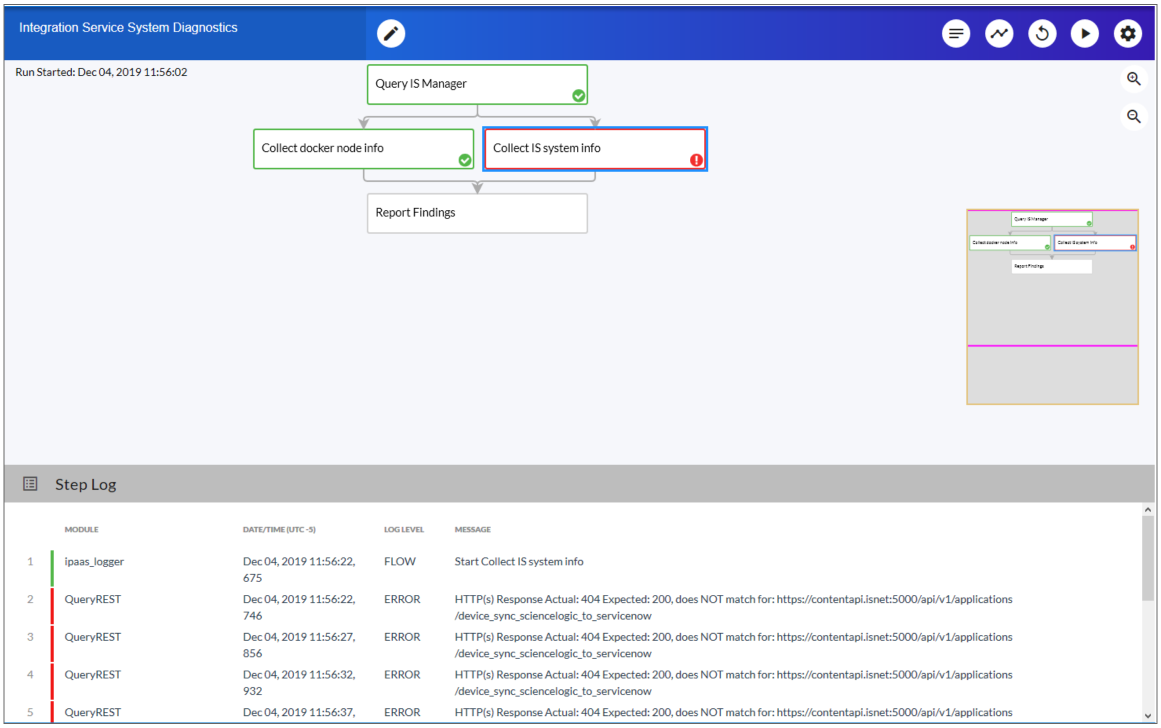
In the example message, the "Query and Cache ServiceNow CIs" step took around 41 seconds to run.

Viewing the Step Logs for an Integration Application

The **Step Log** pane on the **Integration Application** page displays the time, the type of log, and the log messages from a step that you selected in the main pane. All times that are displayed in the **Message** column of this pane are in seconds, such as "stop Query and Cache ServiceNow CI List|***5.644190788269043***".

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.
3. 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 log messages for the selected step:



Run Started: Dec 04, 2019 11:56:02

	MODULE	DATE/TIME (UTC -5)	LOG LEVEL	MESSAGE
1	ipaas_logger	Dec 04, 2019 11:56:22, 675	FLOW	Start Collect IS system info
2	QueryREST	Dec 04, 2019 11:56:22, 746	ERROR	HTTP(s) Response Actual: 404 Expected: 200, does NOT match for: https://contentapi.isnet:5000/api/v1/applications /device_sync_sciencelogic_to_servicenow
3	QueryREST	Dec 04, 2019 11:56:27, 856	ERROR	HTTP(s) Response Actual: 404 Expected: 200, does NOT match for: https://contentapi.isnet:5000/api/v1/applications /device_sync_sciencelogic_to_servicenow
4	QueryREST	Dec 04, 2019 11:56:32, 932	ERROR	HTTP(s) Response Actual: 404 Expected: 200, does NOT match for: https://contentapi.isnet:5000/api/v1/applications /device_sync_sciencelogic_to_servicenow
5	QueryREST	Dec 04, 2019 11:56:37,	ERROR	HTTP(s) Response Actual: 404 Expected: 200, does NOT match for: https://contentapi.isnet:5000/api/v1/applications

> **TIP:** For longer log messages, click the down arrow icon (⌄) in the **Message** column of the **Step Log** pane to open the message. Also, you can triple-click a log message to highlight the entire text block so you can easily copy the log message to another text viewer.

4. Click the gray area of the **Logs** pane to close the pane.
5. To generate more detailed logs when you run this integration application, hover over **[Run]** (▶) and select *Debug Run*.

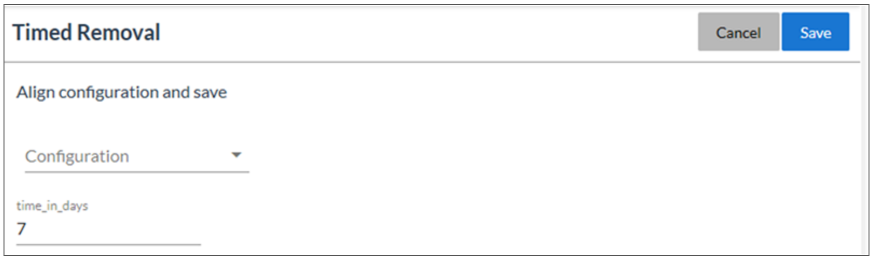
TIP: Log information for a step is saved for the duration of the ***result_expires*** setting in the Integration Service system. The ***result_expires*** setting is defined in the **opt/iservices/scripts/docker-compose.yml** file. The default value for log expiration is 7 days. This environment variable 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:

1. 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:



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.

Chapter 9

Using SL1 to Monitor the Integration Service

Overview

This chapter describes the various ScienceLogic PowerPacks that you can use to monitor the components of the Integration Service. This chapter also describes the suggested settings, metrics, and situations for healthy SL1 and Integration Service system systems.

Use the following menu options to navigate the SL1 user interface:

- To view a pop-out list of menu options, click the menu icon (☰).
- To view a page containing all of the menu options, click the Advanced menu icon (•••).

This chapter covers the following topics:

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.

TIP: You can also use the **Dashboard** page (▦) in the Integration Service user interface to monitor the status of the various tasks, workers, and integration applications that are running on your Integration Service system. You can use this information to quickly determine if your Integration Service instance is performing as expected.

You can download the following PowerPacks from the ScienceLogic Customer Portal to help you monitor your Integration Service system:

- **Linux Base Pack PowerPack**: This PowerPack monitors your Linux-based Integration Service server with SSH (the Integration Service ISO is built on top of an Oracle Linux Operating System). This PowerPack provides key performance indicators about how your Integration Service server is performing. The only configuration you need to do with this PowerPack is to install the latest version of it.
- **Docker PowerPack**: This PowerPack monitors the various Docker containers, services, and Swarm that manage 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. For more information, see ***Configuring the Docker PowerPack***.
- **ScienceLogic: Integration Service PowerPack**. This PowerPack monitors the status of the integration applications in your Integration Service system. Based on the events generated by this PowerPack, you can diagnose why applications failed on the Integration Service. For more information, see ***Configuring the Integration Service PowerPack***.
- **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. For more information, see ***Configuring the Couchbase PowerPack***.
- **AMQP: 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. For more information, see ***Configuring the RabbitMQ PowerPack***.

You can use each of the PowerPacks listed above to monitor different aspects of the Integration Service. Be sure to download and install the latest version of each PowerPack.

The following sub-topics describe the configuration steps you need to take for each PowerPack. For best results, complete these configuration steps in the given order to set up monitoring of the Integration Service within SL1.

Configuring the Docker PowerPack

The Docker PowerPack monitors the various Docker containers, services, and Swarm that manage 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.

To configure the Docker PowerPack to monitor the Integration Service:

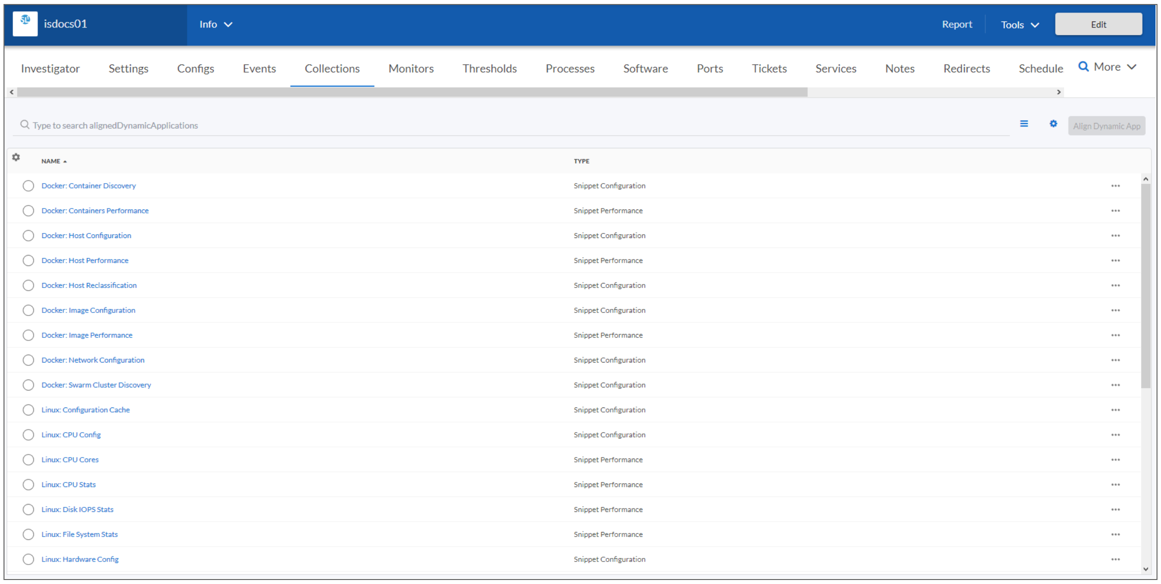
1. Make sure that you have already installed the Linux Base Pack PowerPack.
2. In SL1, go to the **Credential Management** page (System > Manage > Credentials) and click the wrench icon (🔧) for the example **Docker Basic - Dev ssh** credential. The **Credential Editor** modal page appears.
3. Complete the following fields, and keep the other fields at their default settings:
   - ***Credential Name***. Type a new name for the credential.
   - ***Hostname/IP***. Type the hostname or IP address for the Integration Service instance, or type "%D".
   - ***Username***. Type the username for the Integration Service instance.
   - ***Password***. Type the password for the Integration Service instance.
4. Click **[Save As]** and close the **Credential Editor** modal page.
5. On the **Devices** page, click **[Add Devices]** to discover your Integration Service server using the new Docker SSH new credential.

TIP: Use the ***Unguided Network Discovery*** option and search for the new Docker credential on the **Choose credentials** page of the Discovery wizard. Also, be sure to select ***Discover Non-SNMP*** and ***Model Devices*** in the **Advanced options** section. For more information, see the ***Discovery and Credentials***manual.

After the discovery is complete, the Docker and Linux Dynamic Applications will automatically align to your discovered Docker Host for your Integration Service. SL1 creates a new Device record for the Integration Service server and new Device Component records for Docker containers.

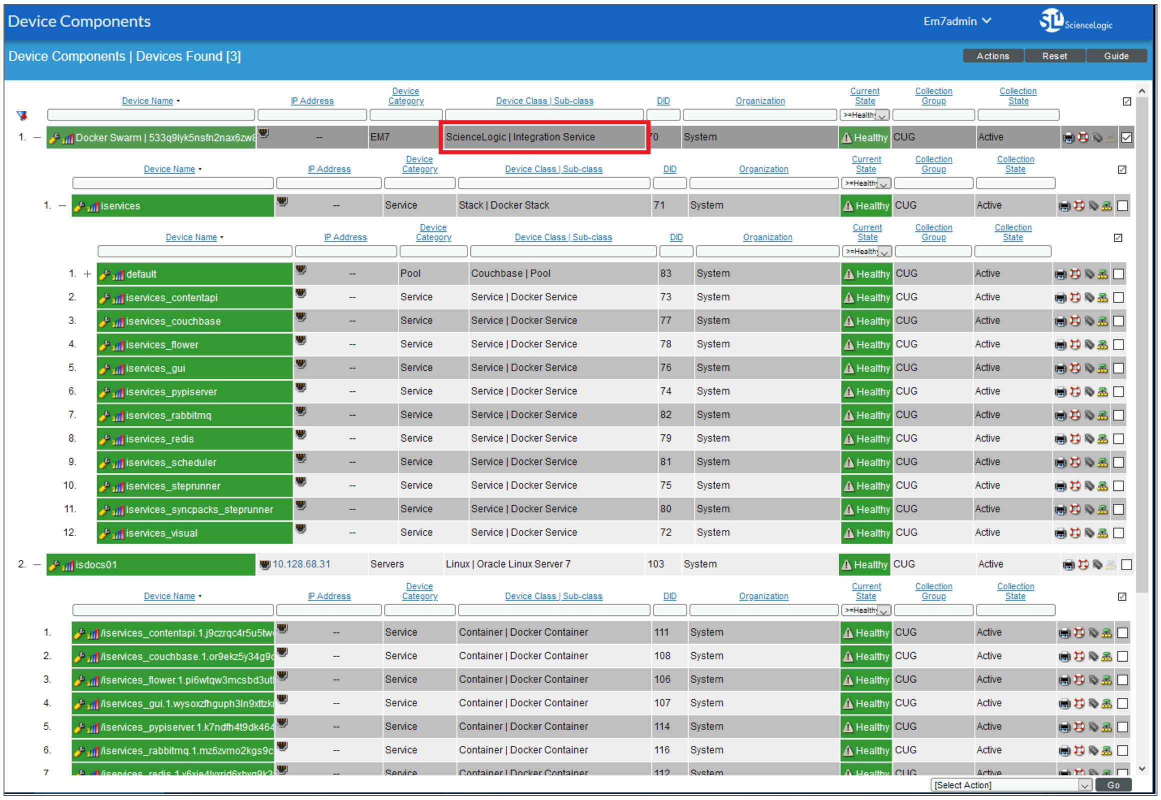
6. Go to the **Devices** page and select the new device representing your Integration Service server.

7. Go to the **[Collections]** tab of the **Device Investigator** page for the new device and make sure that all of the Docker and Linux Dynamic Applications have automatically aligned. This process usually takes a few minutes. A group of Docker and Linux Dynamic Applications should now appear on the **[Collections]** tab:



NAME	TYPE
○ Docker: Container Discovery	Snippet Configuration
○ Docker: Containers Performance	Snippet Performance
○ Docker: Host Configuration	Snippet Configuration
○ Docker: Host Performance	Snippet Performance
○ Docker: Host Reclassification	Snippet Configuration
○ Docker: Image Configuration	Snippet Configuration
○ Docker: Image Performance	Snippet Performance
○ Docker: Network Configuration	Snippet Configuration
○ Docker: Swarm Cluster Discovery	Snippet Configuration
○ Linux: Configuration Cache	Snippet Configuration
○ Linux: CPU Config	Snippet Configuration
○ Linux: CPU Cores	Snippet Performance
○ Linux: CPU Stats	Snippet Performance
○ Linux: Disk IOPS Stats	Snippet Performance
○ Linux: File System Stats	Snippet Performance
○ Linux: Hardware Config	Snippet Configuration

8. To view your newly discovered device components, navigate to the **Device Components** page (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 system on the **Device Components** page:



	Device Name	IP Address	Device Category	Device Class | Sub-class	DID	Organization	Current State	Collection Group	Collection State
1.	Docker Swarm | 533q9lyk5nsfn2nax6zw8	--	EM7	ScienceLogic | Integration Service	70	System	Healthy	CUG	Active

	Device Name	IP Address	Device Category	Device Class | Sub-class	DID	Organization	Current State	Collection Group	Collection State
1.	iservices	--	Service	Stack | Docker Stack	71	System	Healthy	CUG	Active

	Device Name	IP Address	Device Category	Device Class | Sub-class	DID	Organization	Current State	Collection Group	Collection State
1.	default	--	Pool	Couchbase | Pool	83	System	Healthy	CUG	Active
2.	iservices_contentapi	--	Service	Service | Docker Service	73	System	Healthy	CUG	Active
3.	iservices_couchbase	--	Service	Service | Docker Service	77	System	Healthy	CUG	Active
4.	iservices_flower	--	Service	Service | Docker Service	78	System	Healthy	CUG	Active
5.	iservices_gui	--	Service	Service | Docker Service	76	System	Healthy	CUG	Active
6.	iservices_pypiserver	--	Service	Service | Docker Service	74	System	Healthy	CUG	Active
7.	iservices_rabbitmq	--	Service	Service | Docker Service	82	System	Healthy	CUG	Active
8.	iservices_redis	--	Service	Service | Docker Service	79	System	Healthy	CUG	Active
9.	iservices_scheduler	--	Service	Service | Docker Service	81	System	Healthy	CUG	Active
10.	iservices_steprunner	--	Service	Service | Docker Service	75	System	Healthy	CUG	Active
11.	iservices_syncpacks_steprunner	--	Service	Service | Docker Service	80	System	Healthy	CUG	Active
12.	iservices_visual	--	Service	Service | Docker Service	72	System	Healthy	CUG	Active

	Device Name	IP Address	Device Category	Device Class | Sub-class	DID	Organization	Current State	Collection Group	Collection State
2.	isdocs01	10.128.68.31	Servers	Linux | Oracle Linux Server 7	103	System	Healthy	CUG	Active

	Device Name	IP Address	Device Category	Device Class | Sub-class	DID	Organization	Current State	Collection Group	Collection State
1.	/iservices_contentapi.1.j9czrqc4r5u5tw	--	Service	Container | Docker Container	111	System	Healthy	CUG	Active
2.	/iservices_couchbase.1.or9ekz5y34g9c	--	Service	Container | Docker Container	108	System	Healthy	CUG	Active
3.	/iservices_flower.1.pi6wtqw3mcsbd3utl	--	Service	Container | Docker Container	106	System	Healthy	CUG	Active
4.	/iservices_gui.1.wysoxzfhguph3ln9xttzk	--	Service	Container | Docker Container	107	System	Healthy	CUG	Active
5.	/iservices_pypiserver.1.k7ndfh4t9dk464	--	Service	Container | Docker Container	114	System	Healthy	CUG	Active
6.	/iservices_rabbitmq.1.mz6zvmo2kgs9c	--	Service	Container | Docker Container	116	System	Healthy	CUG	Active
7	/iservices_redis.1.v6xie4lvxrid6xbvg9k3	--	Service	Container | Docker Container	112	System	Healthy	CUG	Active

NOTE: If the Docker Swarm root device is modeled with a different device class, click the wrench icon (🔧) for the Docker Swarm root device, click the **[Actions]** button on the **Device Properties** window and select *Device Class*. From the Device Class window, select *ScienceLogic | Integration Service* as the Device Class and click **[Apply]**. Save your changes.

Configuring the Integration Service PowerPack

The *ScienceLogic: Integration Service* PowerPack monitors the status of the integration applications in your Integration Service system. Based on the events generated by this PowerPack, you can diagnose why applications failed on 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.

In addition, before you can run the Dynamic Applications in the *ScienceLogic: Integration Service* PowerPack, you must manually align the Dynamic Application from this PowerPack to your Integration Service device in SL1. These steps are covered in detail below.

Configuring the PowerPack

To configure the Integration Service PowerPack:

1. In SL1, make sure that you have already installed the Linux Base PowerPack, the Docker PowerPack, and the 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. The **Credential Editor** modal page appears.
3. Complete the following fields, and keep the other fields at their default settings:

    - ***Profile Name***. Type a name for the credential.
    - ***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".

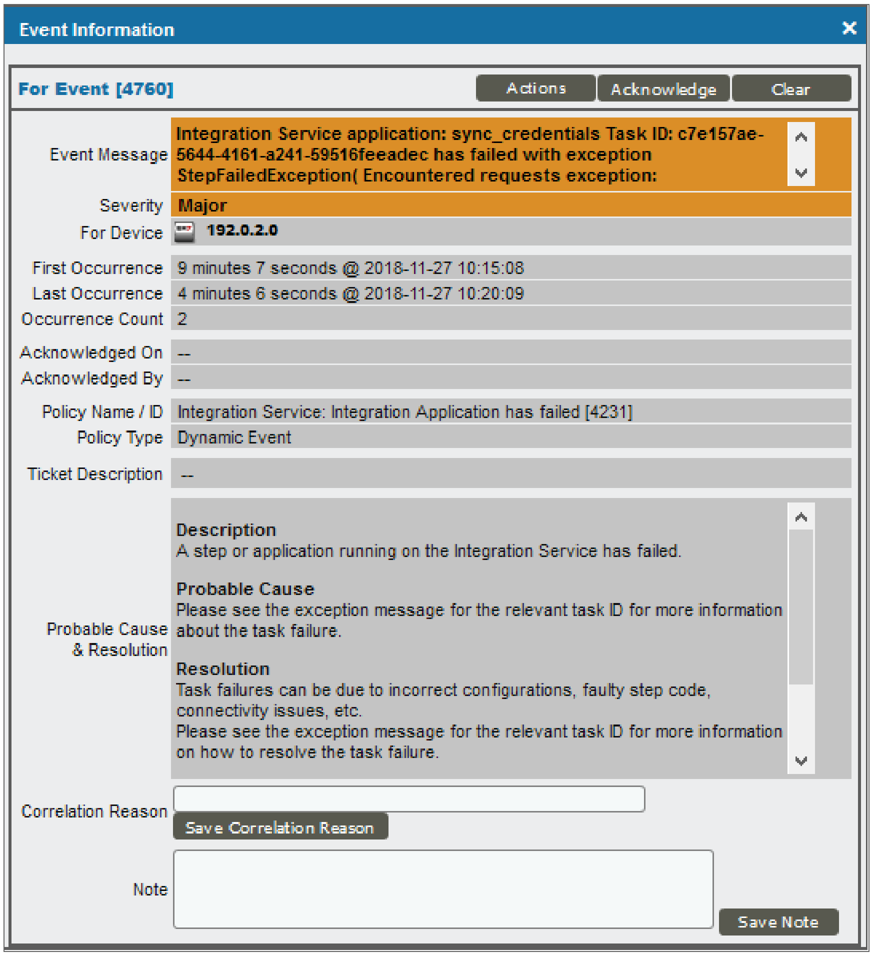
4. Click the **[Save As]** button and close the **Credential Editor** modal page. You will use this new credential to manually align the following Dynamic Applications:

    - REST: Performance Metrics Monitor
    - REST: Performance Metrics Monitor (Couchbase)
    - REST: Performance Metrics Monitor (Integration Service)
    - REST: Performance Metrics Monitor (ServiceNow)

5. Go to the **Devices** page, select the device representing your Integration Service server, and click the **[Collections]** tab.
6. Click **[Edit]**, click **[Align Dynamic App]**, and select *Choose Dynamic Application*. The **Choose Dynamic Application** window appears.
7. In the ***Search*** field, type the name of the first of the Integration Service Dynamic Applications. Select the Dynamic Application and click **[Select]**.
8. Select *Choose Dynamic Application*. The **Choose Credential** window appears.
9. In the ***Search*** field, type the name of the credential you created in steps 2-4, select the new credential, and click **[Select]**. The **Align Dynamic Application** window appears.
10. Click **[Align Dynamic App]**. The Dynamic Application is added to the **[Collections]** tab.
11. Repeat steps 6-10 for each remaining Dynamic Application for this PowerPack, and click **[Save]** when you are done aligning Dynamic Applications.

Events Generated by the PowerPack

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



For Event [4760]

Event Message	Integration Service application: sync_credentials Task ID: c7e157ae-5644-4161-a241-59516feeadec has failed with exception StepFailedException(Encountered requests exception:
Severity	Major
For Device	192.0.2.0
First Occurrence	9 minutes 7 seconds @ 2018-11-27 10:15:08
Last Occurrence	4 minutes 6 seconds @ 2018-11-27 10:20:09
Occurrence Count	2
Acknowledged On	--
Acknowledged By	--
Policy Name / ID	Integration Service: Integration Application has failed [4231]
Policy Type	Dynamic Event
Ticket Description	--
Probable Cause & Resolution	**Description** A step or application running on the Integration Service has failed. **Probable Cause** Please see the exception message for the relevant task ID for more information about the task failure. **Resolution** Task failures can be due to incorrect configurations, faulty step code, connectivity issues, etc. Please see the exception message for the relevant task ID for more information on how to resolve the task failure.
Correlation Reason	
Note	

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-a241-59516feeadec
```

Configuring the Couchbase PowerPack

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.

After you install the Couchbase PowerPack in SL1, create a new Couchbase SOAP/XML credential. Using that credential, you need to manually align the "Couchbase Component Count" and "Couchbase Pool Discovery" Dynamic Applications with the Docker Swarm root device. These steps are covered in detail below.

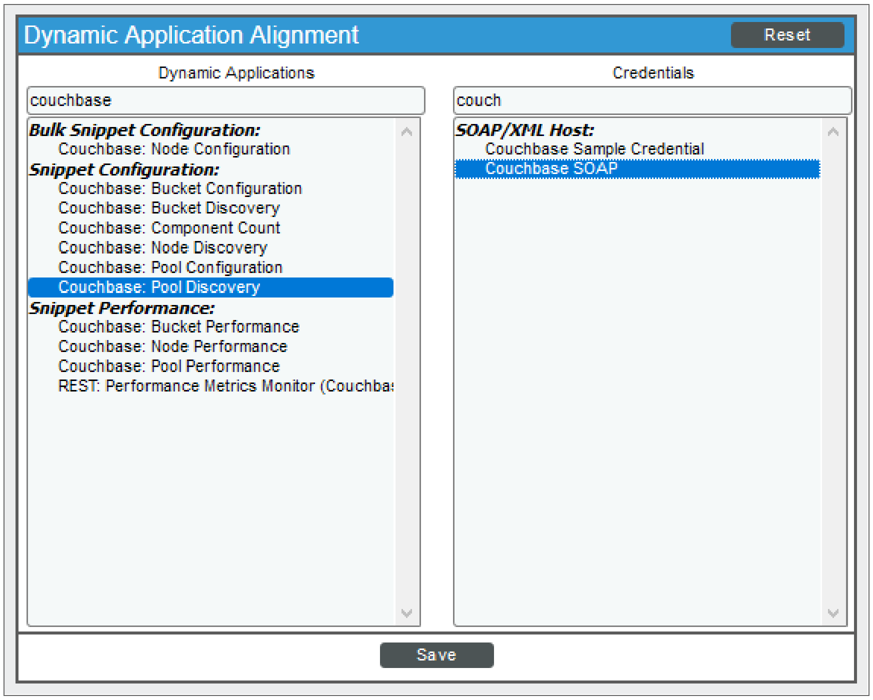
To configure the Couchbase PowerPack:

1. In SL1, navigate to the **Credential Management** page (System > Manage > Credentials) and click the wrench icon (🔧) for the **Couchbase Sample** credential. The **Credential Editor** modal page appears.
2. Complete the following fields, and keep the other fields at their default settings:
    - ***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 Integration Service 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.

3. Click **[Save As]** and close the **Credential Editor** modal page.
4. Go to the **Device Components** page (Devices > Device Components) and expand the Docker Swarm root device by clicking the + icon.
5. Click the wrench icon (🔧) for the “Stack | Docker Stack” component device (iservices) and click the **[Collections]** tab.

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



Dynamic Application Alignment

Reset

Dynamic Applications

couchbase

Bulk Snippet Configuration:
Couchbase: Node Configuration
Snippet Configuration:
Couchbase: Bucket Configuration
Couchbase: Bucket Discovery
Couchbase: Component Count
Couchbase: Node Discovery
Couchbase: Pool Configuration
Couchbase: Pool Discovery
Snippet Performance:
Couchbase: Bucket Performance
Couchbase: Node Performance
Couchbase: Pool Performance
REST: Performance Metrics Monitor (Couchba:

Credentials

couch

SOAP/XML Host:
Couchbase Sample Credential
Couchbase SOAP

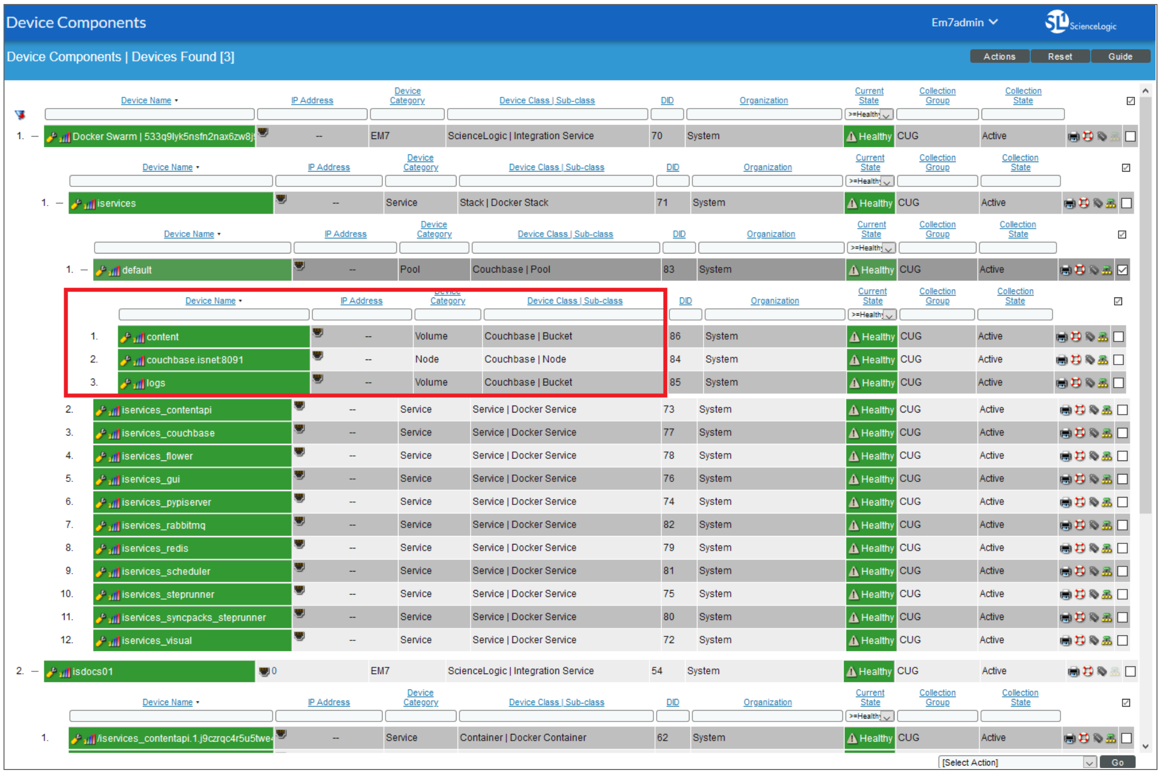
Save

7. Select the "Couchbase: Pool Discovery" Dynamic Application and select the Couchbase credential that you created in steps 1-3. Click **[Save]**.

8. Click the **[Actions]** button, select *Add Dynamic Application* and align the "Couchbase: Component Count" Dynamic Application and the Couchbase credential. Click **[Save]**.

9. 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.

10. Navigate to the **Device Components** page (Devices > Device Components) to see the Couchbase components:



Device Components

Device Components | Devices Found [3]

	Device Name	IP Address	Device Category	Device Class | Sub-class	DID	Organization	Current State	Collection Group	Collection State
1.	Docker Swarm | 533q9lyk5nsfn2nax6zw8j	--	EM7	ScienceLogic | Integration Service	70	System	Healthy	CUG	Active
1.	iservices	--	Service	Stack | Docker Stack	71	System	Healthy	CUG	Active
1.	default	--	Pool	Couchbase | Pool	83	System	Healthy	CUG	Active
1.	content	--	Volume	Couchbase | Bucket	86	System	Healthy	CUG	Active
2.	couchbase.isnet:8091	--	Node	Couchbase | Node	84	System	Healthy	CUG	Active
3.	logs	--	Volume	Couchbase | Bucket	85	System	Healthy	CUG	Active
2.	iservices_contentapi	--	Service	Service | Docker Service	73	System	Healthy	CUG	Active
3.	iservices_couchbase	--	Service	Service | Docker Service	77	System	Healthy	CUG	Active
4.	iservices_flower	--	Service	Service | Docker Service	78	System	Healthy	CUG	Active
5.	iservices_gui	--	Service	Service | Docker Service	76	System	Healthy	CUG	Active
6.	iservices_pypiserver	--	Service	Service | Docker Service	74	System	Healthy	CUG	Active
7.	iservices_rabbitmq	--	Service	Service | Docker Service	82	System	Healthy	CUG	Active
8.	iservices_redis	--	Service	Service | Docker Service	79	System	Healthy	CUG	Active
9.	iservices_scheduler	--	Service	Service | Docker Service	81	System	Healthy	CUG	Active
10.	iservices_steprunner	--	Service	Service | Docker Service	75	System	Healthy	CUG	Active
11.	iservices_syncpacks_steprunner	--	Service	Service | Docker Service	80	System	Healthy	CUG	Active
12.	iservices_visual	--	Service	Service | Docker Service	72	System	Healthy	CUG	Active
2.	isdocs01	0	EM7	ScienceLogic | Integration Service	54	System	Healthy	CUG	Active
1.	/iservices_contentapi.1.j9czrqc4r5u5twe	--	Service	Container | Docker Container	62	System	Healthy	CUG	Active

Configuring the RabbitMQ PowerPack

You can monitor the RabbitMQ service with the AMQP: RabbitMQ PowerPack. This PowerPack monitors RabbitMQ configuration data and performance metrics using Dynamic Applications, and the PowerPack creates a major event in SL1 for any integration applications in the Integration Service that are in a *Failed* state.

After you install the RabbitMQ PowerPack in SL1, create a new SOAP/XML credential. Using that credential, you need to manually align the following Dynamic Applications:

- ScienceLogic: Integration Service Queue Configuration
- AMQP: RabbitMQ Configuration
- AMQP: RabbitMQ Performance

To configure the RabbitMQ PowerPack:

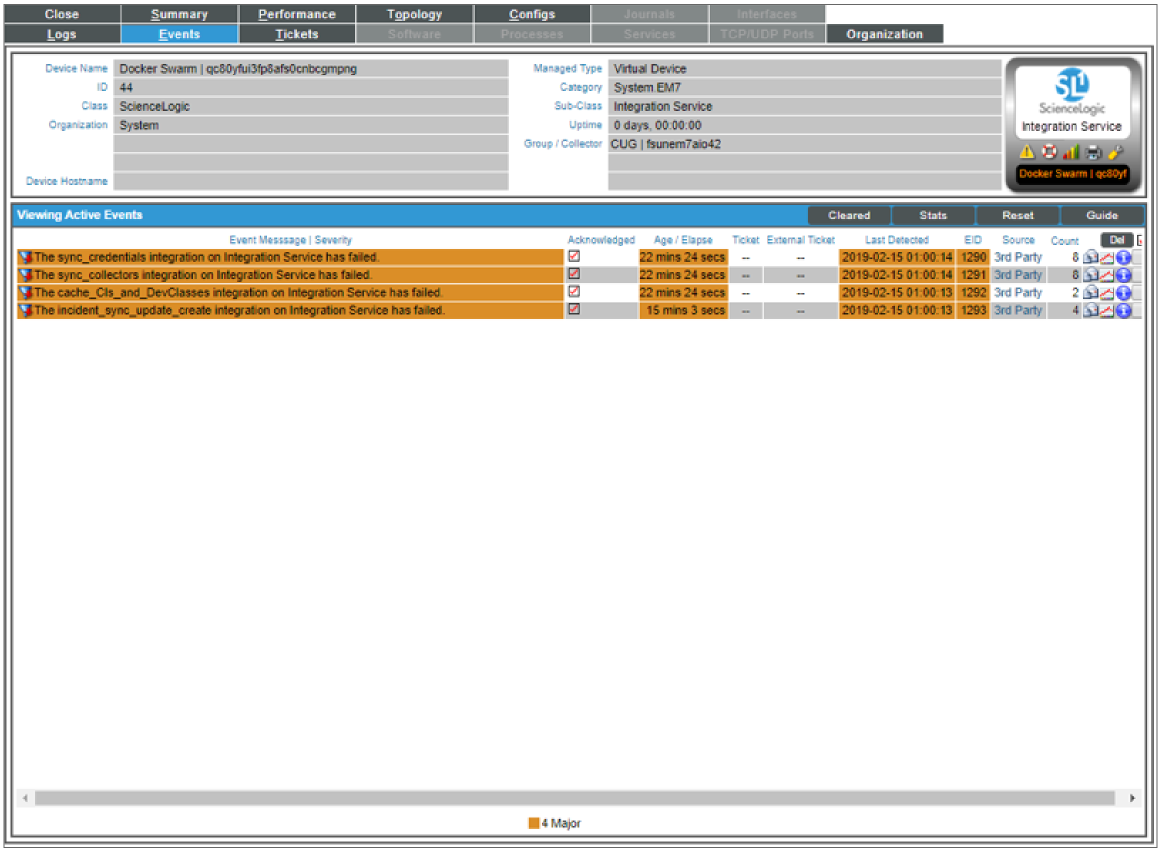
1. In SL1, navigate to the **Credential Management** page (System > Manage > Credentials) and click the wrench icon (🔧) for the **IS Sample** credential. The **Credential Editor** modal page appears.

2. Complete the following fields, and keep the other fields at their default settings:
   - ***Profile Name***. Type a new descriptive name for the credential.
   - ***URL***. Type the full URL for your Integration Service. For Integration Service 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 credentials at the Integration Service’s load balancer. The example above is for a single node deployment.

3. Click **[Save As]** and close the **Credential Editor** modal page.
4. Go to the **Device Components** page (Devices > Device Components) and click the wrench icon (🔧) for the Docker Swarm root device.
5. Click the **[Collections]** tab on the **Device Properties** window.
6. Align the "ScienceLogic: Integration Service Queue Configuration" Dynamic Application by clicking the **[Actions]** and selecting *Add Dynamic Application*. The **Dynamic Application Alignment** modal appears.

7. Select the "ScienceLogic: Integration Service Queue Configuration" Dynamic Application and select the credential that you created in step 2. 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):



Device Name: Docker Swarm | qc80yfui3fp8afs0cnbcgmpng
ID: 44
Class: ScienceLogic
Organization: System
Device Hostname:
Managed Type: Virtual Device
Category: System.EM7
Sub-Class: Integration Service
Uptime: 0 days, 00:00:00
Group / Collector: CUG | fsunem7aio42

Viewing Active Events

Event Messsage | Severity	Acknowledged	Age / Elapse	Ticket	External Ticket	Last Detected	EID	Source	Count
The sync_credentials integration on Integration Service has failed.	☑	22 mins 24 secs	--	--	2019-02-15 01:00:14	1290	3rd Party	8
The sync_collectors integration on Integration Service has failed.	☑	22 mins 24 secs	--	--	2019-02-15 01:00:14	1291	3rd Party	8
The cache_CIs_and_DevClasses integration on Integration Service has failed.	☑	22 mins 24 secs	--	--	2019-02-15 01:00:13	1292	3rd Party	2
The incident_sync_update_create integration on Integration Service has failed.	☑	15 mins 3 secs	--	--	2019-02-15 01:00:13	1293	3rd Party	4

🟧 4 Major

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>*.

8. 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>*

9. Align the "AMQP: RabbitMQ Configuration" and "AMQP: RabbitMQ Performance" Dynamic Applications using the same process as step 6.

Stability of the Integration Service Platform

This topic defines what a healthy SL1 system and a healthy Integration Service system look like, based on the following settings, metrics, and situations.

What makes up a healthy SL1 system?

To ensure the stability of your SL1 system, review the following settings in your SL1 environment:

- The SL1 system has been patched to a version that has been released by ScienceLogic within the last 12 months. ScienceLogic issues a software update at least quarterly. It is important for the security and stability of the system that customers regularly consume these software updates.
- The user interface and API response times for standard requests are within five seconds:
  - Response time for a specific user interface request.
  - Response time for a specific API request.
- At least 20% of local storage is free and available for new data. ***Free space*** is a combination of unused available space within InnoDB datafiles and filesystem area into which those files can grow
- The central system is keeping up with all collection processing:
  - Performance data stored and available centrally within three minutes of collection
  - Event data stored and available centrally within 30 seconds of collection
  - Run book automations are completing normally
- Collection is completing normally. Collection tasks are completing without early termination (sigterm).
- All periodic maintenance tasks are completing successfully:
  - Successfully completing daily maintenance (pruning) on schedule
  - Successfully completing backup on schedule
- High Availability and Disaster Recovery are synchronized (where used):
  - Replication synchronized (except when halted / recovering from DR backup).
  - Configuration matches between nodes.

What makes up a healthy Integration Service system?

To ensure the stability of the Integration Service, review the following settings in your environment:

- The settings from the previous list are being met in your SL1 system.
- You are running a supported version of the Integration Service.
- The memory and CPU percentage of the host remains less than 80% on core nodes.
- Task workloads can be accepted by the API and placed onto the queues for execution.

- The Integration Service API is responding to POST calls to run integration applications within the default timeout of 30 seconds. For standard integration applications triggers, this is usually sub-second.
- The Integration Service Scheduler not configured incorrectly. For example, there are no tasks accidentally set to run every minute or every second.
- Task workloads are actively being pulled from queues for execution by workers. Workers are actively processing tasks, and not just leaving items in queue.
- Worker nodes are all up and available to process tasks.
- Couchbase does not frequently read documents from disk. You can check this value with the "Disk Fetches per second" metric in the Couchbase user interface.
- The Couchbase Memory Data service memory usage is not using all allocated memory, forcing data writes to disk. You can check this value with the "Data service memory allocation" metric in the main Couchbase dashboard.
- Container services are not restarting.
- The RabbitMQ memory usage is not more than 2-3 GB per 10.000 messages in queues. The memory usage might be a little larger if you are running considerably larger tasks.
- RabbitMQ mirrors are synchronized.
- RabbitMQ is only mirroring the dedicated queues, not temporary or TTL queues.
- All Couchbase indexes are populated on all Couchbase nodes.
- The Couchbase nodes are fully rebalanced and distributed.
- The Docker Swarm cluster has at least three active managers in a High Availability cluster.
- For any Swarm node that is also a swarm manager, and that node is running Integration Service services :
    - At least one CPU with 4 GB of memory is available on the host to actively manage the swarm cluster.
    - Any Integration Service services running on this host are not able to consume all of the available resources, causing cluster operations to fail.

Some of the following Integration Service settings might vary, based on your configuration:

- The number of integration applications sitting in queue is manageable. A large number of applications sitting in queue could indicate either a large spike in workload, or no workers are processing.
- The number of failed tasks is manageable. A large number of failed tasks could be caused by ServiceNow timeouts, expected failure conditions, and other situations.
- ServiceNow is not overloaded with custom table transformations that cause long delays when the Integration Service is communicating with ServiceNow.

Chapter

10

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:

Initial 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 Synchronization PowerPack version, if applicable.
2. Make a note of whether the user is running an update set or the Certified Application (also called the "ScienceLogic SL1: CMDB & Incident Automation" 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.

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.

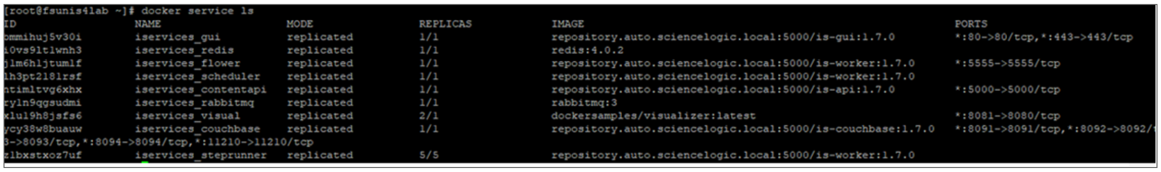
Use the following Docker commands for troubleshooting and diagnosing issues with the Integration Service:

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
ommihuj5v30i        iservices_gui          replicated          1/1                 repository.auto.sciencelogic.local:5000/is-gui:1.7.0      *:80->80/tcp,*:443->443/tcp
i0vs91tlwnh3        iservices_redis        replicated          1/1                 redis:4.0.2
jlm6hljtumlf        iservices_flower       replicated          1/1                 repository.auto.sciencelogic.local:5000/is-worker:1.7.0   *:5555->5555/tcp
lh3pt218lrsf        iservices_scheduler    replicated          1/1                 repository.auto.sciencelogic.local:5000/is-worker:1.7.0
ntimltvg6xhx        iservices_contentapi   replicated          1/1                 repository.auto.sciencelogic.local:5000/is-api:1.7.0      *:5000->5000/tcp
ryln9qgsudmi        iservices_rabbitmq     replicated          1/1                 rabbitmq:3
xlul9h8jsfs6        iservices_visual       replicated          2/1                 dockersamples/visualizer:latest                           *:8081->8080/tcp
ycy38w8buauw        iservices_couchbase    replicated          1/1                 repository.auto.sciencelogic.local:5000/is-couchbase:1.7.0   *:8091->8091/tcp,*:8092->8092/
3->8093/tcp,*:8094->8094/tcp,*:11210->11210/tcp
zlbxstxoz7uf        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 <service_name>
```

Stopping all Integration Service Services

Run the following command to stop all Integration Service services:

```
docker stack rm iservices
```

Restarting Docker

Run the following command to restart Docker:

```
systemctl restart docker
```

NOTE: Restarting Docker does not clear the queue.

Viewing Logs for a Specific Service

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
```

> **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
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": "d6406d3bdbc72300c40bdec0cf9619c2",
      "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

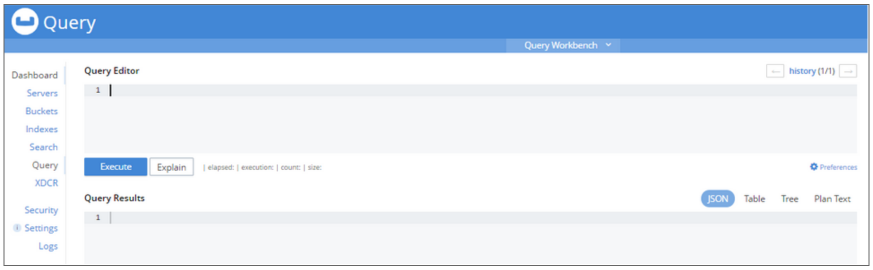
Locate the ID of the Couchbase container using the `docker ps` or `docker service ls` command, and the run the following commands:

```
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. UsingIntegration 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:

- ***Docker PowerPack***. This PowerPack monitors your Linux-based Integration Service server with SSH (the Integration Service ISO is built on top of an Oracle Linux Operating System). This PowerPack provides key performance indicators about how your Integration Service server is performing. For more information on the Docker PowerPack and other PowerPacks that you can use to monitor the Integration Service, see the "Using SL1 to Monitor the Integration Service" chapter in the ***Integration Service Platform*** manual.
- ***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.

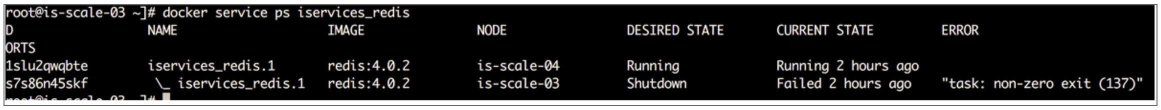
Step 1: Obtain the ID of the failed container for the service

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
```



```
root@is-scale-03 ~]# docker service ps iservices_redis
D                   NAME                    IMAGE               NODE                DESIRED STATE       CURRENT STATE          ERROR
ORTS
1slu2qwqbte         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* 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.

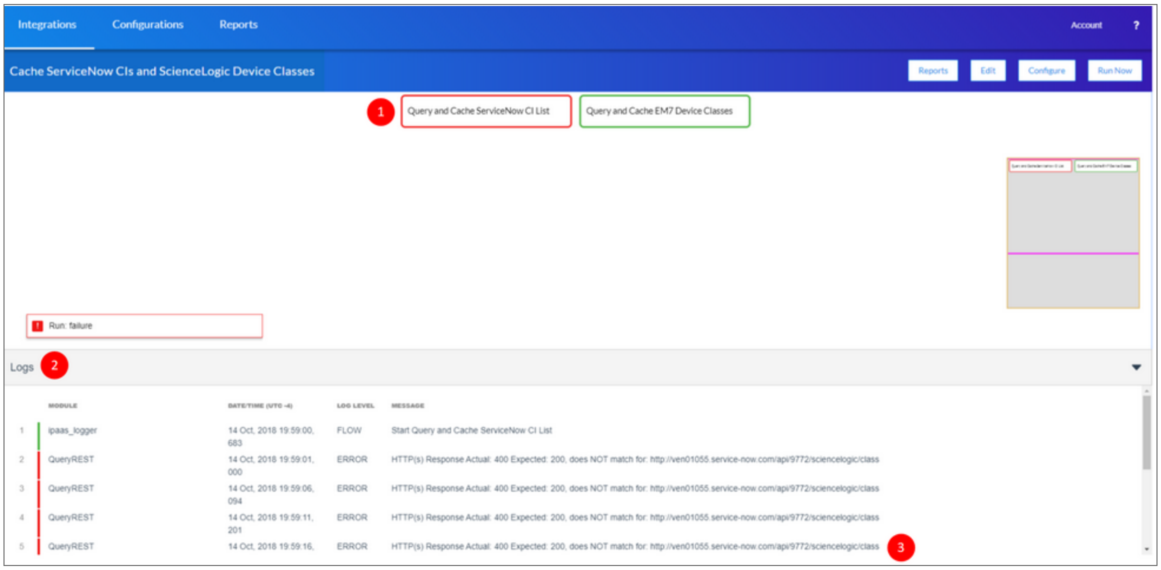
```
sudo docker service ps iservices_couchbase --no-trunc
```

```
"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.



Integrations
Configurations
Reports
Account
?
Cache ServiceNow CIs and ScienceLogic Device Classes
Reports
Edit
Configure
Run Now
1
Query and Cache ServiceNow CI List
Query and Cache EM7 Device Classes
Run: failure
Logs
2
3

```
   MODULE         DATE/TIME (UTC -4)        LOG LEVEL  MESSAGE
1  ipaas_logger   14 Oct, 2018 19:59:00,    FLOW       Start Query and Cache ServiceNow CI List
                  683
2  QueryREST      14 Oct, 2018 19:59:01,    ERROR      HTTP(s) Response Actual: 400 Expected: 200, does NOT match for: http://ven01055.service-now.com/api/9772/sciencelogic/class
                  000
3  QueryREST      14 Oct, 2018 19:59:06,    ERROR      HTTP(s) Response Actual: 400 Expected: 200, does NOT match for: http://ven01055.service-now.com/api/9772/sciencelogic/class
                  094
4  QueryREST      14 Oct, 2018 19:59:11,    ERROR      HTTP(s) Response Actual: 400 Expected: 200, does NOT match for: http://ven01055.service-now.com/api/9772/sciencelogic/class
                  201
5  QueryREST      14 Oct, 2018 19:59:16,    ERROR      HTTP(s) Response Actual: 400 Expected: 200, does NOT match for: http://ven01055.service-now.com/api/9772/sciencelogic/class
```

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)

The logs in the Integration Service use the following **loglevel** settings, from most verbose to least verbose:

- *10*. Debug Mode.
- *20*. Informational.
- *30*. Warning. This is the default settings if you do not specify a loglevel.
- *40*. Error.

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:

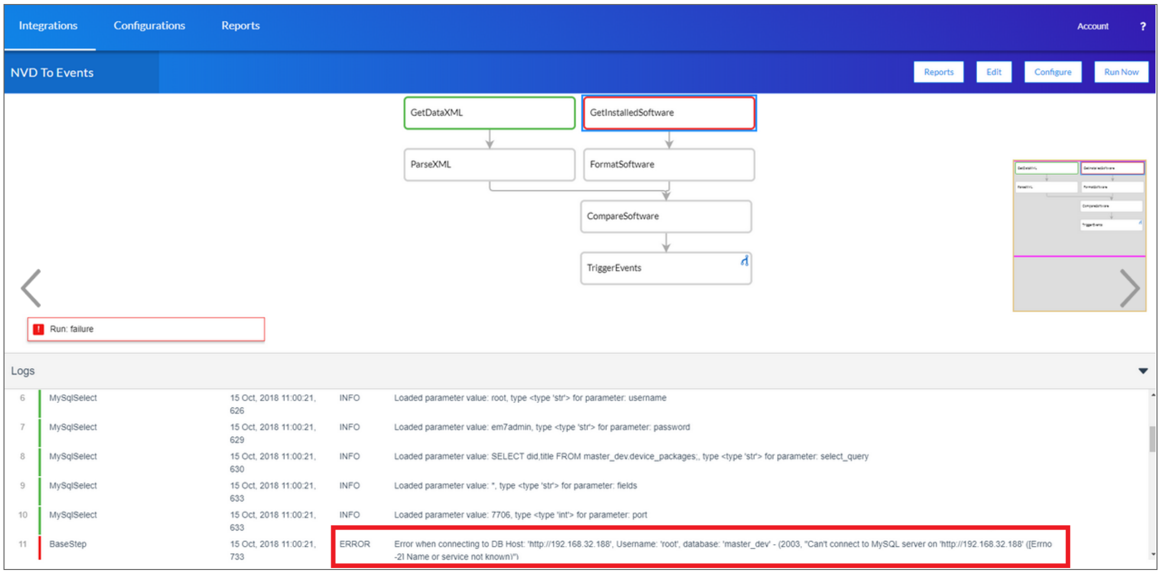
```
https://<integration_service>/api/v1/applications/run
```

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 integration application or step failed:



```
6   MySqlSelect   15 Oct, 2018 11:00:21,   INFO    Loaded parameter value: root, type <type 'str'> for parameter: username
                  626
7   MySqlSelect   15 Oct, 2018 11:00:21,   INFO    Loaded parameter value: em7admin, type <type 'str'> for parameter: password
                  629
8   MySqlSelect   15 Oct, 2018 11:00:21,   INFO    Loaded parameter value: SELECT did,title FROM master_dev.device_packages;, type <type 'str'> for parameter: select_query
                  630
9   MySqlSelect   15 Oct, 2018 11:00:21,   INFO    Loaded parameter value: *, type <type 'str'> for parameter: fields
                  633
10  MySqlSelect   15 Oct, 2018 11:00:21,   INFO    Loaded parameter value: 7706, type <type 'int'> for parameter: port
                  633
11  BaseStep      15 Oct, 2018 11:00:21,   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
                  733                              -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}}'
```

Frequently Asked Questions

This section contains a set of frequently asked questions (FAQs) and the answers to address those situations.

TIP: For additional troubleshooting information specific to multi-tenant environments, see ***Common Problems, Symptoms, and Solutions*** in the *Integration Service for Multi-tenant Environments* appendix.

What is the first thing I should do when I have an issue with the 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 address an "Error when connecting to DB Host" message when access is denied to user "root"?

In this situation, you get an error similar to the following:

```
Error when connecting to DB Host... (1045, "Access denied for user
'root'@'10.86.21.224' (using password: NO)"
```

This issue occurs when the **encryption_key** file from one Integration Service system does not match the **encryption_key** file on another system, such as when you take data from a production Integration Service system to a test Integration Service system.

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.

To address this issue:

1. Copy the **encryption_key** file from the **/etc/iservices/** folder on the production system to the **/etc/iservices/** folder on the test system.
2. Re-upload all integration applications and configurations from the production system to the test system.
3. Remove and redeploy the stack on the test Integration Service after copying the key by running the following command on the test system:

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

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.

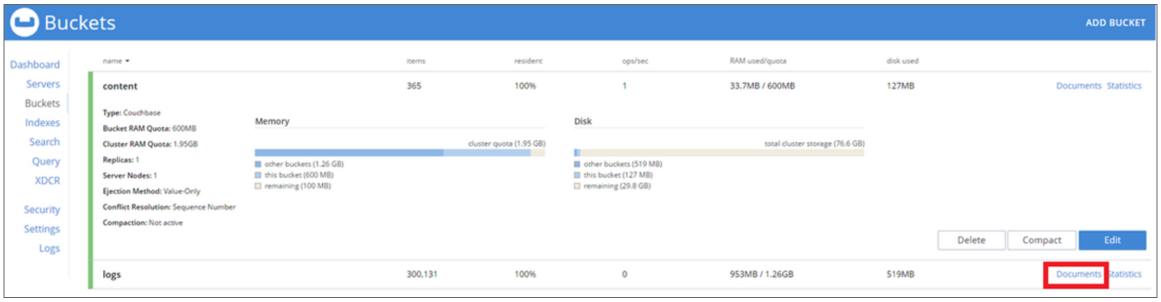
NOTE: This issue only affects versions of the Integration Service prior to version 1.8.2.

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:

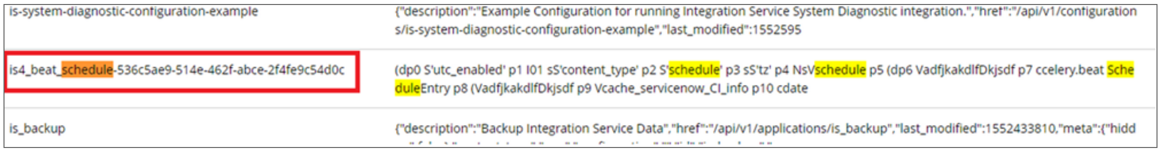
1. Log in to the Couchbase management interface at **https://<*localhost*>:8091**.
2. Navigate to the **[Buckets]** tab.
3. Click the **Documents** link on the **content** bucket:



name ▾	items	resident	ops/sec	RAM used/quota	disk used	
content	365	100%	1	33.7MB / 600MB	127MB	Documents Statistics
logs	300.131	100%	0	953MB / 1.26GB	519MB	Documents Statistics

Type: Couchbase
Bucket RAM Quota: 600MB
Cluster RAM Quota: 1.95GB
Replicas: 1
Server Nodes: 1
Ejection Method: Value-Only
Conflict Resolution: Sequence Number
Compaction: Not active

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/configurations/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 sS'tz' p4 NsVschedule p5 (dp6 VadfjkakdlfDkjsdf p7 ccelery.beat ScheduleEntry p8 (VadfjkakdlfDkjsdf p9 Vcache_servicenow_CI_info p10 cdate
is_backup	{"description":"Backup Integration Service Data","href":"/api/v1/applications/is_backup","last_modified":1552433810,"meta":{"hidd

6. Delete the file.

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

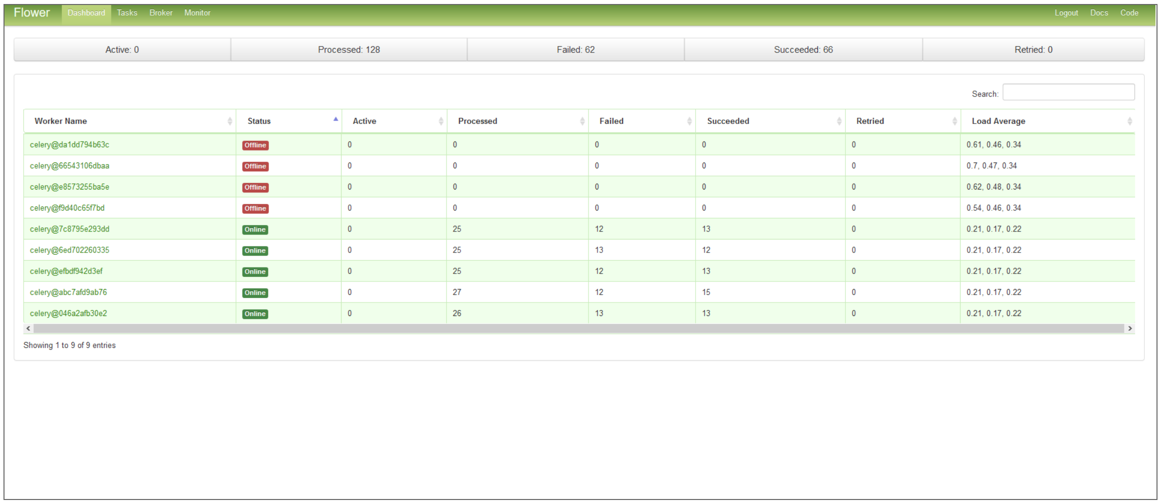
How do I identify and fix a deadlocked state?

If the Integration Service appears to be running, but the Integration Service is not processing any new integrations or tasks, then the Integration Service could be in a ***deadlocked*** state.

A deadlocked state occurs when one or more integration applications include steps that are either ordered improperly or that contain syntax errors. In this situation, tasks are waiting on subsequent tasks to finish executing, but the worker pool is exhausted. As a result, the Integration Service is not able to execute those subsequent tasks.

To identify a deadlocked state with an Integration Service system:

1. Navigate to the Celery Flower interface for the Integration Service by typing the URL or IP address for your Integration Service and adding **/flower** at the end of the URL, such as **https://192.0.2.0/flower/**.
2. Click the **[Dashboard]** tab for Flower. A list of workers appears:



Active: 0 | Processed: 128 | Failed: 62 | Succeeded: 66 | Retried: 0

Worker Name	Status	Active	Processed	Failed	Succeeded	Retried	Load Average
celery@da1dd794b63c	Offline	0	0	0	0	0	0.61, 0.46, 0.34
celery@66543106dbaa	Offline	0	0	0	0	0	0.7, 0.47, 0.34
celery@e8573255ba5e	Offline	0	0	0	0	0	0.62, 0.48, 0.34
celery@f9d40c65f7bd	Offline	0	0	0	0	0	0.54, 0.46, 0.34
celery@7c8795e293dd	Online	0	25	12	13	0	0.21, 0.17, 0.22
celery@6ed702260335	Online	0	25	13	12	0	0.21, 0.17, 0.22
celery@efbdf942d3ef	Online	0	25	12	13	0	0.21, 0.17, 0.22
celery@abc7afd9ab76	Online	0	27	12	15	0	0.21, 0.17, 0.22
celery@046a2afb30e2	Online	0	26	13	13	0	0.21, 0.17, 0.22

Showing 1 to 9 of 9 entries

3. Review the number of active or running tasks for all workers. If all workers have the maximum number of tasks, and no new tasks are being consumed, then you might have a deadlock state.

To fix a deadlocked state in an Integration Service system, perform one of the following steps:

1. Go to the console of the Integration Service system or use SSH to access the server.
2. Log in as ***isadmin*** with the appropriate password.
3. Increase the number of workers by either:

   - Executing the following at the shell prompt:

   ```
   docker service scale iservices_steprunner=x
   ```

   where:

     - *x* is the number of workers.

   - Using a text editor like vi to edit the file **/opt/iservices/scripts/docker-compose.yml**.

     - In the **environment:** section at the top of the file, add the following:

   ```
   worker_threads: number_greater_than_3
   ```

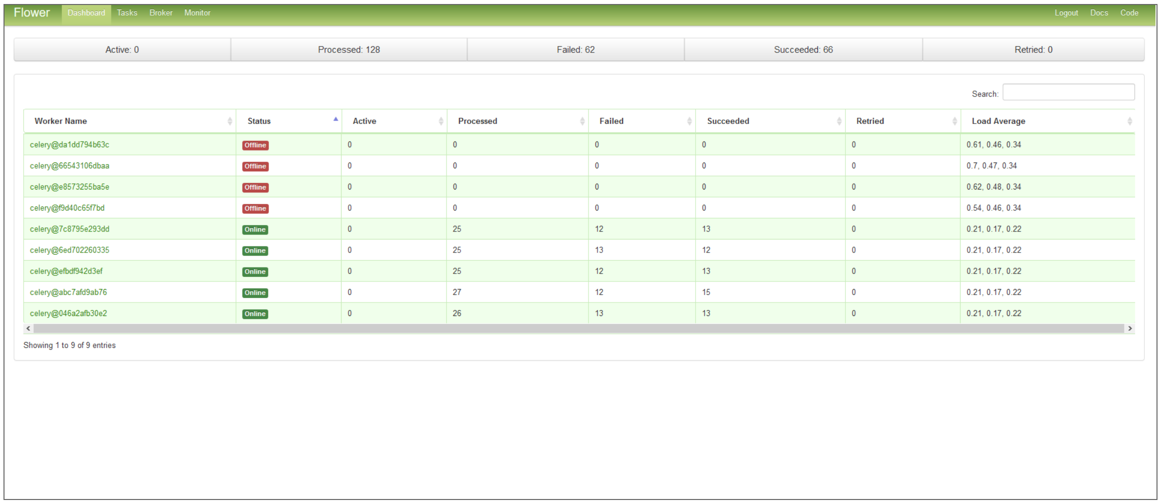
   where:

     - *number_greater_than_3* is an integer greater than 3.

4. After you have updated the docker-compose file, you can update and re-deploy the Integration Service system to pick up the changes in the docker-compose.yml file To do this, execute the following at the shell prompt:

   ```
   docker stack deploy -c /opt/iservices/scripts/docker-compose.yml is4
   ```

5. The Integration Service system should now include additional workers.
6. Navigate to the Celery Flower interface for the Integration Service by typing the URL or IP address for your Integration Service and adding **/flower** at the end of the URL, such as **https://192.0.2.0/flower/**.
7. Click the **[Dashboard]** tab for Flower. A list of workers appears:



Active: 0 | Processed: 128 | Failed: 62 | Succeeded: 66 | Retried: 0

Search:

Worker Name	Status	Active	Processed	Failed	Succeeded	Retried	Load Average
celery@da1dd794b63c	Offline	0	0	0	0	0	0.61, 0.46, 0.34
celery@66543106dbaa	Offline	0	0	0	0	0	0.7, 0.47, 0.34
celery@e8573255ba5e	Offline	0	0	0	0	0	0.62, 0.48, 0.34
celery@f9d40c65f7bd	Offline	0	0	0	0	0	0.54, 0.46, 0.34
celery@7c8795e293dd	Online	0	25	12	13	0	0.21, 0.17, 0.22
celery@6ed702260335	Online	0	25	13	12	0	0.21, 0.17, 0.22
celery@efbdf942d3ef	Online	0	25	12	13	0	0.21, 0.17, 0.22
celery@abc7afd9ab76	Online	0	27	12	15	0	0.21, 0.17, 0.22
celery@046a2afb30e2	Online	0	26	13	13	0	0.21, 0.17, 0.22

Showing 1 to 9 of 9 entries

8. Review the number of active or running tasks for all workers.

How can I optimize workers, queues, and tasks?

The Integration Service system uses Celery to spawn and manage worker processes and queues. You can define environment variables to optimize these worker processes.

You can use the following environment variables to optimize worker processes:

- ***task_soft_time_limit***. Enforces global timeout for all tasks in the Integration Service system. If a task exceeds the specified timeout limit, the Integration Service system terminates the task so that the next task in the queue can be processed. Possible values are:
  - Integer that specifies the time in seconds.
  - Default value is "3600" (1 hour).
- ***optimization***. Determines how tasks are distributed to worker processes. Possible values are:
  - *-Ofair*. Celery distributes a task only to the worker process that is available for work.
  - " " (double-quotation mark, space, double-quotation mark). Distributes and queues all tasks to all available workers. Although this increases performance, tasks in queues might be delayed waiting for long-running tasks to complete.
  - Default value is *-Ofair*.
- ***task_acks_late***. Specifies that if a worker process crashes while executing a task, Celery will redistribute the task to another worker process. Possible values are:
  - *True*. Enables the environment variable.
  - *False*. disabled the environment variable.
  - Default value is "False"

NOTE: Because many integration applications run at regular intervals or are scheduled, the Integration Service system re-executes tasks even if the ***task_acks_late*** environment variable is disabled. in the event of a worker crash, if you want to ensure that tasks are completed, you can enable the ***task_acks_late*** variable. However, be aware that if tasks are not idempotent, the ***task_acks_late*** variable can cause unpredictable results.

To define these environment variables:

1. Either go to the console of the Integration Service system or use SSH to access the server.
2. Log in as ***isadmin*** with the appropriate password.
3. Use a text editor like vi to edit the file **/opt/iservices/scripts/docker-compose.yml**.
4. You can define the environment variables for one or more worker processes. The docker-compose.yml file contains definitions for worker processes. For example, you might see something like this:

```
services:
```

```
steprunner:
  image: sciencelogic/is-worker:latest
  environment:
  LOGLEVEL: 10
  celery_log_level: 10
  logdir: /var/log/iservices
  broker_url: 'pyamqp://guest@rabbit//'
  result_backend: 'redis://redis:6379/0'
  db_host: 'couchbase,localhost'
  secrets:
    - is_pass
    - encryption_key
  deploy:
  replicas: 2
  networks:
    - isnet
  depends_on:
    - redis
    - rabbitmq
    - couchbase
  volumes:
    - "/var/log/iservices:/var/log/iservices"
    - "statedb:/var/run/celery/states/"

steprunner_1:
  image: : sciencelogic/is-worker:latest
  environment:
  LOGLEVEL: 10
  celery_log_level: 10
  task_soft_time_limit: 30
  optimization: ""
  task_acks_late: 'False'
  logdir: /var/log/iservices
  broker_url: 'pyamqp://guest@rabbit//'
  result_backend: 'redis://redis:6379/0'

  db_host: 'couchbase,localhost'
  secrets:
    - is_pass
    - encryption_key
  deploy:
  replicas: 2
  networks:
    - isnet
  depends_on:
    - redis
    - rabbitmq
    - couchbase
  volumes:
    - "/var/log/iservices:/var/log/iservices"
    - "statedb:/var/run/celery/states/"

steprunner_2:
  image: : sciencelogic/is-worker:latest
  environment:
  LOGLEVEL: 10
  celery_log_level: 10
  task_soft_time_limit: 30
  optimization: '-Ofair'
  task_acks_late: 'False'
  logdir: /var/log/iservices
```

```
broker_url: 'pyamqp://guest@rabbit//'
result_backend: 'redis://redis:6379/0'
db_host: 'couchbase,localhost'
secrets:
  - is_pass
  - encryption_key
deploy:
replicas: 2
networks:
  - isnet
depends_on:
  - redis
  - rabbitmq
  - couchbase
volumes:
  - "/var/log/iservices:/var/log/iservices"
  - "statedb:/var/run/celery/states/"
```

5. The services with names that start with "steprunner" are the workers for the Integration Service system. To define the optimization variables, enter the variables and values in the definition of the worker, under the ***environment*** section. See the example in step #3 to see the syntax for environment variables.
6. After you have updated the docker-compose file, you can update and re-deploy the Integration Service system to pick up your changes to the file. To do this, execute the following command:

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml is4
```

Why do I have 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 contributeto 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, go to the **[Events Console]** (classic user interface) or the **Events** page (new user interface) and locate the event that was created.
2. Click the **View Notification Log** mailbox icon (✉) for that event. The **Event Actions Log** window appears.
3. On the **Event Actions Log** window, verify that the Run Book Action was triggered, and that the Run Book Action successfully posted to the Integration Service.
4. 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.
5. Look at the logs for that run of the Integration Service integration application.

What if my Incident does not have a CI?

For an incident with an active event:

1. In SL1, go to the **[Events Console]** (classic user interface) or the **Events** page (new user interface) and locate the event that was created.
2. Click the **View Notification Log** mailbox icon (✉) for that event. The **Event Actions Log** window appears.
3. On the **Event Actions Log** window, locate the Integration Service run ID.
4. Open the integration application that used that run and review the Step Log.
5. Confirm that the device class was mapped in the "Sync Devices from SL1 to ServiceNow" integration application.
6. Confirm that the "Sync Devices from SL1 to ServiceNow" integration application is running at least every 24 hours, and that the "Sync Devices from SL1 to ServiceNow" integration application has run within 24 hours of that event sync run.

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]
```

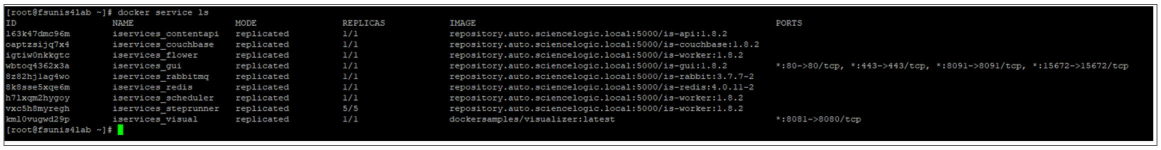
What do I do if I get a Code 500 Error when I try to access the Integration Service user interface?

To address this issue:

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

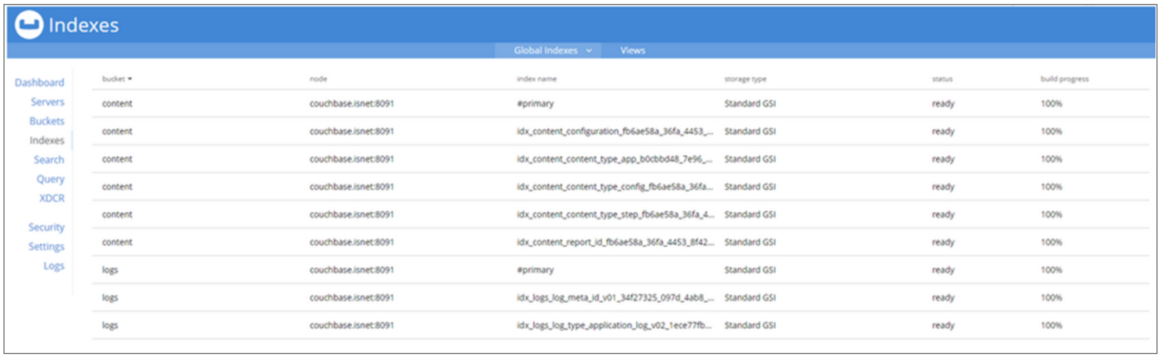
```
docker service ls
```

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



```
[root@fsunis4lab ~]# docker service ls
ID              NAME                  MODE          REPLICAS      IMAGE                                                  PORTS
163k47dmc96m    iservices_contentapi  replicated    1/1           repository.auto.sciencelogic.local:5000/is-api:1.8.2
oaptzsijq7x4    iservices_couchbase   replicated    1/1           repository.auto.sciencelogic.local:5000/is-couchbase:1.8.2
igtiw0nkkgtc    iservices_flower      replicated    1/1           repository.auto.sciencelogic.local:5000/is-worker:1.8.2
wbtoq4362x3a    iservices_gui         replicated    1/1           repository.auto.sciencelogic.local:5000/is-gui:1.8.2       *:80->80/tcp, *:443->443/tcp, *:8091->8091/tcp, *:15672->15672/tcp
8z82hjlag4wo    iservices_rabbitmq    replicated    1/1           repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2
8k8sse5xqe6m    iservices_redis       replicated    1/1           repository.auto.sciencelogic.local:5000/is-redis:4.0.11-2
h7lxqm2hygoy    iservices_scheduler   replicated    1/1           repository.auto.sciencelogic.local:5000/is-worker:1.8.2
vxc5h8myregh    iservices_steprunner  replicated    5/5           repository.auto.sciencelogic.local:5000/is-worker:1.8.2
kml0vugwd29p    iservices_visual      replicated    1/1           dockersamples/visualizer:latest                            *:8081->8080/tcp
[root@fsunis4lab ~]# 
```

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:



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_1ece77fb...	Standard GSI	ready	100%

6. Wait until all **status** entries are *ready* and 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 should I do if I get a 500 Error?

A 500 Internal Server Error will always have some kind of stack trace in the contentapi logs. Run the following command to find the cause of the 500 Error:

```
docker service logs -t iservices_contentapi
```

A 502 or 504 Error might mean that the user interface cannot reach an API container on a node, or the API container cannot reach a database node in the cluster. To address this issue:

- For a cluster, make sure the cluster is healthy, and all database nodes are balanced.
- For a cluster, make sure that the firewall ports are open on all nodes.
- Run the following commands to check the logs for 502 or 504 Errors:

  ```
  docker service logs -t iservices_gui
  docker service logs -t iservices_contentapi
  ```

  The logs will specify which container caused a timeout when trying to reach that container.

What are some common examples of using the iscli tool?

The Integration Service system includes a command line utility called the iscli tool. You can use the iscli tool 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 the iscli tool 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>
```

NOTE: The password for the iscli tool should be the same password as the Integration Service Administrator (*isadmin*) user password. For more information, see ***Changing the Integration Service Password***.

How do I view a specific run of an integration application on the Integration Service?

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
```

Why am I getting an "ordinal not in range" step error?

If you get an "ordinal not in range" error, check your CI Class Mappings to make sure the mappings do not contain international or "special" characters.

For example:

```
AWS | Availability Zone - São Paulo
```

If you find a class mapping with a special character like the above example, remove the class mapping, or rename the device class in SL1 to not include the special characters. Then you can sync the CI classes again.

How do I clear a backlog of Celery tasks in Flower?

To clear a backlog of Celery tasks:

1. docker exec into a bash shell in a worker process. For example:

   ```
   docker exec -it e448db31aaec /bin/bash
   ```

   where *e448db31aaec* is the container ID of the is-worker process on your system

2. Run the Python interpreter.
3. Run the following code:

   ```
   from ipaascommon.celeryapp import app

   app.control.purge()
   ```

Why does traffic from specific subnets not get a response from the Integration Service?

In this situation, you can see traffic going into the host and into the Docker network, the traffic is not being routed back out. Responses were lost in the Docker ingress network, and the client times out.

To address this issue:

1. Remove the Docker service by running the following command:

   ```
   docker stack rm iservices
   ```

2. Remove the default ingress network:

   ```
   docker network rm ingress
   ```

3. Add a newly addressed ingress network:

   ```
   docker network create --driver overlay --ingress --subnet=172.16.0.0/16 --
   gateway=172.16.0.1 ingress
   ```

4. Redeploy the Integration Service:

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

If the containers have an exposed port and you find the following error in the logs, you might need to remove /var/lib/docker/network/files/local-kv.db:

```
error="failed to detect service binding for container iservices_gui…"
```

To address this issue:

1. Remove the Docker service:

   ```
   docker stack rm iservices
   ```

2. Remove the .db file:

   ```
   rm /var/lib/docker/network/files/local-kv.db
   ```

3. Restart the docker daemon

   ```
   systemctl restart docker
   ```

4. Redeploy the Integration Service:

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

What if the Integration Service user interface is down and Incidents are not being generated in ServiceNow?

If the Integration Service user interface is unresponsive and Incidents are not being generated in ServiceNow, a change to the firewall rules during deployment might have impacted the ingress network for Docker.

To address this issue, restart DFocker after any firewall or network configuration change by running the following command:

```
systemctl restart docker
```

Why does the latest tag not exist after the initial ISO installation?

This situation only affects users with custom services that point to the latest tag. To work around this issue, run the tag latest script manually after running the `./pull_start_iservices.sh` command:

```
python /opt/iservices/scripts/system_updates/tag_latest.py
/opt/iservices/scripts/docker-compose.yml
```

Chapter 11

API Endpoints in the Integration Service

Overview

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

This chapter covers the following topics:

Interacting with the API

To view the full documentation for the IS API:

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

Available Endpoints

POST

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

/applications/{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.

/license. Add license data for this Integration Service system.

/roles/owner. Add a new owner assigned a specific role.

/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.

/syncpacks/{syncpackName}/install. Install a specific Synchronization PowerPack version by name.

/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

/about. Retrieve version information about the packages used by this Integration Service system.

/applications. Retrieve a list of all available applications on this Integration Service system.

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

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

/cache/{cache_id}. Retrieve a specific cache to gather information about the user interface and the integration applications.

/configurations. Retrieve a list of all configurations on this Integration Service system.

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

/license. Retrieve license data for this Integration Service system.

/reports. Retrieve a list of paginated reports.

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

/roles. Retrieve a list of available roles on this Integration Service system.

/roles/owner. Retrieve a list of roles assigned to owners on this Integration Service system.

/roles/owner/{owner}. Retrieve the role assigned to a specific owner.

/schedule. Retrieve a list of all scheduled application integrations on this Integration Service system.

/steps. Retrieve a list of all steps on this Integration Service system.

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

/syncpacks. Retrieve a list of all Synchronization PowerPacks on this Integration Service system.

/syncpacks/{synpackName}. Retrieve the full details about a specific Synchronization PowerPack.

/api/v1/syncpacks?only_installed=true. Retrieve a list of only the installed Synchronization PowerPacks on this system.

/api/v1/syncpacks?only_activated=true. Retrieve a list of only the activated Synchronization PowerPacks on this system.

/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.

/license. Delete license data for this Integration Service system.

/roles/owner. Delete a specific owner role.

/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.

/syncpacks/{spName}. Delete a specific Synchronization PowerPack by name.

Appendix

A

Integration Service for Multi-tenant Environments

Overview

This appendix describes the best practices and troubleshooting solutions for deploying the Integration Service in a multi-tenant 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:

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 multi-tenant environment. After the initial High Availability (HA) core services are deployed, the multi-tenant environment differs in the deployment and placement of workers and use of custom queues.

Core Service Nodes

For a multi-tenant 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 addling 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).

Advanced RabbitMQ Administration and Maintenance

This section describes how multi-tenant deployments can use separate virtual hosts and users for each tenant.

Using an External RabbitMQ Instance

In certain scenarios, you might not want to use the default RabbitMQ queue that is prepackaged the Integration Service. For example, you might already have a RabbitMQ production cluster available that you just want to connect with the Integration Service. You can do this by defining a new virtual host in RabbitMQ, and then you configure the Integration Service broker URL for contentapi, steprunner, scheduler services so that they point to the new virtual host.

Any use of an external RabbitMQ server will not be officially supported by ScienceLogic if there are issues in the external RabbitMQ instance.

Setting a User other than Guest for Queue Connections

By default RabbitMQ contains the default credentials *guest/guest*.The Integration Service uses these credentials by default when communicating with RabbitMQ in the swarm cluster. All communication is encrypted and secured within the overlay Docker network.

To add another user, or to change the user that the Integration Service uses when communicating with the queues:

1. Create a new user in RabbitMQ that has full permissions to a virtual host. For more information, see the RabbitMQ documentation.
2. Update the ***broker_url*** environment variable with the new credentials in the **docker-compose** file and then re-deploy.

Configuring the Broker (Queue) URL

When using an external RabbitMQ system, or if you are using credentials other than *guest/guest* to authenticate, you need to update the ***broker_url*** environment variable in the contentapi, steprunner, and scheduler services. You can do this by modifying the environment section of the services in **docker-compose** and changing ***broker_url***. The following line is an example:

```
broker_url: 'pyamqp://username:password@rabbitmq-hostname/v-host'
```

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 multi-tenant 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 multi-tenant 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**:

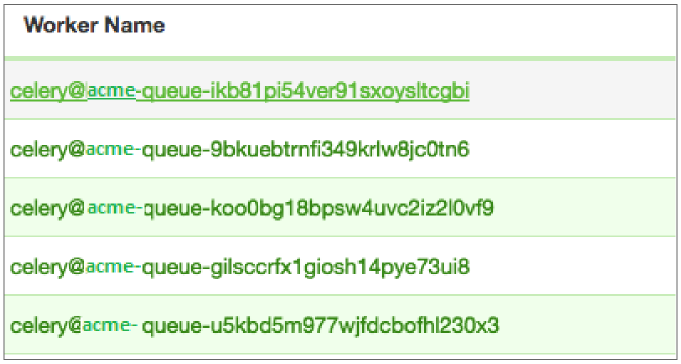
1. Copy an existing steprunner service definition.
2. Change the steprunner service name to something unique for the stack:
   - steprunner-acmequeue
   - steprunner-acmequeue-catchup
3. Modify the "replicas" value to specify how many workers should be listening to this queue:
   - steprunner-acmequeue 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:

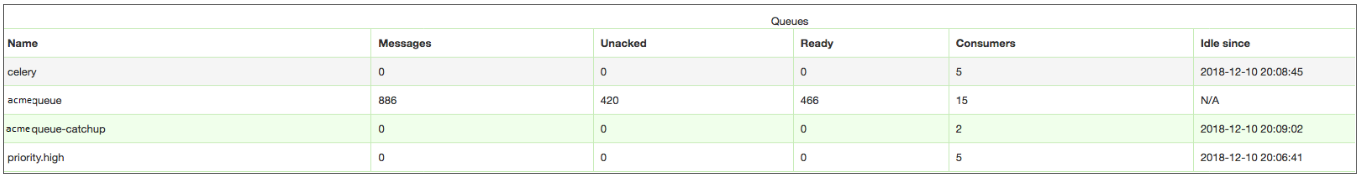
```
steprunner-acme-catchup:
  image: sciencelogic/is-worker:latest
  depends_on:
  - couchbase
  - rabbitmq
  - redis
  hostname: "acme-catchup-{{.Task.ID}}"
  deploy:
    placement:
      constraints:
        - node.labels.customer == acme
    resources:
      limits:
        memory: 2G
    replicas: 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-9bkuebtrnfi349krlw8jc0tn6
celery@acme-queue-koo0bg18bpsw4uvc2iz2l0vf9
celery@acme-queue-gilsccrfx1giosh14pye73ui8
celery@acme- queue-u5kbd5m977wjfdcbofhl230x3

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

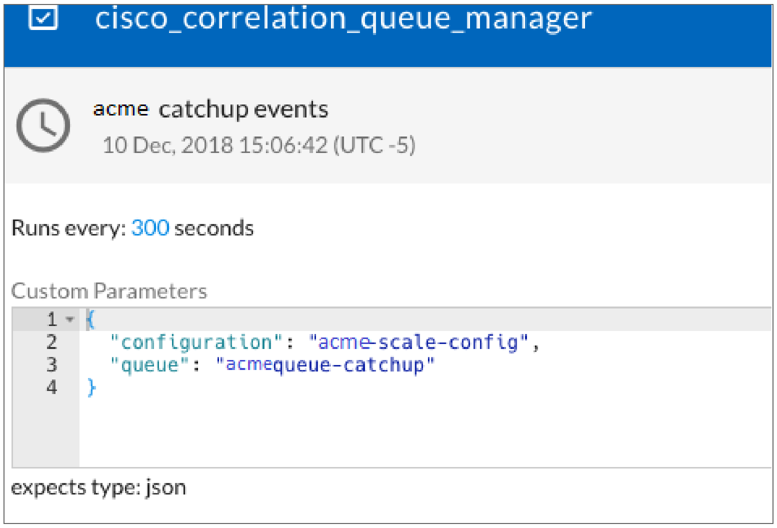
Name	Messages	Unacked	Ready	Consumers	Idle since
celery	0	0	0	5	2018-12-10 20:08:45
acmequeue	886	420	466	15	N/A
acme queue-catchup	0	0	0	2	2018-12-10 20:09:02
priority.high	0	0	0	5	2018-12-10 20:06:41

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 acmecqueue-catchup queue:



☑ cisco_correlation_queue_manager

acme catchup events

10 Dec, 2018 15:06:42 (UTC -5)

Runs every: 300 seconds

Custom Parameters

```
{
   "configuration": "acme-scale-config",
   "queue": "acmequeue-catchup"
}
```

expects type: json

In the example above, the **cisco_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 effect 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 the other 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": "sl1_host",
    "value": "${config.em7_host}"
  },
  {
    "encrypted": false,
    "name": "sl1_id",
    "value": "${config.em7_id}"
  },
  {
    "encrypted": false,
    "name": "sl1_db_port",
    "value": 7706
  },
  {
    "encrypted": false,
    "name": "snow_host",
    "value": "<arecord>.service-now.com"
  },
  {
    "encrypted": true,
    "name": "em7_password",
    "value": "<password>"
  },
  {
    "encrypted": false,
    "name": "sl1_user",
    "value": "${config.em7_user}"
  },
  {
    "encrypted": false,
    "name": "sl1_password",
    "value": "${config.em7_password}"
  },
  {
    "encrypted": false,
    "name": "sl1_db_user",
    "value": "${config.em7_db_user}"
  },
  {
    "encrypted": false,
    "name": "sl1_db_password",
    "value": "${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 *cbbackup* 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 multi-tenant 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 andRredis 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".	When viewing the Docker Visualizer user interface, some services are displayed as "undefined", and states aren't accurate. **Impact**: Cannot use Visualizer to get the current state of the stack.	Failing docker stack deployment: https://github.com/dockersamples/docker-swarm-visualizer/issues/110	Ensure your stack is healthy, and services are deployed correctly. If no services are failing and things are still showing as undefined, ***elect a new swarm leader***. **To prevent:** Ensure your configuration is valid before deploying.

Tool	Issue	Symptoms	Cause	Solution
RabbitMQ	RabbitMQ queues encountered a node failure and are in a "Network partition state" (split-brain scenario).	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. Log in to the RabbitMQ admin user interface, which displays a message similar to "RabbitMQ experienced a network partition and the cluster is paused". **Impact**: The RabbitMQ cluster is paused and waiting for user intervention to clean the split-brain state.	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) **Note**: ScienceLogic sets the *autoheal* policy by default	Handle the split-brain partition state and ***resynchronize your RabbitMQ queues***. **Note**: This is enabled by default. **To prevent:** Set a partition handling policy. See the ***Resiliency*** section for more information.

Tool	Issue	Symptoms	Cause	Solution

RabbitMQ, ***continued***		Execing into the RabbitMQ container and running **rabbitmqcli cluster-status** shows nodes in a partition state like the following:
		`[{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',[]}]}]`		

Integration Service steprunners and RabbitMQ	Workers constantly restarting, no real error message.	Workers of a particular queue are not stable and constantly restart. **Impact**: One queue's workers will not be processing.	Multi-node failure in RabbitMQ, when it loses majority and can not failover. Queues go out of sync because of broken swarm.	***Recreate queues*** for the particular worker. ***Resynchronize queues***. **To prevent**: Deploy enough nodes to ensure quorum for failover.

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

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

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 `--max-tasks-per-child` setting. At high CPU workloads, the `max-tasks-per-child` setting causes workers to exit prematurely.	Restart the flower service by running the following command: `docker service update --force iservices_flower` You can also remove the `--max-tasks-per-child` 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: `docker node ls` **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**: *Use drained managers to maintain swarm*.
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.	*Identify the cause of the service not deploying*.
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

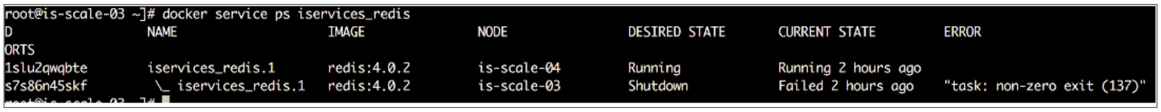
Step 1: Obtain the ID of the failed container for the service

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
D                  NAME                    IMAGE          NODE           DESIRED STATE    CURRENT STATE        ERROR
ORTS
1slu2qwqbte        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

   ```
   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.

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/RabbitMG.

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
```

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

SL1 ScienceLogic

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

International: +1-703-354-1010