Red Hat Fuse 7.3

Integrating Applications with Fuse Online

For business users who want to share data between different applications and services.

Last Updated: 2020-03-11

Red Hat Fuse 7.3 Integrating Applications with Fuse Online

For business users who want to share data between different applications and services.

Legal Notice

Abstract

Fuse Online provides integration as a service.

Table of Contents

PREFACE

Red Hat Fuse is a distributed, cloud-native integration solution that provides choices for architecture, deployment, and tools. Fuse Online is Red Hat's web-based Fuse distribution. Syndesis is the open source project for Fuse Online. Fuse Online runs on OpenShift Online, OpenShift Dedicated, and OpenShift Container Platform.

This guide provides information and instructions for using Fuse Online's web interface to integrate applications. The content is organized as follows:

- Chapter 1, *High level overview of Fuse Online*
- Chapter 2, *How to get ready to create integrations*
- Chapter 3, *What to expect when using Fuse Online for the first time*
- Chapter 4, *About connections to applications that you want to integrate*
- Chapter 5, *Creating integrations*
- Chapter 6, *Creating an integration that is triggered by a REST API call*
- Chapter 7, *Mapping integration data to fields for the next connection*
- Chapter 8, *Managing integrations*
- Chapter 9, *Customizing Fuse Online*
- Chapter 10, *Installing and managing Fuse Online on OpenShift Container Platform*

To learn how to use Fuse Online by creating sample integrations, see the sample integration tutorials.

To obtain support, in Fuse Online, in the upper right, click ? and then select **Support**.

CHAPTER 1. HIGH LEVEL OVERVIEW OF FUSE ONLINE

With Fuse Online, you can obtain data from an application or service, operate on that data if you need to, and then send the data to a completely different application or service. No coding is required to accomplish this.

The following topics provide a high-level overview of Fuse Online:

- Section 1.1, “How Fuse Online works”
- Section 1.2, “Who Fuse Online is for”
- Section 1.3, “Benefits of using Fuse Online”
- Section 1.4, “Descriptions of Fuse Online constructs”

1.1. HOW FUSE ONLINE WORKS

Fuse Online provides a web browser interface that lets you integrate two or more different applications or services without writing code. It also provides features that allow you to introduce code if it is needed for complex use cases.

Fuse Online lets you enable data transfer between different applications. For example, a business analyst can use Fuse Online to capture tweets that mention customers and then leverage the data obtained from Twitter to update Salesforce accounts. Another example is a service that makes stock trade recommendations. You can use Fuse Online to capture recommendations for buying or selling stocks of interest and forward those recommendations to a service that automates stock transfers.

To create and run a simple integration, the main steps are:

1. Create a connection to each application that you want to integrate.
2. Select the start connection. This connection is to the application that contains the data that you want to share with another application.
   Alternatively, you can start the integration with a timer or a webhook that accepts an HTTP request.
3. Select the finish connection. This connection is to the application that receives data from the start connection and that completes the integration.
4. Map data fields from the start connection to data fields in the finish connection.
5. Give the integration a name.
6. Click **Publish** to start running the integration.

Another kind of integration is an API provider integration. An API provider integration allows REST API clients to invoke commands that trigger execution of the integration. To create and run an API provider integration, you upload an OpenAPI 2.0 document to Fuse Online. This document specifies the operations that clients can call. For each operation, you specify and configure the flow of connections and steps, such as data mapper or filter steps, that executes that operation. A simple integration has one flow while an API provider integration has a flow for each operation.

The Fuse Online dashboard lets you monitor and manage integrations. You can see which integrations are running, and you can start, stop, and edit integrations.

1.2. WHO FUSE ONLINE IS FOR

Fuse Online is for business experts in, for example, finance, human resources, or marketing, who do not want to write code in order to share data between two different applications. Their use of a variety of software-as-a-service (SaaS) applications gives them an understanding of business requirements, workflows, and relevant data.

As a business user, you can use Fuse Online to:

- Capture tweets that mention your company, filter them, and create new contacts in your Salesforce environment when the tweet is from an unknown source.
- Identify Salesforce lead updates and then execute a SQL stored procedure to keep your related database up to date.
- Subscribe for orders received by an AMQ broker and then operate on those orders with a custom API.
- Obtain data from an Amazon S3 bucket and add it to a Dropbox folder.

These are just a few examples of what a business user can do without writing code.

1.3. BENEFITS OF USING FUSE ONLINE

Fuse Online has a number of benefits:

- Integrate data from different applications or services without writing code.
- Run the integration on OpenShift Online in the public cloud or on OpenShift Container Platform on site.
- Use the visual data mapper to map data fields in one application to data fields in another application.
- Leverage all the benefits of open source software. You can extend features, and customize interfaces. If Fuse Online does not provide a connector for an application or service that you want to integrate then a developer can create the connector that you need.

1.4. DESCRIPTIONS OF FUSE ONLINE CONSTRUCTS

To use Fuse Online, you create an integration by working with connectors, connections, actions, steps, and flows. It is helpful to have a basic understanding each of these constructs.

Integrations

In Fuse Online, there are simple integrations and API provider integrations.

A simple integration is a set of ordered steps that Fuse Online executes. This set includes:

- A step that connects to an application to start the integration. This connection provides the initial data that the integration operates on. A subsequent connection can provide additional data.
- A step that connects to an application to complete the integration. This connection receives any data that was output from previous steps and finishes the integration.

- Optional additional steps that connect to applications between the start and finish connections. Depending on the position of the additional connection in the sequence of integration steps, an additional connection can do any or all of the following:
    - Provide additional data for the integration to operate on
    - Process the integration data
    - Output processing results to the integration
- Optional steps that operate on data between connections to applications. Typically, there is a step that maps data fields from the previous connection to data fields that the next connection uses.

An API provider integration publishes a REST API service for which you provided an OpenAPI schema. A call from a REST API client triggers execution of an API provider integration. The call can invoke any operation that the REST API implements. While a simple integration has one flow of execution, an API provider integration has a flow for each operation. Each operation flow connects to the applications and processes the data according to the steps that you added to that operation's flow when you created the integration. Each operation flow ends by returning a response that you specify to the client whose call triggered execution of the integration.

Connectors

Fuse Online provides a set of connectors. A connector represents a specific application that you want to obtain data from or send data to. Each connector is a template for creating a connection to that specific application. For example, you use the Salesforce connector to create a connection to Salesforce.

An application that you want to connect to might use the OAuth protocol to authenticate users. In this case, you register your Fuse Online environment as a client that can access that application. The registration is associated with the connector for that application. You need to register a particular Fuse Online environment only once with each application that uses OAuth. The registration extends to each connection that you create from that connector.

If Fuse Online does not provide a connector that you need, a developer can create the required connector.

Connections

Before you can create an integration, you must create a connection to each application or service that you want to obtain data from or send data to. To create a connection, you select a connector and add configuration information. For example, to connect to an AMQ broker in an integration, you create a connection by selecting the AMQ connector, and then following prompts to identify the broker to connect to and the account to use for the connection.

A connection is one specific instance of the connector that it is created from. You can create any number of connections from one connector. For example, you can use the AMQ connector to create three AMQ connections where each connection accesses a different broker.

To create a simple integration, you select a connection to start the integration, a connection to end the integration, and optionally one or more connections for accessing additional applications. To create an API provider integration, you can add one or more connections to each operation flow. Any number of integrations and operation flows can use the same connection. A particular integration or flow can use the same connection more than once.

For details, see About connecting to applications .

Actions

In an integration, each connection performs exactly one action. As you create an integration, you choose a connection to add to the flow and then you choose the action that the connection performs. For example, when you add a Salesforce connection to a flow, you choose from a set of actions that includes, but is not limited to, creating a Salesforce account, updating a Salesforce account, and searching Salesforce.

Some actions require additional configuration and Fuse Online prompts you for this information.

Steps

A simple integration is a set of ordered steps. In an API provider integration, each operation flow is a set of ordered steps.

Each step operates on data. Some steps operate on data while connected to an application or service outside Fuse Online. These steps are connections. Between connections, there can be other steps that operate on data in Fuse Online. Typically, the set of steps includes a step that maps data fields used in a previous connection to data fields that are used in the next connection in the flow. Except for the start connection in a simple integration, each step operates on data it receives from the previous steps.

To operate on data between connections, Fuse Online provides steps for:

- Mapping data fields in one application to data fields in another application.
- Filtering data so that the integration continues only when the data being processed meets criteria that you define.
- Spliting a collection of records into individual records so that Fuse Online executes subsequent steps iteratively, once for each record.
- Aggregating individual records into a collection so that Fuse Online executes subsequent steps once for the collection.
- Generating equivalent, consistent output by inserting data into a Freemarker, Mustache, or Velocity template.
- Logging information in addition to the default logging that Fuse Online automatically provides.

To operate on data between connections in a way that is not built into Fuse Online, you can upload an extension that provides a custom step. See Developing extensions.

Flow

A simple integration has one flow, which is the set of steps that it executes. An API provider integration has a flow for each operation that the REST API defines. Each operation’s flow is the set of steps that processes a call that invokes that operation.

In a flow, each step can operate on the data that is output from the previous steps. To determine the steps that you need in a flow, see Planning integrations.

CHAPTER 2. HOW TO GET READY TO CREATE INTEGRATIONS

Some planning and an understanding of the workflow for creating an integration can help you create integrations that meet your needs. The following topics provide information for getting ready to create integrations.

- Section 2.1, “Considerations for planning your integrations”
- Section 2.2, “General workflow for creating a simple integration”
- Section 2.3, “Example workflow for creating a Salesforce to database simple integration”

2.1. CONSIDERATIONS FOR PLANNING YOUR INTEGRATIONS

Consider the following questions before you create an integration.

How do you want to trigger execution of the integration?

- Do you want to set a timer to trigger execution at intervals that you specify?
- Do you want to send an HTTP request?
- Do you want to connect to an application to obtain data from?
    - In that application, what triggers the action that obtains the data? For example, an integration that starts by obtaining data from Twitter might trigger on a Twitter mention.
    - What are the data fields of interest?
    - What credentials does Fuse Online use to access this application?
- Do you want to publish a REST API service so that a client can invoke a REST API call that triggers execution of the flow for an operation?
    - Is the OpenAPI schema for the service already defined?
    - If not, what operations will the service define?

To finish a simple integration:

- Which application receives the data?
- In that application, what action does the integration perform?
- What are the data fields of interest?
- What credentials does Fuse Online use to access this application?

In a flow’s set of steps:

- Do you need to access any other applications? For any other applications that need to be accessed:
    - Which application does the flow need to connect to?
    - What action should the connection perform?

    - What are the data fields of interest?
    - What credentials should the connection use to connect to this application?

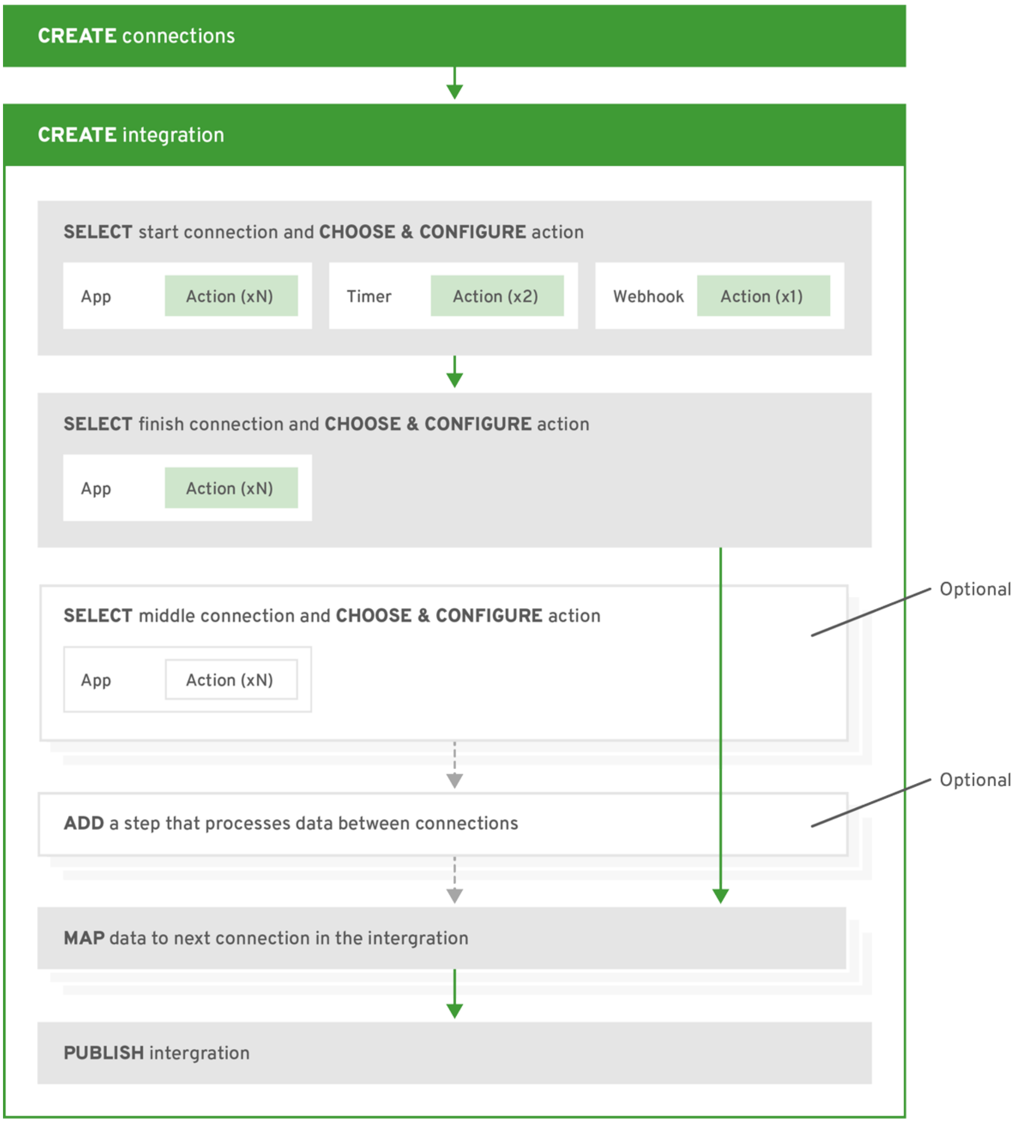
- Does the flow need to operate on the data between connections? For example:
    - Should the flow filter the data it operates on?
    - Do field names differ between source and target applications? If they do then data mapping is required.
    - Does the flow operate on a collection? If it does, can the flow use the data mapper to process the collection or does the flow need to split a collection into individual records? Does the flow need to aggregate records into a collection?
    - Would a template be helpful for outputting data in a consistent form?
    - Do you want to send information about messages being processed to the integration’s log?
    - Does the flow need to operate on the data in some customized way?

2.2. GENERAL WORKFLOW FOR CREATING A SIMPLE INTEGRATION

After you log in to the Fuse Online console, you can start creating connections to the applications that you want to integrate. For each application that you want to integrate and that uses the OAuth protocol, register Fuse Online as a client of that application. Applications that you must register with include:

- Dropbox
- Google applications (Gmail, Calendar, Sheets)
- Salesforce
- SAP Concur
- Twitter

With registration in place for those applications, the workflow for creating a simple integration looks like this:


CREATE connections
CREATE integration
SELECT start connection and CHOOSE & CONFIGURE action
App
Action (xN)
Timer
Action (x2)
Webhook
Action (x1)
SELECT finish connection and CHOOSE & CONFIGURE action
App
Action (xN)
SELECT middle connection and CHOOSE & CONFIGURE action
App
Action (xN)
Optional
ADD a step that processes data between connections
Optional
MAP data to next connection in the intergration
PUBLISH intergration

FUSE_19_0319

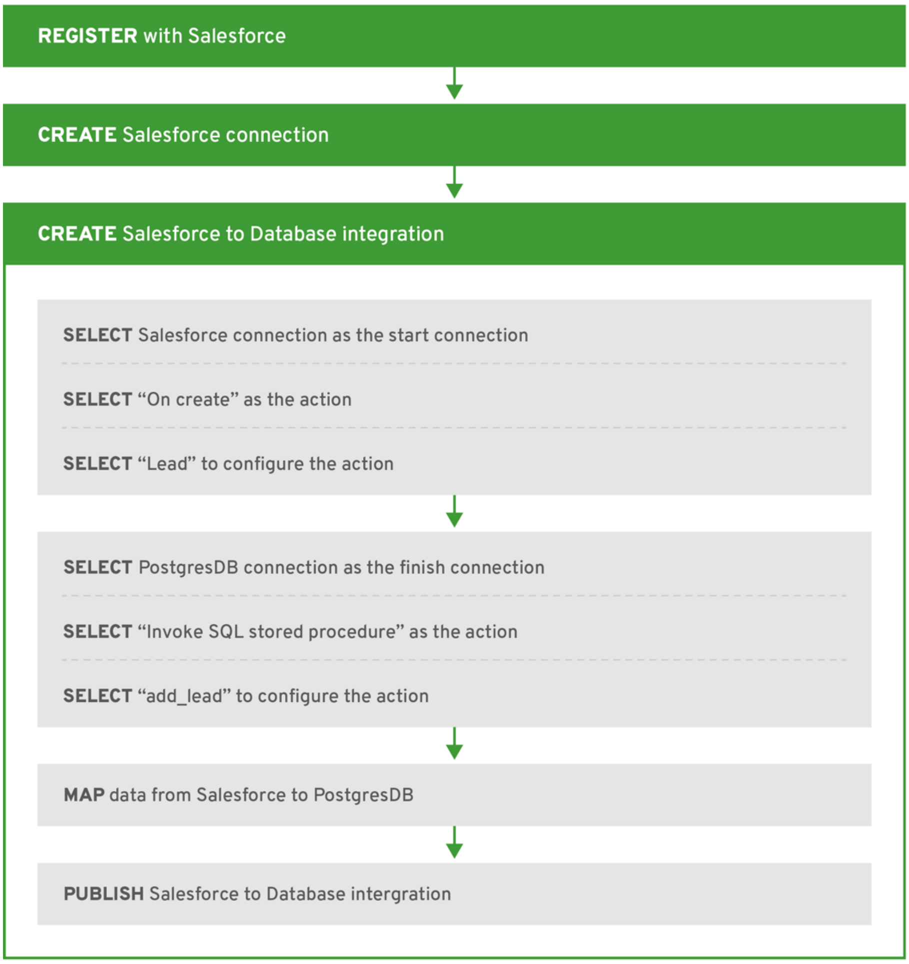
Additional resource

Workflow for creating API provider integrations.

2.3. EXAMPLE WORKFLOW FOR CREATING A SALESFORCE TO DATABASE SIMPLE INTEGRATION

The best way to understand the workflow for using Fuse Online to create a simple integration is to create the sample integrations by following the instructions in the sample integration tutorials.

The following diagram shows the workflow for creating the sample Salesforce to Database integration.


REGISTER with Salesforce
CREATE Salesforce connection
CREATE Salesforce to Database integration
SELECT Salesforce connection as the start connection
SELECT “On create” as the action
SELECT “Lead” to configure the action
SELECT PostgresDB connection as the finish connection
SELECT “Invoke SQL stored procedure” as the action
SELECT “add_lead” to configure the action
MAP data from Salesforce to PostgresDB
PUBLISH Salesforce to Database intergration

FUSE_478143_1118

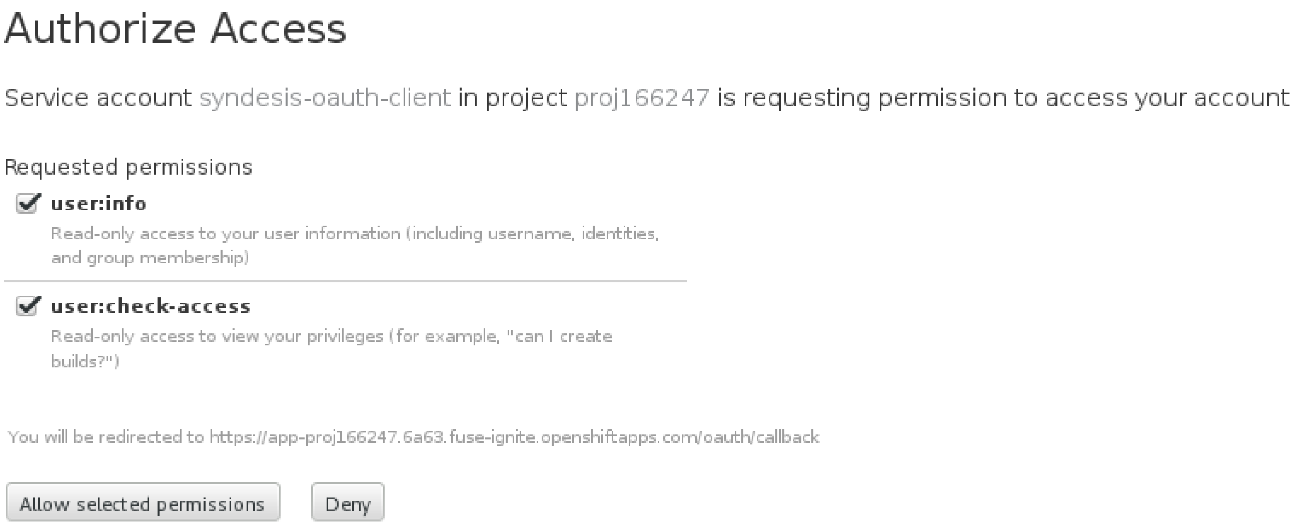
After you publish an integration, the Fuse Online dashboard displays **Running** next to the integration name when the integration is ready to be executed.

Additional resource

Configuring and publishing an API provider quickstart integration .

CHAPTER 3. WHAT TO EXPECT WHEN USING FUSE ONLINE FOR THE FIRST TIME

To enable access to Fuse Online on OpenShift Online, Red Hat provides a link. Clicking this link displays the **Red Hat OpenShift Online Log In**page, which prompts you to log in by using your Red Hat account. Logging in prompts you to authorize access from Fuse Online to your OpenShift Online account:



Authorize Access

Service account syndesis-oauth-client in project proj166247 is requesting permission to access your account

Requested permissions

☑ **user:info**
Read-only access to your user information (including username, identities, and group membership)

☑ **user:check-access**
Read-only access to view your privileges (for example, "can I create builds?")

You will be redirected to https://app-proj166247.6a63.fuse-ignite.openshiftapps.com/oauth/callback

Click **Allow selected permissions**. You need to do this only once. The next time you click the Fuse Online access link that is in your "Welcome to the Red Hat Fuse Online Evaluation" email message, Fuse Online immediately appears.

To use Fuse Online on OpenShift Container Platform, follow the installation instructions, which are at the end of Integrating Applications with Fuse Online.

Red Hat supports using Fuse Online in the following browsers:

- Chrome
- Firefox
- Microsoft Edge

CHAPTER 4. ABOUT CONNECTIONS TO APPLICATIONS THAT YOU WANT TO INTEGRATE

To connect to applications that you want to integrate, the main steps are:

1. Create a connection to each application or service that you want to integrate.

2. Create an integration that has a connection to each application that you want to integrate.

The procedure for creating a connection varies for each application or service. The details for creating each kind of connection and configuring it for a particular integration are in Connecting Fuse Online to Applications and Services.

The following topics provide general information about connections:

- Section 4.1, “About creating connections from Fuse Online to applications”
- Section 4.2, “General procedure for obtaining authorization”
- Section 4.3, “About connection validation”
- Section 4.4, “About adding connections to integrations”
- Section 4.5, “How to view and edit connection information”
- Section 4.6, “About creating a connection from a custom connector”

4.1. ABOUT CREATING CONNECTIONS FROM FUSE ONLINE TO APPLICATIONS

To create a connection, you select the connector for the application that you want to connect to and then enter values in input fields to configure a connection to that application. The configuration details that you need to provide vary for each application. After configuring the connection, you give it a name that helps you distinguish it from any other connections to the same application. Optionally, you can specify a description of the connection.

You can use the same connector to create any number of connections to that application. For example, you might use the AMQ connector to create three different connections. Each AMQ connection could specify a different broker.

For examples, see:

- Create AMQ connections
- Create HTTP and HTTPS connections
- Create Slack connections

4.2. GENERAL PROCEDURE FOR OBTAINING AUTHORIZATION

In an integration, you might want to connect to an application that uses the OAuth protocol to authenticate access requests. To do this, you must register your installation of Fuse Online for access to that application. Registration authorizes all connections from your Fuse Online installation to a given application. For example, if you register your Fuse Online installation with Salesforce, all connections from your Fuse Online installation to Salesforce use the same Salesforce client ID and the same Salesforce client secret that registration provided.

In each Fuse Online environment, for each application that uses OAuth, only one registration of Fuse Online as a client is required. This registration lets you create multiple connections and each connection can use different user credentials.

While the specific steps vary for each OAuth application that you want to connect to, registration always provides your Fuse Online environment with a client ID and a client secret. Some applications use other labels for the client ID and client secret. For example, Salesforce generates a consumer key and a consumer secret.

For some OAuth applications, Fuse Online provides an entry in its **Settings** page for adding the client ID and client secret that registration provides. To see which applications this applies to, in the left panel of Fuse Online, click **Settings**.

Prerequisite

In the Fuse Online **Settings** page, there is an entry for the application that uses the OAuth protocol to authorize access.

Procedure overview

1. In the Fuse Online **OAuth Application Management** page, in the entry for the application with which you want to register Fuse Online, click **Register** to display the client ID and client secret fields.

2. Near the top of the **OAuth Application Management** page, where you see **During registration, enter this callback URL:**, copy that URL to the clipboard.

3. In another browser tab, go to the web site for the application that you want to register with and perform the steps that are required to obtain a client ID and secret. One of these steps requires you to enter the callback URL for your Fuse Online environment. Paste the URL that you copied to the clipboard in the second step.

4. In Fuse Online, on the **Settings** page, paste the client ID and client secret and save the settings.

Additional resources

- Examples of registering applications that have entries in the **Settings** page:
  - Registering Fuse Online as a Salesforce client
  - Registering Fuse Online as a Twitter client
- Example of registering with an application that does not have an entry in the Fuse Online **Settings** page: Registering Fuse Online as a Dropbox client
- Information about using custom connectors that let you access applications that use the OAuth protocol: Creating connections from custom connectors .

4.3. ABOUT CONNECTION VALIDATION

After obtaining authorization for Fuse Online to access an application that uses OAuth, you can create one or more connections to that application. When you create a connection to an OAuth application, Fuse Online validates it to confirm that authorization is in place. At any time, you can validate the connection again to ensure that authorization is still in place.

Some OAuth applications grant access tokens that have an expiration date. If the access token expires, you can reconnect to the application to obtain a new access token.

To validate a connection that uses OAuth or to obtain a new access token for an OAuth application:

1. In the left panel, click **Connections**.
2. Click the connection that you want to validate or for which you want to obtain a new access token.
3. In the connection's details page, click **Validate** or click **Reconnect**.

If validation or reconnection fails, then check with the application/service provider to determine if the application's OAuth keys, IDs, tokens, or secrets are still valid. It is possible that an item has expired or been revoked.

If you find that an OAuth item is invalid, has expired, or been revoked, obtain new values and paste them into the Fuse Online settings for the application. See the instructions in Connecting Fuse Online to Applications and Services for registering the application whose connection did not validate. With the updated settings in place, follow the instructions above to try to validate the updated connection. If validation is successful, and there is a running integration that is using this connection, restart the integration. To restart an integration, stop it and then start it.

If validation fails and reconnection fails but everything appears to be valid at the service provider, then try reregistering your Fuse Online environment with the application and then recreate the connection. Fuse Online validates the connection when you recreate it. If you recreate the connection, and there is an integration that is using the connection, then you must edit the integration to delete the old connection and add the new connection. If the integration is running, then you must stop it and restart it.

4.4. ABOUT ADDING CONNECTIONS TO INTEGRATIONS

When you add a connection to a simple integration or to an operation flow, Fuse Online displays a list of the actions that the connection can perform when it connects to the application. You must select exactly one action. In a running integration, each connection performs only the action you choose. For example, when you add a Twitter connection as an integration's start connection, you might choose the **Mention** action, which monitors Twitter for tweets that mention your Twitter handle.

Selection of some actions prompts you to specify one or more parameters, which configure the action. For example, if you add a Salesforce connection to an integration and choose the **On create** action then you must indicate the type of object whose creation you are interested in, such as a lead or a contact.

4.5. HOW TO VIEW AND EDIT CONNECTION INFORMATION

After you create a connection, Fuse Online assigns an internal identifier to the connection. This identifier does not change. You can change the connection's name, description, or configuration values and Fuse Online recognizes it as the same connection.

There are two ways to view and edit information about a connection:

- In the left panel, click **Connections** and then click any connection to view its details.
- In the left panel, click **Integrations** and then click any integration to view its summary page. In the integration's flow diagram:

- For a simple integration, click a connection icon to view that connection’s details.
- For an API provider integration, click ↩ to display the integration’s operation list. Click the operation whose flow contains the connection that you want to view details for.

On the **Connection Details** page, for the connection that you want to edit, click ✏ next to a field to edit that field. Or, for some connections, below the configuration fields, click **Edit** to change configuration values. If you change any values, be sure to click **Save**.

If you update a connection that is used in an integration that is running, you must republish the integration.

For connections to applications that use the OAuth protocol to authorize access, you cannot change the login credentials that the connection uses. To connect to the application and use different login credentials, you must create a new connection.

4.6. ABOUT CREATING A CONNECTION FROM A CUSTOM CONNECTOR

After you upload an extension that defines a custom connector, the custom connector is available for use. You use custom connectors to create connections in the same way that you use Fuse Online-provided connectors to create connections.

A custom connector might be for an application that uses the OAuth protocol. Before you create a connection from this kind of connector, you must register your Fuse Online environment for access to the application that the connector is for. You do this in the interface for the application that the connector is for. The details for how to register your Fuse Online environment vary for each application.

For example, suppose the custom connector is for creating connections to Yammer. You would need to register your Fuse Online environment by creating a new application within Yammer. Registration provides a Yammer client ID for Fuse Online and a Yammer client secret value for Fuse Online. A connection from your Fuse Online environment to Yammer must provide these two values.

Note that an application might use different names for these values, such as consumer ID or consumer secret.

After you register your Fuse Online environment, you can create a connection to the application. When you configure the connection, there should be parameters for entering the client ID and the client secret. If these parameters are not available, you need to talk with the extension developer and ask for an updated extension that lets you specify the client ID and client secret.

CHAPTER 5. CREATING INTEGRATIONS

After some planning and preparation, you are ready to create an integration. In the Fuse Online web interface, when you click **Create Integration**, Fuse Online guides you through the procedure to create an integration.

Prerequisites

- Considerations for planning your integrations
- According to the kind of integration that you want to create:
  - An understanding of the general workflow for creating a simple integration
  - An understanding of the general workflow for creating an API provider integration

The following topics provide information and instructions for creating an integration:

- Section 5.1, “Preparation for creating an integration”
- Section 5.2, “Procedure for creating a simple integration”
- Section 5.3, “How to process a collection in a flow”
- Section 5.4, “About adding steps between connections”
- Section 5.5, “Adding a data mapping step”
- Section 5.6, “Adding a basic filter step”
- Section 5.7, “Adding an advanced filter step”
- Section 5.8, “Adding a template step”
- Section 5.9, “Adding a custom step”

5.1. PREPARATION FOR CREATING AN INTEGRATION

Preparation for creating an integration starts with answers to the questions listed in Considerations for planning your integrations. After you have a plan for the integration, you need to do the following before you can create the integration:

1. Determine whether an application that you want to connect to uses the OAuth protocol. For each application that uses OAuth, register Fuse Online as a client that is authorized to access that application. Applications that use the OAuth protocol include:
   - Dropbox
   - Google applications (Gmail, Calendar, Sheets)
   - Salesforce
   - SAP Concur
   - Twitter

2. Determine whether an application that you want to connect to uses HTTP basic authentication. For each application that does, identify the user name and password for accessing that application. You need to provide this information when you create the connection.

3. For each application that you want to integrate, create a connection.

Additional resources

- General procedure for obtaining authorization

- About creating connections

5.2. PROCEDURE FOR CREATING A SIMPLE INTEGRATION

Fuse Online guides you through the procedure for creating a simple integration. It prompts you to choose the start connection, the finish connection, optional middle connections, and other steps. When your integration is complete, you can publish it so that it is running or you can save it for publication at a later time.

To learn about the procedure for creating an API provider integration, see Section 6.3, "Creating an API provider integration".

Prerequisites

- You have a plan for what the steps in the integration will be.

- You created a connection to each application or service that you want to connect to in this integration.

Procedure

1. In the left panel in Fuse Online, click **Integrations**.

2. In the upper right, click **Create Integration**.

3. Choose and configure the start connection:

    a. On the **Choose a Start Connection** page, click the connection that you want to use to start the integration. When this integration is running, Fuse Online will connect to this application and obtain data that you want the integration to operate on.

    b. On the **Choose an Action** page, click the action you want this connection to perform. The available actions vary for each connection.

    c. On the page for configuring the action, enter values in the fields.

    d. Optionally, if the connection requires data type specification, Fuse Online prompts you to click **Next** to specify the input and/or output type of the action.

    e. Click **Done** to add the start connection.

    As an alternative to connecting to an application, a start connection can be a timer that triggers integration execution at intervals that you specify or it can be a webhook that accepts HTTP requests.

4. Choose and configure the finish connection:

   a. On the **Choose a Finish Connection** page, click the connection you want to use to complete the integration. When this integration is running, Fuse Online will connect to this application with the data that the integration has been operating on.

   b. On the **Choose an Action** page, click the action you want this connection to perform. The available actions vary for each connection.

   c. On the page for configuring the action, enter values in the fields.

   d. Optionally, if the connection requires data type specification, Fuse Online prompts you to click **Next** to specify the input and/or output type of the action.

   e. Click **Next** to add the finish connection.

   As an alternative to connecting to an application, a finish connection can send information to the integration’s log about the messages that the integration processed. To do this, select **Log** when Fuse Online prompts you to choose the finish connection.

5. Optionally, add one or more connections between the start connection and the finish connection. For each connection, choose its action and enter any required configuration details.

6. Optionally, add one or more steps that operate on integration data between connections. See About adding steps between connections .

7. In the flow visualization panel on the left, look for any ⚠ icons. These warnings indicate that a data mapper step is needed before this connection. Add the required data mapper steps.

8. When the integration contains all needed steps, click **Save** or **Publish** according to whether you want to start running the integration.

9. In the **Integration Name** field, enter a name that distinguishes this integration from any other integrations.

10. In the **Description** field, enter a description, for example, you can indicate what this integration does.

11. If you are ready to start running the integration, in the upper right, click **Publish**. Otherwise, click **Save**.

Results

In the Fuse Online **Integrations** page, you can see your new integration in the list of integrations. If you published the integration, then you can see that Fuse Online is in the process of publishing it. It may take a few moments for the status of the integration to become **Running**. If you saved the integration, then **Stopped** appears on the integration’s entry.

Click the integration’s entry to see a summary of the integration, including its version history, logs for any executions, and aggregate execution metrics.

5.3. HOW TO PROCESS A COLLECTION IN A FLOW

Sometimes, a connection returns a collection, which contains multiple values that are all the same type. When a connection returns a collection, the flow can operate on the collection in a number of ways, including:

- Execute each step once for the collection.

- Execute each step once for each element in the collection.
- Execute some steps once for the collection and execute other steps once for each element in the collection.

To decide how to operate on a collection in a flow, you need to know which applications the flow connects to, whether they can handle collections, and what you want the flow to accomplish. You can then use the information in the following topics to add steps to a flow that processes a collection:

- Section 5.3.1, “About processing collections”
- Section 5.3.2, “Using the data mapper to process collections”
- Section 5.3.3, “Adding a split step”
- Section 5.3.4, “Adding an aggregate step”
- Section 5.3.5, “Example of processing a collection in a flow”

5.3.1. About processing collections

The easiest way for a flow to process a collection is to use the data mapper to map fields that are in a source collection to fields that are in a target collection. For many flows, this is all that is required. For example, a flow might obtain a collection of employee records from a database and then insert those records into a spreadsheet. Between the database connection and the Google Sheets connection, a data mapper step maps the database fields to the Google Sheets fields. Since both the source and the target are collections, when Fuse Online executes the flow, it calls the Google Sheets connection once. In that call, Fuse Online iterates through the records and correctly populates the spreadsheet.

In some flows, you might need to split a collection into individual objects. For example, consider a flow that connects to a database and obtains a collection of employees who will lose allotted time off if they do not use it before a certain date. The flow then needs to send an email notification to each of these employees. In this flow, you would add a split step after the database connection. You would then add a data mapper step that maps the source fields for an employee record to target fields in a Gmail connection that sends a message. When Fuse Online executes the flow, it executes the data mapper step and the Gmail connection once for each employee.

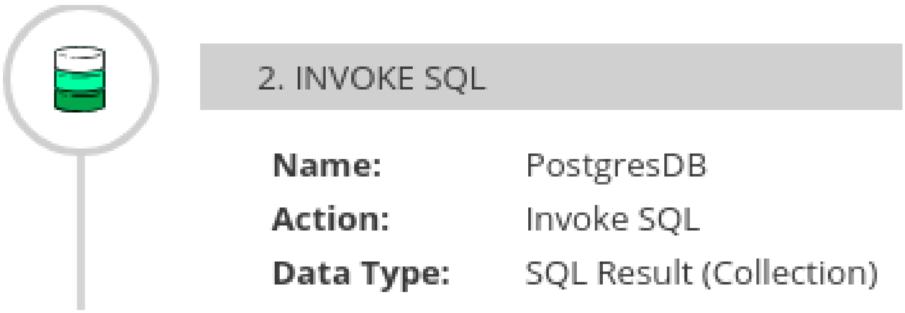
Sometimes, after you split a collection in a flow, and after the flow executes some steps once for each element that was in the collection, you want the flow to operate on the collection again. Consider the example in the previous paragraph. Suppose that after a Gmail connection sends a message to each employee, you want to add a list of the employees who were notified to a spreadsheet. In this scenario, after the Gmail connection, add an aggregate step to create a collection of employee names. Then add a data mapper step that maps fields in the source collection to fields in the target Google Sheets connection. When Fuse Online executes the flow, it executes the new data mapper step and the Google Sheets connection once for the collection.

These are the most common scenarios for processing a collection in a flow. However, much more complex processing is also possible. For example, when the elements in a collection are themselves collections, you can nest split and aggregate steps inside other split and aggregate steps.

5.3.2. Using the data mapper to process collections

In a flow, when a step outputs a collection and when a subsequent connection that is in the flow expects a collection as the input, you can use the data mapper to specify how you want the flow to process the collection.

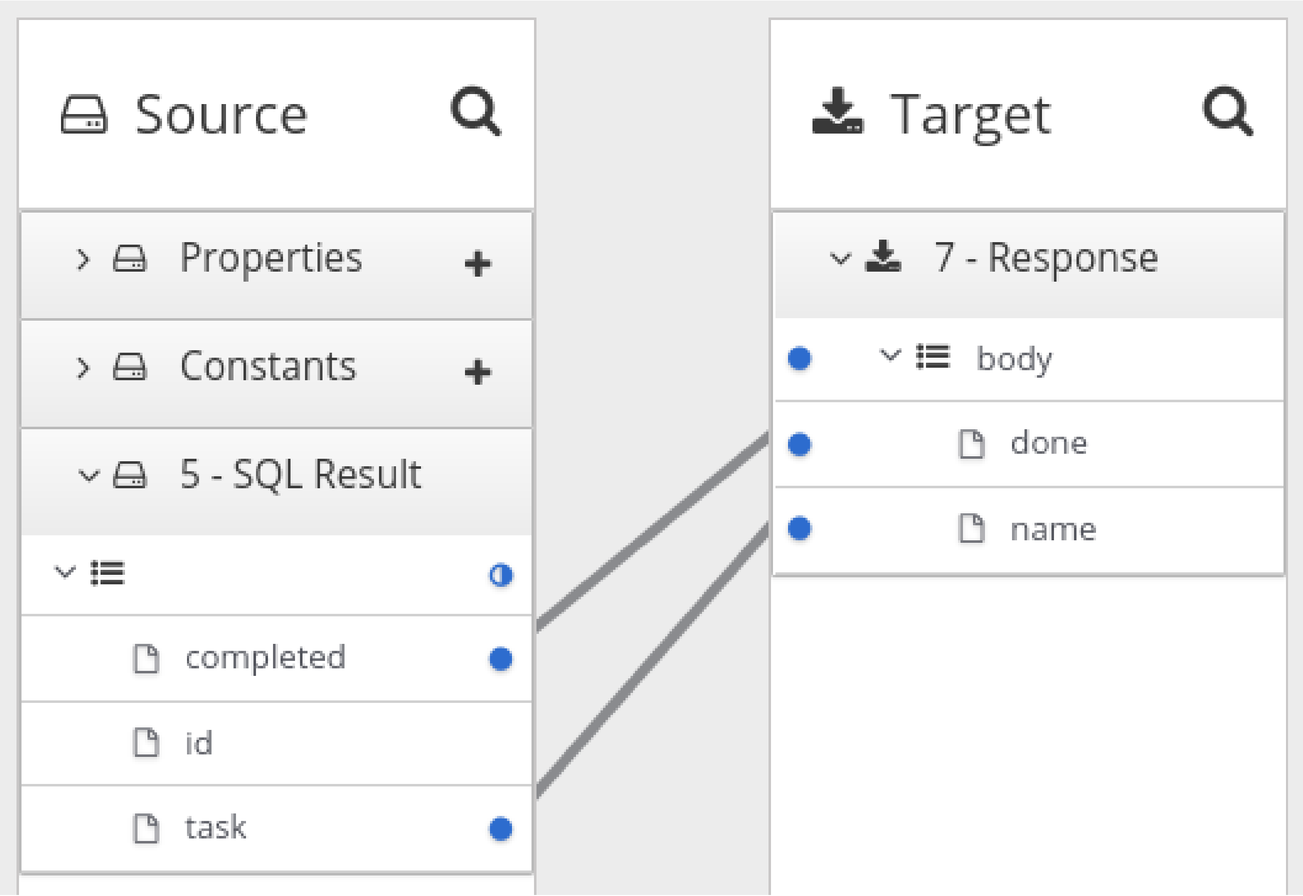
When a step outputs a collection, the flow visualization panel displays **Collection** in the details about the step. For example:



2. INVOKE SQL

Name: PostgresDB
Action: Invoke SQL
Data Type: SQL Result (Collection)

Add a data mapper step after the step that provides the collection and before the step that needs the mappings. Exactly where in the flow this data mapper step needs to be depends on the other steps in the flow. The following image shows mappings from source collection fields to target collection fields:



In the source and target panels, the data mapper displays ☰ to indicate a collection. When a source collection or a target collection contain only primitive types, the data mapper does not display collection fields because there is no need to. You can map from/to the collection itself.

When a collection contains more than one kind of primitive type or when it contains at least one complex type then the data mapper displays the collection's child fields. You can map from/to each field.

When Fuse Online executes the flow, it iterates over the source collection elements to populate the target collection elements. If you map one or more source collection fields to a target collection or to target collection fields, the target collection elements contain values for only the mapped fields.

If you map a source collection or a field in a source collection to a target field that is not in a collection, then when Fuse Online executes the flow, it assigns the value from only the last element in the source collection. Any other elements in the collection are ignored in that mapping step. However, any subsequent mapping steps can access all elements in the source collection.

When a connection returns a collection that is defined in a JSON or Java document, the data mapper can usually process the source document as a collection.

5.3.3. Adding a split step

During execution of a flow, when a connection returns a collection of objects, Fuse Online executes subsequent steps once for the collection. If you want to execute subsequent steps once for each object that is in the collection, add a split step. For example, a Google Sheets connection returns a collection of row objects. To execute subsequent steps once for each row, add a split step after the Google Sheets connection.

Ensure that the input to a split step is always a collection. If a split step gets a source document that is not a collection type, the step splits the input at each space. For example, Fuse Online splits “Hello world!” input into two elements: “Hello” and “world!”, and passes those two elements to the next step in the flow. In particular, XML data is not a collection type.

Prerequisites

- You are creating or editing a flow.
- The flow already has all the connections that it requires.
- In the flow visualization, the connection that obtains the source data indicates that the data is a **(Collection)**.

Procedure

1. In the visualization panel on the left, where you want to add a split step to the flow, click the ➕.
2. Click **Split**. This step does not require any configuration.

Additional information

Typically, you want to add any split steps and aggregate steps before you add data mapper steps. This is because whether the data is a collection or individual objects affects the mappings. If you add a data mapper step and then add a split step, you usually need to redo the mappings. Likewise, if you remove a split or aggregate step, then you would need to redo any mappings.

5.3.4. Adding an aggregate step

In a flow, add an aggregate step where you want Fuse Online to create a collection from individual objects. During execution, after an aggregate step, instead of executing subsequent steps once for each object, Fuse Online executes subsequent steps once for the collection.

When deciding whether to add an aggregate step to a flow, consider the connections in the flow. After a split step, for each subsequent connection, Syndesis connects to that application once for each element in the flow’s data. For some connections, it might be preferable to connect once rather than multiple times.

Prerequisites

- You are creating or editing a flow.
- The flow already has all the connections that it requires.
- A previous step split a collection into individual objects.

Procedure

1. In the flow visualization panel on the left, where you want to add an aggregate step to the flow, click the ➕ .
2. Click **Aggregate**. This step does not require any configuration.

Additional information

Typically, you want to add any split and aggregate steps before you add data mapper steps. This is because whether the data is a collection or individual objects affects the mappings. If you add a data mapper step and then add an aggregate step, you usually need to redo the mappings. Likewise, if you remove an aggregate step, then you would need to redo any mappings.

5.3.5. Example of processing a collection in a flow

This simple integration obtains a collection of tasks from the sample database provided with Fuse Online. The flow splits the collection into individual task objects and then filters these objects to find the tasks that have been done. The flow then aggregates the completed tasks in a collection, maps the fields in that collection to fields in a spreadsheet, and finishes by adding a list of completed tasks to a spreadsheet.

The procedure below provides instructions for creating this simple integration.

Prerequisites

- You created a Google Sheets connection.
- In the account that the Google Sheets connection accesses, there is a spreadsheet for receiving the database records.

Procedure

1. Click **Create Integration**.
2. Add the start connection:
   a. On the **Choose a Start Connection** page, click **PostgresDB**.
   b. On the **Choose an Action** page, click **Periodic SQL Invocation**.
   c. In the **SQL Statement** field, enter **select * from todo** and click **Next**.

   This connection returns a collection of task objects.
3. Add the finish connection:
   a. On the **Choose a Finish Connection** page, click your Google Sheets connection.

b. On the **Choose an Action** page, click **Append values to a sheet**

c. In the **SpreadsheetId** field, enter the ID of the spreadsheet to add the list of tasks to.

d. In the **Range** field, enter **A:B** as the target columns that you want to append values to. The first column, **A**, is for the task IDs. The second column, **B**, is for the task names.

e. Accept the defaults for **Major Dimension** and for **Value Input Option**, and click **Next**.

The Google Sheets connection finishes the flow by adding each element in a collection to a spreadsheet.

4. Add a split step to the flow:

    a. In the visualization panel, click the plus sign.

    b. Click **Split**.

    After the flow executes the split step, the result is a set of individual task objects. Fuse Online executes the subsequent steps in the flow once for each individual task object.

5. Add a filter step to the flow:

    a. In the visualization panel, after the split step, click the plus sign.

    b. Click **Basic Filter** and configure the filter as follows:

        i. Click in the first field and select **completed**, which is the name of the field that contains the data that you want to evaluate.

        ii. In the second field, select **equals** as the condition that the **completed** field value must satisfy.

        iii. In the third field, specify **1** as the value that must be in the **completed** field. **1** indicates that the task has been completed.

    c. Click **Done**.

    During execution, the flow executes the filter step once for each task object. The result is a set of individual, completed task objects.

6. Add an aggregate step to the flow:

    a. In the visualization panel, after the filter step, click the plus sign.

    b. Click **Aggregate**.

    Now the result set contains one collection, which contains an element for each completed task.

7. Add a data mapper step to the flow:

    a. In the visualization panel, after the aggregate step, click the plus sign.

    b. Click **Data Mapper** and map the following fields from the SQL result source collection to the Google Sheets target collection:

        - **id** to **A**

        - **task** to **B**

   c. Click **Done**.

8. Click **Publish**.

Results

When the integration is running, it obtains tasks from the sample database every minute and then adds the completed tasks to the first sheet in the spreadsheet. The integration maps the task ID to the first column, **A**, and it maps the task name to the second column, **B**.

5.4. ABOUT ADDING STEPS BETWEEN CONNECTIONS

Although it is not a requirement, the recommendation is to add all needed connections to a flow and then, according to the processing that you want the flow to execute, add additional steps between connections. In a flow, each step operates on data obtained from the previous connection(s) and any previous steps. The resulting data is available to the next step in the flow.

Often, you must map data fields that are received from a connection to data fields that the next connection in the flow can operate on. After you add all connections to a flow, check the flow visualization panel on the left. For each connection that requires data mapping before it can operate on the input data, Fuse Online displays ⚠ . Click this icon to see **Data Type Mismatch: Add a data mapping step before this connection to resolve the difference.**

You can click the link in the message to display the **Configure Mapper** page in which you add and specify a data mapping step. However, the recommendation is to add other needed steps, and then add data mapper steps last.

5.5. ADDING A DATA MAPPING STEP

Almost all integrations require data mapping. A data mapper step maps data fields from the previous connection(s) and any other steps to data fields that the next connection in the flow can operate on. For example, suppose the integration data contains a **Name** field and the next connection in the flow has a **CustomerName** field. You need to map the source **Name** field to the target **CustomerName** field.

Prerequisite

You are creating or editing a flow.

Procedure

1. In the left panel, where you want to add a data mapper step, click the ➕ .
2. Click **Data Mapper** to display source and target fields in the data mapper canvas.

Next step

See Mapping data fields to the next connection in the flow .

5.6. ADDING A BASIC FILTER STEP

You can add a step to a flow to filter the data that the flow operates on. In a filter step, Fuse Online inspects the data and continues only if the content meets criteria that you define. For example, in a flow that obtains data from Twitter, you can specify that you want to continue execution by operating only on tweets that contain "Red Hat".

Prerequisites

- The flow contains all connections that it needs to.
- You are creating or editing a flow.

Procedure

1. In the left panel, where you want to add a filter step to the flow, click the ➕ .
2. Click **Basic Filter**.
3. On the **Configure Basic Filter Step** page, in the **Continue only if incoming data match**field, select one of the following options:
   - Accept the default that all defined rules must be satisfied.
   - Indicate that only one rule must be satisfied by selecting **ANY of the following**.
4. Define the filter rule:
   a. **The data you want to evaluate** Enter the name of the field that contains the content you want the filter to evaluate. For example, suppose the data coming in to the step consists of tweets that mention your Twitter handle. You want to continue execution only when the tweet contains certain content. The tweet is in a field named **text** so you enter or select **text** as the value in the first field.
   You can define the field value in the following ways:
      - Start typing. The data name field has a typeahead feature that provides a list of possible completions for you in a pop-up box. Select the correct one from the box.
      - Click in the field. A dropdown box appears with a list of available fields. Select the field of interest from the list.

   b. **Must meet this condition**: Select a condition from the dropdown box. The setting defaults to **Contains**. For execution to continue, the condition that you select in this field must be true for the value that you enter in the third field.

   c. **For this value**: Enter a value to filter on. For example, if you want to operate on mentions of a certain product from the Twitter feed, you would enter the product name here.
5. Optionally, click **+ Add another rule** and define another rule.
   You can delete a rule by clicking the trash can icon next to the entry.
6. When the filter step is complete, click **Done** to add it to the flow.

Additional resource

If you cannot define the filter you need in a basic filter step, see Adding an advanced filter step .

5.7. ADDING AN ADVANCED FILTER STEP

In a filter step, Fuse Online inspects the data and continues executing the flow only if the content meets criteria that you define. If the basic filter step does not let you define the exact filter that you need, then add an advanced filter step.

Prerequisites

- The flow contains all connections that it needs to.
- You are creating or editing a flow.

Procedure

1. In the left panel, where you want to add an advanced filter step to the flow, click the ➕ .
2. Click **Advanced Filter**.
3. In the edit box, use the Camel Simple Expression language to specify a filter expression. For example, the following expression evaluates to true when the message header's **type** field is set to **widget**:

   ```
   ${in.header.type} == 'widget'
   ```

   In the following example, the expression evaluates to true when the body of the message contains a **title** field:

   ```
   ${in.body.title}
   ```

4. Click **Done** to add the advanced filter step to the flow.

5.8. ADDING A TEMPLATE STEP

In a flow, a template step takes data from a source and inserts it into the format that is defined in a template that you upload to Fuse Online. The benefit of a template step is that it provides data output in a consistent format that you specify.

In the template, you define placeholders and specify static text. When you create the flow, you add a template step, map source fields to the template placeholders, and then map template content to the next step in the flow. When Fuse Online executes the flow, it inserts the values that are in the mapped source fields into an instance of the template and makes the result available to the next step in the flow.

If a flow includes a template step then it is most likely the only template step in that flow. However, more than one template step in a flow is allowed.

Fuse Online supports the following kinds of templates: Freemarker, Mustache, Velocity.

Prerequisites

- You must be creating or editing a flow.
- If you are creating a simple integration then it must already have its start and finish connections.

Procedure

1. In the visualization panel on the left, click the ➕ where you want to add a template step.

2. Click **Template**. The **Upload Template** page opens.

3. Specify the template type, which is Freemarker, Mustache, or Velocity.

4. To define the template, do one of the following:

   - Drag and drop a template file or a file that contains text that you want to modify to create a template, into the template editor.

   - Click **browse to upload**, navigate to a file, and upload it.

   - In the template editor, start typing to define a template.

5. In the template editor, ensure that the template is valid for use with Fuse Online. Examples of valid templates are after this procedure. Fuse Online displays ✖ to the left of a line that contains a syntax error. Hovering over a syntax error indicator displays hints about how to resolve the error.

6. Click **Done** to add the template step to the flow.
   If the **Done** button is not enabled then there is at least one syntax error that you must correct.

   Input to a template step must be in the form of a JSON object. Consequently, you must add a data mapping step before a template step.

7. To add a data mapper step before the template step:

   a. In the left panel, click the ➕ that is immediately before the template step that you just added.

   b. On the **Choose a Step** page, click **Data Mapper**.

   c. In the data mapper, map a source field to each template placeholder field.
      For example, using the example templates that are after this procedure, map a source field to each of these template fields:

      - **time**

      - **name**

      - **text**

   d. In the upper right, click **Done** to add the data mapper step to the flow.

   Output from a template step is always a JSON object. Consequently, you must add a data mapper step after a template step.

8. To add a data mapper step after the template step:

   a. In the left panel, click the ➕ that is immediately after the template step that you just added.

b. On the **Choose a Step** page, click **Data Mapper**.

c. In the data mapper, map the template's **message** field, which always contains the result of inserting source fields into the template, to a target field. For example, suppose that a Gmail connection is next in the flow and you want to send the result of the template step as the content of a Gmail message. To do this, you would map the **message** source field to the **text** target field.

d. In the upper right, click **Done**.

Examples of templates

Example of a Mustache template:

```
At {{time}}, {{name}} tweeted:
{{text}}
```

Freemarker and Velocity support this example template:

```
At ${time}, ${name} tweeted:
${text}
```

Velocity also supports syntax without braces, as shown in this example:

```
At $time, $name tweeted:
$text
```

A placeholder cannot contain a **.** (period).

Additional resources

For details about mapping fields, see Mapping integration data to fields for the next connection in the flow.

5.9. ADDING A CUSTOM STEP

If Fuse Online does not provide a step that you need in a flow, a developer can define one or more custom steps in an extension. A custom step operates on data between connections in a flow.

You add a custom step to a flow in the same way that you add a built-in step. For a simple integration, choose the start and finish connections, add other connections as needed and then add additional steps. For an API provider integration, select the operation whose flow executes the custom step, add connections as needed to the flow, and then add other steps. When you add a step, Fuse Online operates on the data it receives from the previous step(s) in the flow.

Prerequisites

- You uploaded the custom step extension to Fuse Online. See Making extensions available .
- You are creating or editing a flow.
- The flow already has all the connections that it requires.

Procedure

1. In the visualization panel on the left, where you want to add a custom step to the flow, click the ➕ .

2. Click the custom step that you want to add.
   The available steps includes any custom steps that are defined in extensions that were uploaded to your Fuse Online environment.

3. Respond to prompts for any information that is required to perform the step. This information varies for each custom step.

CHAPTER 6. CREATING AN INTEGRATION THAT IS TRIGGERED BY A REST API CALL

To trigger execution of an integration on demand, start the integration with a REST API service that you provide. Integrations that start this way are referred to as *API provider integrations*. An API provider integration allows REST API clients to invoke commands that trigger execution of the integration.

When Fuse Online publishes an API provider integration, any client with network access to the integration endpoints can trigger execution of the integration.

NOTE

If you are running Fuse Online on OpenShift Container Platform on-site, you can configure the Fuse Online server to enable Red Hat 3scale discovery of API provider integration APIs. By default, Fuse Online annotates an API provider integration's API service definition for use with 3scale but does not expose those APIs for automatic 3scale discovery. Without 3scale discovery, there is no access control. With 3scale discovery, you can set access policies, centralize control, and provide high availability for your API provider integration APIs. For more information, see the API Gateway documentation that is available from the Red Hat 3scale documentation page.

See also: Configuring Fuse Online to enable 3scale discovery of APIs .

The following topics provide information and instructions for creating API provider integrations:

- Section 6.1, "Benefit, overview, and workflow for creating API provider integrations"
- Section 6.2, "How OpenAPI operations relate to API provider integration flows"
- Section 6.3, "Creating an API provider integration"
- Section 6.4, "Creating the operation flows for an API provider integration"
- Section 6.5, "Configuring and publishing an example API provider quickstart integration"
- Section 6.6, "Testing the example API provider quickstart integration"

6.1. BENEFIT, OVERVIEW, AND WORKFLOW FOR CREATING API PROVIDER INTEGRATIONS

An API provider integration starts with a REST API service. This REST API service is defined by an OpenAPI 2.0 document that you provide when you create an API provider integration. After you publish an API provider integration, Fuse Online deploys the REST API service on OpenShift. The benefit of an API provider integration is that REST API clients can invoke calls that trigger execution of the integration.

Multiple execution flows

An API provider integration has multiple execution paths, referred to as flows. Each operation that the OpenAPI document defines has its own flow. In Fuse Online, for each operation that the OpenAPI document defines, you add connections and other steps to the execution flow for that operation. These steps process the data as required for the particular operation.

Example execution flow

For example, consider a human resources application that calls a REST API service that Fuse Online has made available. Suppose the call invokes the operation that adds a new employee. The operation flow that handles this call could:

- Connect to an application that creates an expense report for new employee equipment.
- Connect to a SQL database to add an internal ticket for setting up new equipment.
- Connect to Google mail to send a message to the new employee that provides orientation information.

Ways to trigger execution

There are many ways to call the REST APIs that trigger integration execution, including:

- A web browser page that takes data input and generates the call.
- An application that explicitly calls the REST APIs, such as the **curl** utility.
- Other APIs that call the REST API, for example, a webhook.

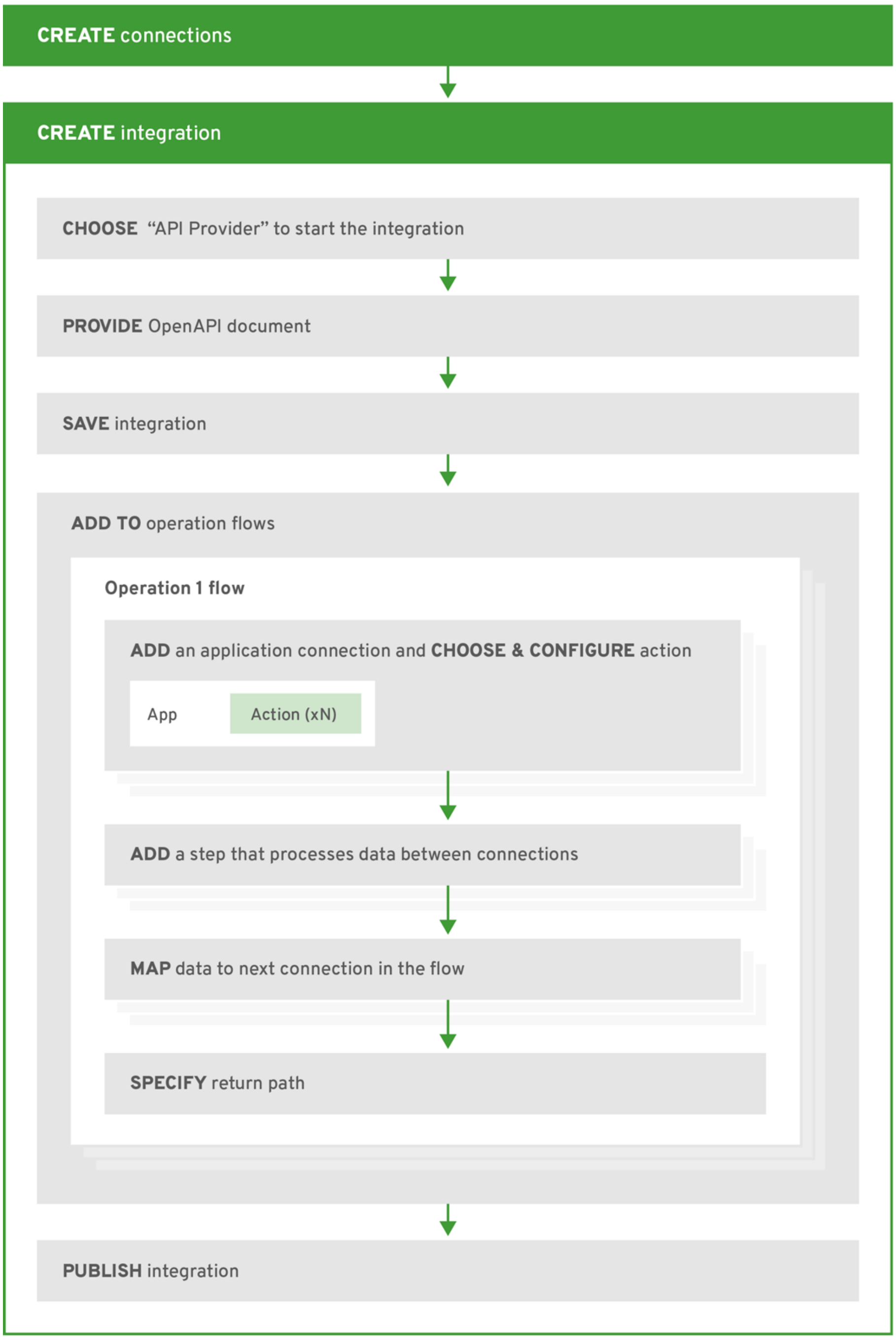
Ways to edit a flow

For each operation, you can edit its flow by:

- Adding connections to the applications that need to process the data.
- Adding steps between connections, including split, aggregate, and data mapping steps.
- Altering the return code in the HTTP response that finishes the flow. The response goes to the application that invoked the call that triggered execution of the integration.

Workflow for creating an API provider integration

The **general** workflow for creating an API provider integration is shown in the following diagram:


CREATE connections
CREATE integration
CHOOSE "API Provider" to start the integration
PROVIDE OpenAPI document
SAVE integration
ADD TO operation flows
Operation 1 flow
ADD an application connection and CHOOSE & CONFIGURE action
App
Action (xN)
ADD a step that processes data between connections
MAP data to next connection in the flow
SPECIFY return path
PUBLISH integration

FUSE_14_0419

Publishing an API provider integration

After you publish an API provider integration, in the integration's summary page, Fuse Online displays the external URL for your REST API service. This external URL is the base URL that clients use to call your REST API services.

Testing an API provider integration

To test an API provider integration's flows, you can use the **curl** utility. For example, suppose that the following **curl** command triggers execution of the flow for the **Get Task by ID** operation. The HTTP **GET** command is the default request so there is no requirement to specify **GET**. The last part of the URL specifies the ID of the task to get:

```
curl -k https://i-task-api-proj319352.6a63.fuse-ignite.openshiftapps.com/api/todo/1
```

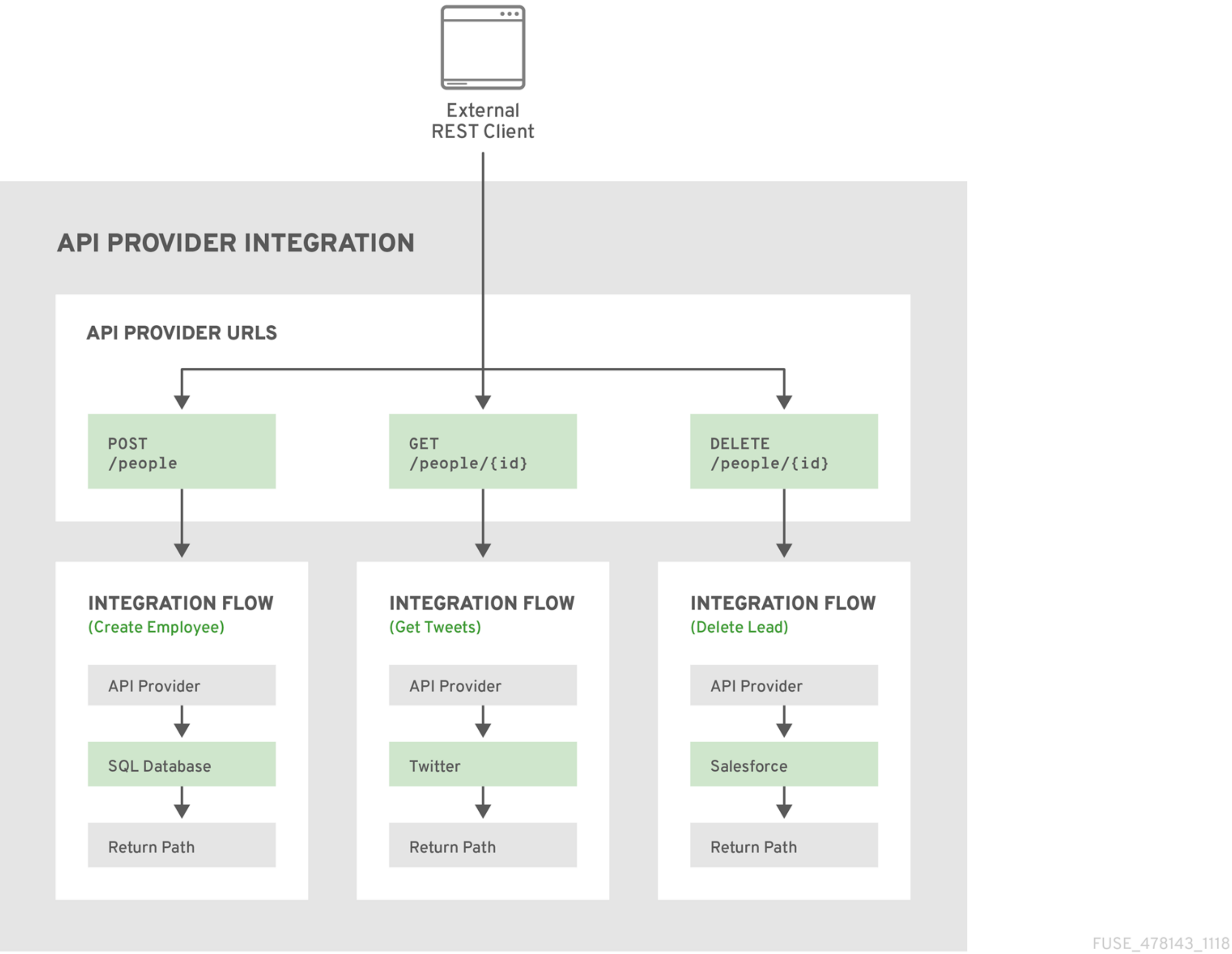
6.2. HOW OPENAPI OPERATIONS RELATE TO API PROVIDER INTEGRATION FLOWS

An API provider integration's OpenAPI document defines the operations that REST API clients can call. Each OpenAPI operation has its own API provider integration flow. Consequently, each operation can also have its own REST API service URL. Each URL is defined by the API service's base URL and optionally by a subpath. REST API calls specify an operation's URL to trigger execution of the flow for that operation.

Your OpenAPI document determines which HTTP verbs (such as **GET**, **POST**, **DELETE** and so on) you can specify in calls to your REST API service URLs. Examples of calls to API provider URLs are in the instructions for trying out the API provider quickstart example .

Illustration of an API provider integration example

The following diagram shows an API provider integration that processes data about people. An external REST API client invokes the REST API URLs that are deployed by the API provider integration. Invocation of a URL triggers execution of the flow for one REST operation. This API provider integration has 3 flows. Each flow can use any connection or step that is available in Fuse Online. The REST API along with its flows is one Fuse Online API provider integration, which is deployed in one OpenShift pod.



Editing the OpenAPI document while creating an API provider integration

After you specify an OpenAPI 2.0 document for your API provider integration, you can update the document as needed while you define the execution flows for the API operations. To do this, click **View/Edit API Definition** in the upper right of a page in which you are editing the API provider integration. This displays your OpenAPI document in the Apicurito editor. Edit and save the document to make changes that are reflected in Fuse Online.

Considerations while editing the OpenAPI document:

- **operationId properties for synchronization**
  Synchronization between the versions of the OpenAPI document in the Apicurito editor and in the Fuse Online integration editor depend on a unique **operationId** property that is assigned to each operation that is defined in the document. You can assign a specific **operationId** property value to each operation, or use the one that Fuse Online generates automatically.

- **Request and response definitions**
  In each operation's definition, you can supply a JSON schema that defines the operation's request and response. Fuse Online uses the JSON schema:

  - As the basis for the operation's input and output data shapes

  - To display operation fields in the data mapper

- **No cyclic schema references**
  A JSON schema for an API provider integration operation cannot have cyclic schema references. For example, a JSON schema that specifies a request or response body cannot reference itself as a whole nor reference any part of itself through intermediate JSON schemas.

6.3. CREATING AN API PROVIDER INTEGRATION

To create an API provider integration, provide an OpenAPI document (**.json**, **.yaml**, or **.yml** file) that defines the operations that the integration can perform. Fuse Online creates an execution flow for each operation. Edit the flow for each operation to add connections and steps that process integration data according to the requirements for that operation.

Prerequisites

- You are able to provide or define an OpenAPI document for the REST API operations that you want the integration to perform.
  To experiment, download the **task-api.json** file, which is an OpenAPI document for an API provider quickstart. You can upload this file when Fuse Online prompts you to provide an OpenAPI document.

- You have a plan for the flow for each OpenAPI operation.

- You created a connection for each application or service that you want to add to an operation's flow.

Procedure

1. In Fuse Online, in the left navigation panel, click **Integrations**.

2. In the upper right, click **Create Integration**.

3. On the **Choose a Start Connection**page, click **API Provider**.

4. On the **Start integration with an API call**page:

   - If you have an OpenAPI 2.0 document that defines the REST API operations, upload the OpenAPI document.

- If you need to define the OpenAPI 2.0 document, select **Create from scratch**.

5. Click **Next**.

   - If you uploaded a document, review or edit it:

     a. Click **Review/Edit** to open the Apicurito editor.

     b. Review and edit as needed.

     c. In the upper right, click **Save** or **Cancel** to close the editor.

     d. Click **Next** and go to step 6 in this procedure.

   - If you are creating a document, then in the Apicurito editor that Fuse Online opens:

     a. Define the OpenAPI document.

     b. In the upper right, click **Save**, which closes the editor.

     c. Click **Next**.

   For information about using the Apicurito editor, see Designing and developing an API definition with Apicurito.

6. Enter a name and description for your integration and click **Save and continue**. You are ready to define a flow for each operation .

6.4. CREATING THE OPERATION FLOWS FOR AN API PROVIDER INTEGRATION

The OpenAPI document that defines your REST API service defines the operations that the service can perform. After you create an API provider integration, you can edit the flow for each operation.

Each operation has exactly one flow. In an operation flow, you can add connections to other applications and services, as well as steps that operate on data between connections.

As you add to operation flows, you might find that you need to update the OpenAPI document that the API provider integration is based on. To do this, click **View/Edit API Definition** in the upper right of a page in which you are editing your API provider integration. This displays your document in the Apicurito editor. In your OpenAPI definition, as long as each operation has a unique **operationId** property, you can save your updates in Apicurito and Fuse Online can synchronize the API provider integration’s flow definitions to have your updates.

Prerequisites

- You created an API provider integration, gave it a name, and saved it.
- You created a connection to each application or service that you want an operation flow to connect to. For details, see the information about creating connections.
- Fuse Online is displaying the list of operations that the API defines.

Procedure

1. In the **Operations** list page, click the entry for an operation whose flow you want to edit.

2. Add all desired connections to this flow. For each connection that you want to add:

    a. In the flow visualization panel on the left, click the plus sign where you want to add a connection.

    b. On the **Choose a Step** page, click the connection that you want to add to this flow.

    c. Click the action that you want this connection to perform.

    d. Configure the action by entering data in the labeled fields.

    e. Click **Done**.

    f. In the upper right, click **Save**.

    For each connection that you want to add to this flow, repeat this subset of instructions. Add all desired connections to the flow before you continue.

3. In this operation flow, to process data between connections:

    a. In the flow visualization panel on the left, click the plus sign where you want to add a step.

    b. On the **Choose a Step** page, click the step that you want to add.

    c. Configure the step by entering data in the labeled fields.

    d. Click **Done**.

    e. In the upper right, click **Save**.
    For help, see Adding steps between connections .

    Repeat this subset of instructions to add another step that processes data between connections.

4. In the flow visualization panel on the left, check for data type mismatch ⚠ icons, which indicate that the connection cannot process the incoming data. You need to add a data mapper step here. Go back to the previous subset of instructions. On the **Choose a Step** page, click **Data Mapper**, and define the needed mappings. For help, see Mapping integration data to fields in the next connection.

5. In the flow visualization panel on the left, at the bottom, hover over the ↩ icon to see that it is the **Provided API Return Path**
    Every API provider integration finishes each operation flow by sending a response to the REST API caller that triggered execution of the operation flow. The response contains the return code that is specified here.

    In this release, whenever an API call triggers execution of this flow, the return code is the code that is specified in this step. Error handling is expected to be supported in a future release.

    Accept the flow's default return code, **200 OK**, or specify another return code as follows:

    a. Click the ↩ icon.

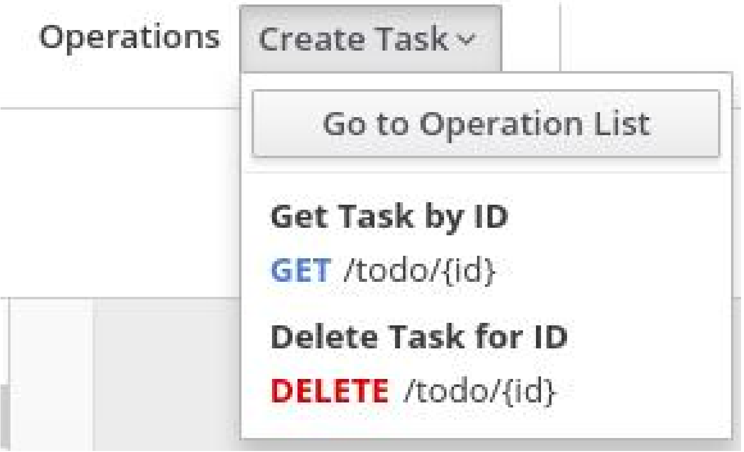
    b. Click in the **Return Code** input field, which displays a list of possible return codes.

    c. Scroll to the return code that you want and click it.

d. Click **Next**.

6. When this flow has all needed connections and steps, do one of the following:

- **Publish** - To start running the integration, in the upper right, click **Publish**. This builds the integration, deploys the REST API service to OpenShift, and makes the integration available to be executed. You can publish the integration each time that you complete the creation of an operation's flow or each time that you edit an operation's flow.
- **Edit another flow** - To edit the flow for another operation, select it from the **Operations** drop-down list, which is at the top of the page, and looks something like this:



Repeat this procedure to edit another operation's flow.

Next steps

When an API provider integration is running in Fuse Online on OpenShift Online or on OpenShift Dedicated, you can use the **curl** utility to confirm that it is working as expected. For examples of doing this, see the description of the API provider quickstart.

When an API provider integration is running in Fuse Online on OpenShift Container Platform, an administrator might have set the **CONTROLLERS_EXPOSE_VIA3SCALE** environment variable to true, which makes the integration's API discoverable in Red Hat 3scale. When this environment variable is set to true, Fuse Online does not provide an external URL for an API provider integration. To test the integration, ask a 3scale administrator for guidance.

6.5. CONFIGURING AND PUBLISHING AN EXAMPLE API PROVIDER QUICKSTART INTEGRATION

Fuse Online provides an API provider quickstart integration that you can import into your Fuse Online environment. This quickstart includes an OpenAPI document for a task management API. After importing the quickstart integration, you configure return codes, and then publish the integration. After you complete the procedure described below, the Task API integration is running and ready to be executed.

The API provider quickstart helps you quickly learn how to configure, publish, and test an API provider integration. But it is not a real-world example of how useful an API provider integration can be. For a real-world example, suppose that you already used Fuse Online to publish several simple integrations. You could define an OpenAPI document for triggering execution of those integrations. To do this, you would edit the flow for each OpenAPI operation to be almost the same as the simple integrations that you already published.

Prerequisites

- Fuse Online is open in a browser.

Procedure

1. To import the Task API quickstart integration:
    a. Go to https://github.com/syndesisio/syndesis-quickstarts/tree/1.6/api-provider and download **TaskAPI-export.zip**.

    b. In Fuse Online, in the left navigation panel, click **Integrations**.

    c. In the upper right, click **Import**.

    d. Drag and drop the **TaskAPI-export.zip** file that you downloaded. Fuse Online indicates that it has successfully imported the file.

    e. In the left navigation panel, click **Integrations** to see an entry for the **Task API** integration that you just imported. The entry indicates that configuration is required.

2. To configure the Task API integration:
    a. Click the **Task API** entry to display the integration summary.

    b. In the upper right, click **Edit Integration** to display a list of the operations that this API provides.

    c. To configure the flow for the **Create Task** operation:

        i. Click the **POST** entry for the **Create Task** operation to display a page that prompts you to add to the integration.
        Connections and steps have already been added to this flow. In the operation flow visualization panel on the left, move your cursor over the icons to see what they represent. Click a step to view its configuration. When you are done examining one, click another one to examine it. When you click a database connection, you can see the SQL statement that it executes.

        ii. In the operation flow visualization panel on the left, click the ↩ icon at the bottom. You might have to scroll down to see it. When you hover over it, Fuse Online displays **Provided API Return Path**

        iii. If you need to, scroll up on the right, click in the **Return Code** input field and scroll to select **201 Created**.

        iv. Click **Next**.

        v. In the upper right, click **Save**.

        vi. At the top, to the right of **Operations**, click the down arrow and then click **Go to Operations List**.

    d. To configure the flow for the **Delete Task for ID**operation:

        i. Click the **DELETE** entry for the **Delete Task for ID**operation to display a page that prompts you to add to the integration.
        This operation flow has a data mapper step and a connection to the sample database that is provided with Fuse Online. In the operation flow visualization panel, click the database connection to view the SQL statement that it executes.

   ii. In the operation flow visualization panel, click the ↩ icon at the bottom.

   iii. Click in the **Return Code** input field and scroll to select **200 OK**.

   iv. Click **Next**.

   v. In the upper right, click **Save**.

e. To configure the flow for the **Get Task by ID** operation:

   i. At the top of the page, to the right of **Operations**, click the down arrow and then click **Get Task by ID**.
   This operation flow has two data mapper steps, a connection to the sample database that is provided with Fuse Online, and a log step.

   ii. In the operation flow visualization panel, click an icon for a step or connection to view its configuration. When you are done examining one, click another one.

   iii. In the operation flow visualization panel, click the ↩ icon at the bottom.

   iv. Click in the **Return Code** input field and scroll to select **200 OK**.

   v. Click **Next**.

3. In the upper right, click **Publish**.
   Fuse Online displays the integrations list page and shows publication progress as it assemblies, builds, deploys, and starts the integration.

4. When the entry for the **Task API** integration displays **Running**, click the entry to display the summary page for the integration, which provides the external URL for the Task API service. It looks something like this:
   https://i-task-api-proj319352.6a63.fuse-ignite.openshiftapps.com/api/

   This is where Fuse Online makes the Task API service available. REST API calls specify URLs that start with this base URL.

   If you are using Fuse Online on OpenShift Container Platform, if the external URL is not on the integration's summary page, then an administrator has set the **CONTROLLERS_EXPOSE_VIA3SCALE** environment variable to true. To test this API provider integration, ask a 3scale administrator for guidance.

6.6. TESTING THE EXAMPLE API PROVIDER QUICKSTART INTEGRATION

When the Fuse Online Task API quickstart integration is running, you can invoke **curl** utility commands that send HTTP requests to the Task API service. How you specify the HTTP request determines the flow that the call triggers.

Prerequisites

- Fuse Online indicates that the Task API integration is **Running**.

- If your Fuse Online environment is running on OCP, Fuse Online is not configured to expose APIs to 3scale.

Procedure

1. In Fuse Online, in the left navigation panel, click **Integrations**.

2. Click the **Task API** integration to display its summary.

3. Copy the integration's external URL.

4. In a terminal, invoke a command such as the following to assign the integration's external URL to the **externalURL** environment variable. Be sure to replace the URL in this sample command with the URL that you copied and add /**api** to the end.

```
export externalURL="https://i-task-api-proj319352.6a63.fuse-ignite.openshiftapps.com/api"
```

5. Invoke a **curl** command that triggers execution of the flow for the **Create Task** operation:

```
curl -k --header "Content-Type: application/json" --request POST --data '{ "task":"my new task!"}' $externalURL/todo
```

- **-k** allows **curl** to proceed and operate even for server connections that are otherwise considered insecure.

- **--header** indicates that the command is sending JSON format data.

- **--request** specifies the HTTP **POST** command, which stores data.

- **--data** specifies the JSON format content to be stored. In this example the content is **{ "task":"my new task!"}**.

- **$externalURL/todo** is the URL to invoke.
  This command sends an HTTP **POST** request to the Task API service, which triggers execution of the **Create Task** operation's flow. Flow execution adds a new task to the sample database and returns a message such as the following to indicate what it did:

```
{"completed":false,"id":1,"task":"my new task!"}
```

6. Invoke a **curl** command that triggers execution of the flow for the **Get Task by ID** operation:

```
curl -k $externalURL/todo/1
```

To obtain a task, the **curl** command needs to specify only the URL. The HTTP **GET** command is the default request. The last part of the URL specifies the ID of the task to get.

7. Invoke a **curl** command that triggers execution of the flow for the **Delete Task for ID**operation:

```
curl -k -X DELETE $externalURL/todo/1
```

This command invokes the HTTP **DELETE** command with the same URL as the command that obtained a task by its ID.

CHAPTER 7. MAPPING INTEGRATION DATA TO FIELDS FOR THE NEXT CONNECTION

In most flows, you need to map data fields that have already been obtained or processed to data fields that the next connection in the flow can process. Fuse Online provides a data mapper to help you do this. In a flow, at each point where you need to map data fields, add a data mapper step. Details for mapping data fields are in the following topics:

7.1. IDENTIFYING WHERE DATA MAPPING IS NEEDED

Fuse Online displays warning icons to indicate where a flow requires data mapping.

Prerequisites

- You are creating or editing a flow.
- The flow contains all connections that it requires.

Procedure

1. In the flow visualization panel on the left, look for any ⚠ icons.
2. Click the icon to see the message. A **Data Type Mismatch** notification indicates that you need to add a data mapper step before that connection.
3. In the flow visualization panel, click the plus sign that appears before the connection that has the data type mismatch notification.
4. In the page that prompts you to choose a step, click **Data Mapper**.

7.2. FINDING THE DATA FIELD THAT YOU WANT TO MAP

In a flow with relatively few steps, mapping data fields is easy and intuitive. In more complex flows or in flows that handle large sets of data fields, mapping from source to target is easier when you have some background about how to use the data mapper.

The data mapper displays two columns of data fields:

- **Sources** is a list of the data fields that are obtained or processed in all previous steps in the flow.
- **Target** is a list of the data fields that the next connection in the flow expects and can process.

To quickly find the data field that you want to map, you can do any of the following:

- Search for it.
  The **Sources** panel and the **Target** panel each have a search field at the top. If the search field is not visible, click 🔍 at the top right of the **Sources** or **Target** panel.
- Enter the names of the fields that you want to map.
  To do this, in the upper right of the **Configure Mapper** page, click the plus sign to display the **Mapping Details** panel. In the **Sources** section, enter the name of the source field. In the **Action** section, accept the default **Map**, which maps one field to one other field. Or, select **Combine** or **Separate**. In the **Target** section, enter the name of the field that you want to map to.
- Expand and collapse folders to limit the visible fields.
  To view the data fields available in a particular step, expand the folder for that step.

  As you add steps to a flow, Fuse Online numbers and renumbers them to indicate the order in which Fuse Online processes the steps. When you are creating or editing a flow, these numbers are visible in the visualization panel on the left. When you are adding a data mapping step, the step numbers appear in the folder labels in the **Sources** panel and in the **Target** panel.

  The folder label also displays the name of the data type that is output by that step. Connections to applications such as Twitter, Salesforce, and SQL define their own data types. For connecting to applications such as Amazon S3, AMQ, AMQP, Dropbox, and FTP/SFTP, you define the connection’s input and/or output type when you add the connection to a flow and select the action that the connection performs. When you specify the data type, you also give the type a name. The type name you specify appears as the name of a folder in the data mapper. If you specified a description when you declared the data type, then the type description appears when you hover over the step folder in the mapper.

7.3. MAPPING ONE SOURCE FIELD TO ONE TARGET FIELD

The default mapping behavior maps one source field to one target field. For example, map the **Name** field to the **CustomerName** field.

Procedure

1. In the **Sources** panel, click the data field that you want to map from.
   You might need to expand a step to see the data fields that it provides.

When there are many source fields, you can search for the field of interest by clicking the 🔍 and entering the name of the data field in the search field.

2. In the **Target** panel, click the data field that you want to map to.
   The data mapper displays a line that connects the two fields that you just selected.

3. Optionally, preview the data mapping result. This is useful when you add a transformation to the mapping or when the mapping requires a type conversion.

   a. In the upper right of the data mapper, click ⚙ and select **Show Mapping Preview** to display a text input field on the source field and a read-only result field on the target field.

   b. In the source field's data input field, enter text.

   c. Click somewhere outside this text box to display the mapping result in the read-only field on the target field.

   d. Optionally, to see the result of a transformation, add a transformation in the **Mapping Details** panel.

   e. Hide the preview fields by clicking ⚙ again and selecting **Show Mapping Preview**.

4. Optionally, to confirm that the mapping is defined, in the upper right, click ▦ to display the defined mappings.
   You can also preview data mapping results in this view. If preview fields are not visible, click ⚙ and select **Show Mapping Preview**. Enter data as described in the previous step. In the table of defined mappings, preview fields appear for only the selected mapping. To see preview fields for another mapping, select it.

   Click ▦ again to display the data field panels.

5. In the upper right, click **Done** to add the data mapper step to the integration.

Alternative procedure

Here is another way to map a single source field to a single target field:

1. In the **Configure Mapper** page, in the upper right, click the plus sign to display the **Mapping Details** panel.

2. In the **Sources** section, enter the name of the source field.

3. In the **Action** section, accept the default **Map** action.

4. In the **Target** section, enter the name of the field that you want to map to and click **Enter**.

7.4. EXAMPLE OF MISSING OR UNWANTED DATA WHEN COMBINING OR SEPARATING FIELDS

In a data mapping, you might need to identify missing or unwanted data when a source or target field contains compound data. For example, consider a **long_address** field that has this format:

number street apartment city state zip zip+4 country

Suppose that you want to separate the **long_address** field into discrete fields for **number**, **street**, **city**, **state**, and **zip**. To do this, you select **long_address** as the source field and then select the target fields. You then add padding fields at the locations for the parts of the source field that you do not want. In this example, the unwanted parts are *apartment*, *zip+4*, and *country*.

To identify the unwanted parts, you need to know the order of the parts. The order indicates an index for each part of the content in the compound field. For example, the **long_address** field has 8 ordered parts. Starting at 1, the index of each part is:

1	*number*
2	*street*
3	*apartment*
4	*city*
5	*state*
6	*zip*
7	*zip+4*
8	*country*

In the data mapper, to identify *apartment*, *zip+4*, and *country* as missing, you add padding fields at indexes 3, 7, and 8. See Combining multiple source fields into one target field.

Now suppose that you want to combine source fields for **number**, **street**, **city**, **state**, and **zip** into a **long_address** target field. Further suppose that there are no source fields to provide content for *apartment*, *zip+4*, and *country*. In the data mapper, you need to identify these fields as missing. Again, you add padding fields at indexes 3, 7, and 8. See Separating one source field into multiple target fields.

7.5. COMBINING MULTIPLE SOURCE FIELDS INTO ONE TARGET FIELD

In a data mapper step, you can combine multiple source fields into one compound target field. For example, you can map the **FirstName** and **LastName** fields to the **CustomerName** field.

Prerequisite

For the target field, you must know what type of content is in each part of this compound field, the order and index of each part of the content, and the separator between parts, such as a space or comma. See Example of missing or unwanted data.

Procedure

1. In the **Target** panel, click the field into which you want to map more than one source field.

2. In the **Sources** panel, if there is a field that contains the fields that you want to map to the target field, then click that container field to map all contained fields to the target field.
   To individually select each source field, click the first field that you want to combine into the target field. For each of the other fields that you want to combine into the target field, hover over that field, and press **CTRL-Mouse1** (**CMD-Mouse1** on MacOS).

   The data mapper automatically changes the field action from **Map** to **Combine**.

   When you are done you should see a line from each of the source fields to the target field.

3. In the **Mapping Details** panel, in the **Separator** field, accept or select the character that the data mapper inserts in the target field between the content from different source fields. The default is a space.

4. In the **Mapping Details** panel, under **Sources**, ensure that the source fields are in the same order as the corresponding content in the compound target field.
   If necessary, drag and drop source fields to achieve the same order. The data mapper automatically updates the index numbers to reflect the new order.

5. If you mapped a source field to each part of the compound target field, then skip to the next step.
   If the target field expects data that is not available to be mapped, then in the **Mapping Details** panel, edit the index of each source field so that it is the same as the index of the corresponding data in the compound target field. The data mapper automatically adds padding fields as needed to indicate missing data.

   If you accidentally create too many padding fields, click the trash-can icon on each extra padding field to delete it.

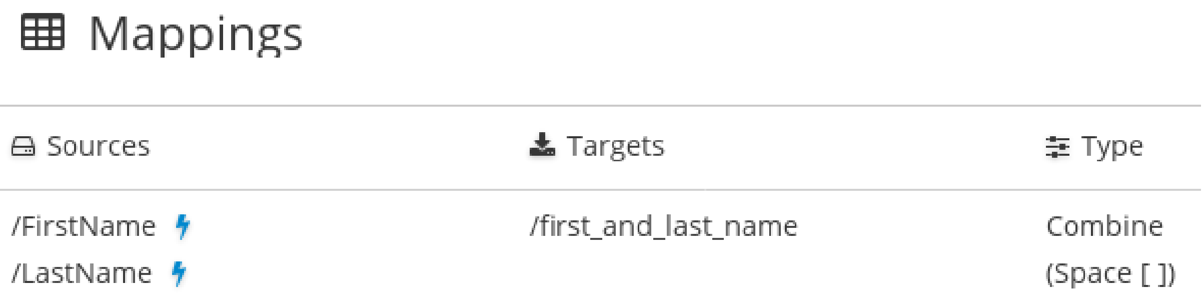
6. Optionally, preview the data mapping result:

   a. In the upper right of the data mapper, click ⚙ and select **Show Mapping Preview** to display a text input field on each source field for the currently selected mapping and a read-only result field on the target field of the currently selected mapping.

   b. In the source data input fields, enter text. Click outside the text box to display the mapping result in the read-only field on the target field.
      If you reorder the source fields or add a transformation to the mapping then the result field on the target field reflects this. If the data mapper detects any errors, it displays informative messages at the top of the **Mapping Details** panel.

   c. Hide the preview fields by clicking ⚙ again and selecting **Show Mapping Preview**.
      If you redisplay the preview fields, any data that you entered in them is still there and it remains there until you exit the data mapper.

7. To confirm that the mapping is correctly defined, in the upper right, click ▦ to display the mappings defined in this step. A mapping that combines the values of more than one source field into one target field looks like this:



Mappings

Sources	Targets	Type
/FirstName ϟ /LastName ϟ	/first_and_last_name	Combine (Space [ ])

.

You can also preview mapping results in this view. Click ⚙ , select **Show Mapping Preview**, and enter text as described in the previous step. Preview fields appear for only the selected mapping. Click another mapping in the table to view preview fields for it.

7.6. SEPARATING ONE SOURCE FIELD INTO MULTIPLE TARGET FIELDS

In a data mapper step, you can separate a compound source field into multiple target fields. For example, map the **Name** field to the **FirstName** and **LastName** fields.

Prerequisite

For the source field, you must know what type of content is in each part of this compound field, the order and index of each part of the content, and the separator between parts, such as a space or comma. See Example of missing or unwanted data.

Procedure

1. In the **Sources** panel, click the field whose content you want to separate.
2. In the **Target** panel, click the first field that you want to separate the source field data into.
3. In the **Target** panel, for each additional target field that you want to contain some of the data from the source field, hover over the field and press **CTRL-Mouse1** (**CMD-Mouse1** on MacOS) to select it.
   The data mapper automatically changes the field action to **Separate**.

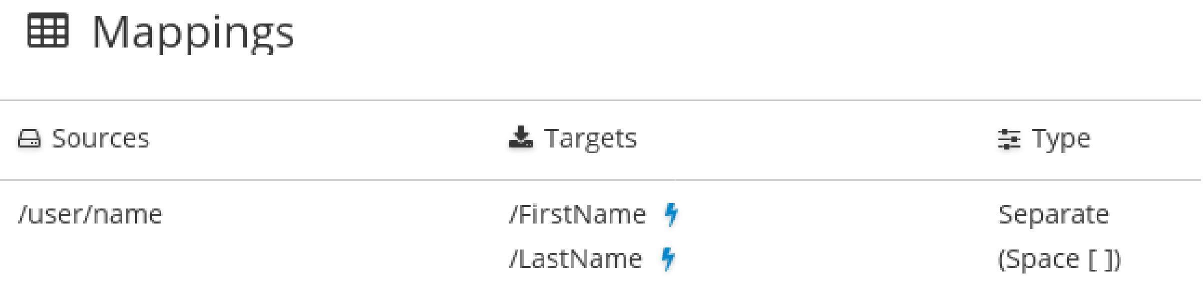
   When you are done selecting target fields, you should see lines from the source field to each of the target fields.
4. In the **Mapping Details** panel, in the **Separator** field, accept or select the character in the source field that indicates where to separate the source field values. The default is a space.
5. In the **Mapping Details** panel, under **Targets**, ensure that the target fields are in the same order as the corresponding content in the compound source field.
   If necessary, drag and drop target fields to achieve the same order. The data mapper automatically updates the index numbers to reflect the new order.
6. If you mapped each part of the compound source field to a target field, then skip to the next step.
   If the source field contains data that you do not need, then in the **Mapping Details** panel, edit the index of each target field so that it is the same as the index of the corresponding data in the compound source field. The data mapper automatically adds padding fields as needed to indicate unwanted data.
7. Optionally, preview the data mapping result:

a. In the upper right of the data mapper, click ⚙ and select **Show Mapping Preview** to display a text input field on the source field and read-only result fields on each target field.

b. In the source field's data input field, enter text. Be sure to enter the separator character between the parts of the field. Click outside the text box to display the mapping result in the read-only fields on the target fields.
If you reorder the target fields or add a transformation to a target field then the result fields on the target fields reflect this. If the data mapper detects any errors, it displays informative messages at the top of the **Mapping Details** panel.

c. Hide the preview fields by clicking ⚙ again and selecting **Show Mapping Preview**.
If you redisplay the preview fields, any data that you entered in them is still there and it remains there until you exit the data mapper.

8. To confirm that the mapping is correctly defined, click ▦ to display the mappings defined in this step. A mapping that separates the value of a source field into multiple target fields looks like this:



▦ Mappings

Sources	Targets	Type
/user/name	/FirstName /LastName	Separate (Space [ ])

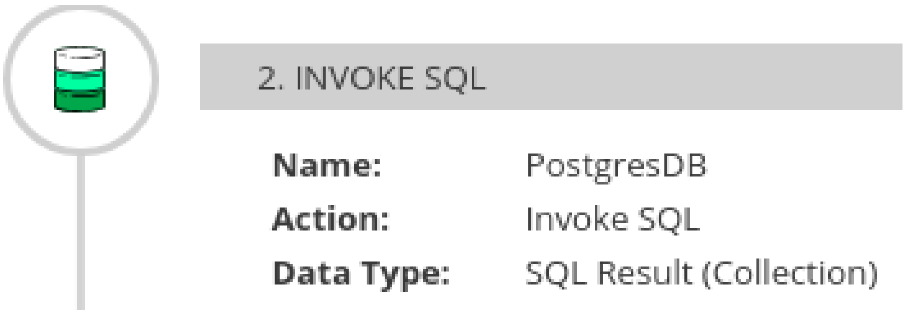
.

You can also preview mapping results in this view. Click ⚙, select **Show Mapping Preview**, and enter text as described in the previous step. Preview fields appear for only the selected mapping. Click another mapping in the table to view preview fields for it.

7.7. USING THE DATA MAPPER TO PROCESS COLLECTIONS

In a flow, when a step outputs a collection and when a subsequent connection that is in the flow expects a collection as the input, you can use the data mapper to specify how you want the flow to process the collection.

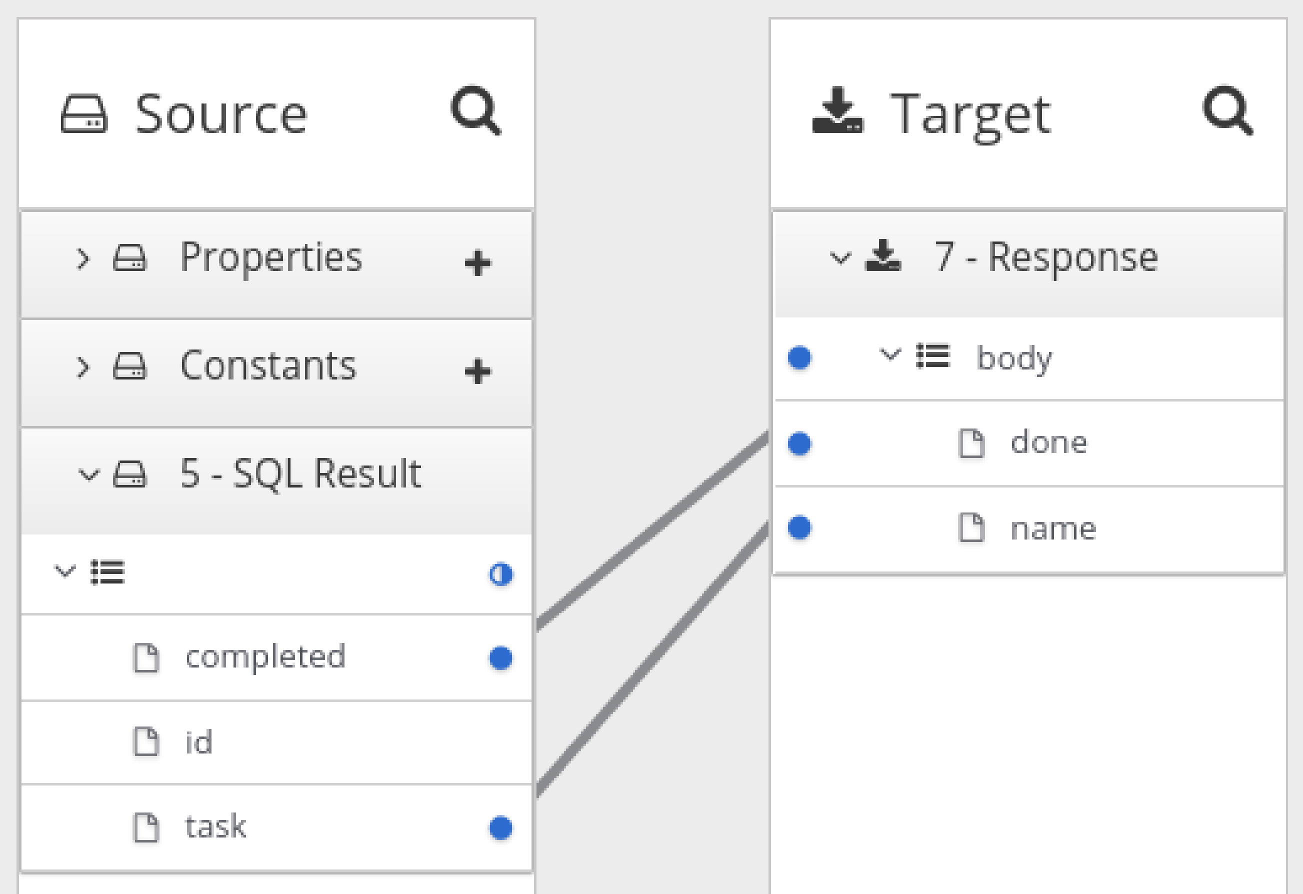
When a step outputs a collection, the flow visualization panel displays **Collection** in the details about the step. For example:



2. INVOKE SQL

Name: PostgresDB
Action: Invoke SQL
Data Type: SQL Result (Collection)

Add a data mapper step after the step that provides the collection and before the step that needs the mappings. Exactly where in the flow this data mapper step needs to be depends on the other steps in the flow. The following image shows mappings from source collection fields to target collection fields:



In the source and target panels, the data mapper displays ☰ to indicate a collection. When a source collection or a target collection contain only primitive types, the data mapper does not display collection fields because there is no need to. You can map from/to the collection itself.

When a collection contains more than one kind of primitive type or when it contains at least one complex type then the data mapper displays the collection's child fields. You can map from/to each field.

When Fuse Online executes the flow, it iterates over the source collection elements to populate the target collection elements. If you map one or more source collection fields to a target collection or to target collection fields, the target collection elements contain values for only the mapped fields.

If you map a source collection or a field in a source collection to a target field that is not in a collection, then when Fuse Online executes the flow, it assigns the value from only the last element in the source collection. Any other elements in the collection are ignored in that mapping step. However, any subsequent mapping steps can access all elements in the source collection.

When a connection returns a collection that is defined in a JSON or Java document, the data mapper can usually process the source document as a collection.

7.8. MAPPING BETWEEN COLLECTIONS AND NON-COLLECTIONS

In the data mapper **Source** and **Target** panels:

- ☰ indicates a collection. If the collection contains one primitive type, you can map directly from or to that collection. If the collection contains two or more different types, the data mapper displays the collection's child fields and you can map to or from the collection's fields.

- ▬ indicates an expandable container that is a complex type. A complex type contains multiple fields of different types. A field in a complex type can be a type that is a collection, such as an array. You cannot map a complex type container itself. You can map only the fields that are in the complex type.

To toggle the display of data types, such as **(COMPLEX)**, **STRING**, **INTEGER**, in the upper right of the data mapper, click ⚙ and click **Show Types**.

The following table shows the default behavior when mapping between collection fields and non-collection fields.

When you map from this source	To this target	During execution
A collection. (No child fields appear in the data mapper.)	A field that is not in a collection.	The data mapper maps the value that is in the last element in the source collection to the target field.
A field that is in a collection.	A field that is not in a collection.	The data mapper maps the mapped field's value that is in the last element in the source collection to the target field.
A field that is not in a collection.	A collection. (No child fields appear in the data mapper.)	The data mapper maps the value that is in the mapped source field to the first (and only) element in the collection.
A field that is not in a collection.	A field that is in a collection.	The data mapper maps the value that is in the mapped source field to the first (and only) element in the collection.

If a source collection contains fields that you do not map, those fields are still available to subsequent steps that are in the flow.

Changing default behavior when mapping from a collection field

When you map from a collection field to a non-collection field, the default behavior is that the target field gets its value from the last element in the source collection. You can change this default behavior in the following ways:

- To map from the element that you choose, apply the **Item At** transformation to the source and specify an index. For example, to map the value that is in the first element that is in the collection, specify **0** for the index.
- To map all values that are in all elements that are in a source collection, apply the **Concatenate** transformation to the source collection or source collection field and optionally specify a delimiter. The default delimiter is a space. For example, consider this source collection:
  - In the first element, the value in the **city** field is **Boston**.
  - In the second element, the value in the **city** field is **Paris**.
  - In the third element, the value in the **city** field is **Tokyo**.

During execution, the data mapper populates the target field with **Boston Paris Tokyo**.

Changing default behavior when mapping from a non-collection field

When you map from a non-collection field to a collection field, the default behavior is that the target collection contains one element, which contains the non-collection, source field value. You can change the default behavior when the source field contains a series of values that are separated by the same delimiter. For example, consider a non-collection, source **cities** field that contains:

Boston Paris Tokyo

You would map this to a target collection or to a target field that is in a collection. On the source **cities** field, add the **Split** transformation. During execution, the data mapper splits the value of the **cities** field at the space delimiter. The result is a collection that contains three elements. In the first element, the value of the **city** field is **Boston**. In the second element, the value of the **city** field is **Paris**. In the third element, the value of the **city** field is **Tokyo**.

7.9. TRANSFORMING SOURCE OR TARGET DATA

In the data mapper, after you define a mapping, you can transform any field in the mapping. Transforming a data field defines how you want to store the data. For example, you could specify the **Capitalize** transformation to ensure that the first letter of a data value is uppercase.

Procedure

1. Map the fields. This can be a one-to-one mapping, a combination mapping, or a separation mapping.
2. In the **Mapping Details** panel, under **Sources** or under **Targets**, in the box for the field that you want to transform, click the arrow that points to the trash can. This displays a field where you can select the transformation that you want the data mapper to perform.
3. Click in this field to display the list of transformations.
4. Click the transformation that you want to perform.
5. If the transformation requires any input parameters, specify them in the appropriate input fields.
6. To add another transformation, click the arrow that points to the trash can again.

Additional resource

- Available transformations.

7.10. VIEWING THE MAPPINGS IN A STEP

While you are adding or editing a data mapper step, you can view the mappings already defined in this step. This lets you check whether the correct mappings are in place.

Prerequisites

- You are creating or editing an integration.
- You are adding a data mapper step. That is, the data mapper is visible.

Procedure

Procedure

1. In the upper right, click ▦ .
2. To dismiss the list of mappings and redisplay the source and target fields, click ▦ again.

7.11. DESCRIPTIONS OF AVAILABLE TRANSFORMATIONS

The following table describes the available transformations. The date and number types refer generically to any of the various forms of these concepts. That is, number includes, for example, **integer**, **long**, **double**. Date includes, for example, **date**, **Time**, **ZonedDateTime**.

Transformation	Input Type	Output Type	Parameter (* = required)	Description
AbsoluteValue	number	number	None	Return the absolute value of a number.
AddDays	date	date	**days**	Add days to a date. The default is 0 days.
AddSeconds	date	date	**seconds**	Add seconds to a date. The default is 0 seconds.
Append	string	string	string	Append a string to the end of a string. The default is to append nothing.
Camelize	string	string	None	Convert a phrase to a camelized string by removing whitespace, making the first word lowercase, and capitalizing the first letter of each subsequent word.
Capitalize	string	string	None	Capitalize the first character in a string.
Ceiling	number	number	None	Return the whole number ceiling of a number.

Transformation	Input Type	Output Type	Parameter (* = required)	Description
Contains	any	Boolean	**value**	Return true if a field contains the specified value.
ConvertAreaUnit	number	number	**fromUnit*** **toUnit** *	Convert a number that represents an area to another unit. For the **fromUnit** and **toUnit** parameters, select the appropriate unit from the **From Unit** and **To Unit** menus. The choices are: **Square Foot**, **Square Meter**, or **Square Mile**.
ConvertDistanceUnit	number	number	**fromUnit** * **toUnit** *	Convert a number that represents a distance to another unit. For the **fromUnit** and **toUnit** parameters, select the appropriate unit from the **From Unit** and **To Unit** menus. The choices are: **Foot**, **Inch**, **Meter**, **Mile**, or **Yard**.
ConvertMassUnit	number	number	**fromUnit** * **toUnit** *	Convert a number that represents mass to another unit. For the **fromUnit** and **toUnit** parameters, select the appropriate unit from the **From Unit** and **To Unit** menus. The choices are: **Kilogram** or **Pound**.

Transformation	Input Type	Output Type	Parameter (* = required)	Description
ConvertVolumeUnit	number	number	**fromUnit** * **toUnit** *	Convert a number that represents volume to another unit. For the **fromUnit** and **toUnit** parameters, select the appropriate unit from the **From Unit** and **To Unit** menus. The choices are: **Cubic Foot**, **Cubic Meter**, **Gallon US Fluid**, or **Liter**.
DayOfWeek	date	number	None	Return the day of the week (1 through 7) that corresponds to the date.
DayOfYear	date	number	None	Return the day of the year (1 through 366) that corresponds to the date.
EndsWith	string	Boolean	**string**	Return true if a string ends with the specified **string**, including case.
Equals	any	Boolean	**value**	Return true if a field is equal to the specified **value**, including case.
FileExtension	string	string	None	From a string that represents a file name, return the file extension without the dot.

Transformation	Input Type	Output Type	Parameter (* = required)	Description
Floor	number	number	None	Return the whole number floor of a number.
Format	any	string	**template** *	In **template**, replace each placeholder (such as **%s**) with the value of the input field and return a string that contains the result. This is similar to mechanisms that are available in programming languages such as Java and C.
IndexOf	string	number	**string**	In a string, starting at 0, return the first index of the specified **string**. Return **-1** if it is not found.
IsNull	any	Boolean	None	Return true if a field is null.
LastIndexOf	string	number	**string**	In a string, starting at 0, return the last index of the specified **string**. Return **-1** if it is not found.
Length	any	number	None	Return the length of the field, or **-1** if the field is null.
Lowercase	string	string	None	Convert a string to lowercase.

Transformation	Input Type	Output Type	Parameter (* = required)	Description
Normalize	string	string	None	Replace consecutive whitespace characters with a single space and trim leading and trailing whitespace from a string.
PadStringLeft	string	string	**padCharacter** * **padCount** *	Insert the character supplied in **padCharacter** at the beginning of a string. Do this the number of times specified in **padCount**.
PadStringRight	string	string	**padCharacter** * **padCount** *	Insert the character supplied in **padCharacter** at the end of a string. Do this the number of times specified in **padCount**.
Prepend	string	string	**string**	Prefix **string** to the beginning of a string. the default is to prepend nothing.
ReplaceAll	string	string	**match** * **newString**	In a string, replace all occurrences of the supplied matching string with the supplied **newString**. The default **newString** is an empty string.

Transformation	Input Type	Output Type	Parameter (* = required)	Description
ReplaceFirst	string	string	**match** * **newString** *	In a string, replace the first occurrence of the specified **match** string with the specified **newString**. The default **newString** is an empty string.
Round	number	number	None	Return the rounded whole number of a number.
SeparateByDash	string	string	None	Replace each occurrence of whitespace, colon (:), underscore (_), plus (+), and equals (=) with a hyphen (-).
SeparateByUnderscore	string	string	None	Replace each occurrence of whitespace, colon (:), hyphen (-), plus (+), and equals (=) with an underscore (_).
StartsWith	string	Boolean	**string**	Return true if a string starts with the specified string (including case).

Transformation	Input Type	Output Type	Parameter (* = required)	Description
Substring	string	string	**startIndex** * **endIndex**	Retrieve a segment of a string from the specified inclusive **startIndex** to the specified exclusive **endIndex**. Both indexes start at zero. **startIndex** is inclusive. **endIndex** is exclusive. The default value of **endIndex** is the length of the string.
SubstringAfter	string	string	**startIndex** * **endIndex** **match** *	Retrieve the segment of a string after the specified **match** string from the specified inclusive **startIndex** to the specified exclusive **endIndex**. Both indexes start at zero. The default value of **endIndex** is the length of the string after the supplied **match** string.

Transformation	Input Type	Output Type	Parameter (* = required)	Description
SubstringBefore	string	string	**startIndex** * **endIndex** **match** *	Retrieve a segment of a string before the supplied **match** string from the supplied inclusive **startIndex** to the supplied exclusive **endIndex**. Both indexes start at zero. The default value of **endIndex** is the length of the string before the supplied **match** string.
Trim	string	string	None	Trim leading and trailing whitespace from a string.
TrimLeft	string	string	None	Trim leading whitespace from a string.
TrimRight	string	string	None	Trim trailing whitespace from a string.
Uppercase	string	string	None	Convert a string to uppercase.

7.12. TROUBLESHOOTING DATA MAPPING

A data shape change that affects a field that is already mapped might prevent the data mapper from loading a document. In this situation, when you try to edit a data mapper step that maps the affected field, the data mapper cannot display the source and target panels. Instead, it displays an error that indicates that it cannot load or cannot find the document. The error message looks like one of the following messages:

- **Data Mapper UI Initialization Error: Could not load document '-La_rwMD_ggphAW6nE9o': undefined undefined**
- **Could not find document for mapped field 'last_name' at URI atlas:json:-LaX4LMC1CfVJYp3JXM6**

You must delete this data mapper step and replace it with a new data mapper step in which you map the updated fields.

While a data shape change to a mapped field always requires you to redo the mapping, you do not always need to delete and remove the data mapper step. For example, if an XML instance specifies an input data shape and you change the name of an element, the data mapper removes the mapping that was to/from the old field name. You just need to map to/from the field with the updated name.

It is possible to change the data shape for a mapped field in the following ways:

- In an API provider integration, while editing a flow, you edit the OpenAPI document that defines the operation.
  Changing the data shape of the operation response always prevents the data mapper from being able to load the document.

- In a flow, you edit the input data type and/or the output data type for one of these kinds of connections:

  - Amazon S3
  - AMQ
  - AMQP
  - Dropbox
  - FTP/SFTP
  - HTTP/HTTPS
  - Kafka
  - IRC
  - MQTT

CHAPTER 8. MANAGING INTEGRATIONS

Each Fuse Online environment is hosted on OpenShift Online or OpenShift Container Platform (OCP). A common set up is to have a Fuse Online development environment, a Fuse Online test environment, and a Fuse Online deployment environment.

To facilitate this, Fuse Online provides the ability to export an integration from one Fuse Online environment and then import that integration into another Fuse Online environment.

The information and procedures for managing integrations are the same in each kind of Fuse Online environment, unless specifically noted.

The following topics provide information to help you manage your integrations:

- Section 8.1, “About integration lifecycle handling”
- Section 8.2, “Putting integrations into and out of service”
- Section 8.3, “Logging information about integration execution”
- Section 8.4, “Monitoring integrations”
- Section 8.5, “Testing integrations”
- Section 8.6, “Tips for troubleshooting integration execution”
- Section 8.7, “Updating integrations”
- Section 8.8, “Deleting integrations”
- Section 8.9, “Copying an integration to another environment”

8.1. ABOUT INTEGRATION LIFECYCLE HANDLING

After you create and publish an integration, you might want to update what the integration does. You can edit a draft of the published integration and then replace the running version with the updated version. To facilitate this, for each integration, Fuse Online maintains multiple versions as well as each version’s state. An understanding of integration versions and states helps you to manage your integrations.

Description of integration versions

In each Fuse Online environment, each integration can have multiple versions. Support for multiple integration versions has several benefits:

- If you publish a version that does not work correctly, then you can return to running a correct version of the integration. To do that, you stop the incorrect version and start a version that runs the way you want it to.
- As requirements or tools change, you can incrementally update an integration. You do not need to create a new integration.

Fuse Online assigns a new version number each time it starts running a new version of an integration. For example, suppose you publish the Twitter to Salesforce sample integration. After it has been running, you update the integration to use a different account to connect to Twitter. You then publish the updated integration. Fuse Online stops the running version of the integration, and publishes the updated version of the integration with an incremented version number.

The initial integration that was running is version 1. The updated integration that is now running is version 2. If you edit version 2, for example to use a different account to connect to Salesforce, and you publish that version then it becomes version 3 of the integration.

There can be exactly one draft version of an integration. Fuse Online has a definition for the draft version of an integration but it has never run this version of the integration. The draft version of an integration does not have a number. When you edit an integration, you are updating the draft version of the integration.

In Fuse Online, you can see a list of the versions of an integration in the integration's summary page. To view this page, in the left navigation panel, click **Integrations** and then click the entry for the integration whose versions you want to see.

Description of integration states

In Fuse Online, in the list of integration versions, each entry indicates the state of that version, which is one of the following:

State	Description
Running	A **Running** version is executing; it is in service. Only one version of an integration can be running. That is, only one version at a time can be in the **Running** state.
Stopped	A **Stopped** version is not running. The draft version of an integration is in the **Stopped** state. Each integration that was running at one time and then stopped is in the **Stopped** state. If no version of this integration is in the **Running** state, then you can start a version that is stopped.
Pending	A **pending** version is in transition. Fuse Online is in the process of either starting this version of the integration or stopping this version of the integration, but the integration is not yet running or stopped.
Error	An integration version that is in the **Error** state encountered an OpenShift error while being started or while running. The error suspended start-up or execution. If this happens, try starting an earlier integration version that ran correctly. Alternatively, contact technical support for help. To do that, in any Fuse Online page, in the upper right, click the ? icon and select **Support**.

8.2. PUTTING INTEGRATIONS INTO AND OUT OF SERVICE

After you create an integration, you can save it as a draft or publish it to start running it. When you publish an integration, Fuse Online assembles the needed resources, builds the integration runtime, deploys the OpenShift pod that will run the integration, and then starts running the integration.

At any time, you can click a button to stop running the integration. When you want to start the integration again, Fuse Online already has what it needs so starting it takes less time than when you published it to run it for the first time.

The process of starting a version of an integration for the first time is referred to as publishing the integration. The following topics provide details:

- Section 8.2.1, “About publishing integrations”
- Section 8.2.2, “Stopping integrations”
- Section 8.2.3, “Starting integrations”
- Section 8.2.4, “Restarting older integration versions”

8.2.1. About publishing integrations

To run a version of an integration for the first time, you publish it. Publishing an integration builds and deploys the integration runtime. The integration starts running. After publishing an integration, you can stop it and restart it. Exactly one version of an integration can be running at one time.

Alternatives for publishing

To run an integration for the first time, publish it by doing one of the following:

- At the end of the procedure in which you create or edit the integration, in the upper right, click **Publish**.
- Publish the draft version of an integration:
  1. In the left Fuse Online panel, click **Integrations**.
  2. In the list of integrations, click the entry for the integration that you want to publish.
  3. On the integration’s summary page, in the **Details** tab, to the right of the draft entry, click ⋮ and select **Publish**.

About the publication progress

Fuse Online displays the progress of the publication process, which has several stages:

1. **Assembling** creates pod resources needed to build the integration.
2. **Building** gets the integration ready to deploy.
3. **Deploying** waits for deployment of the pod that will run the integration.
4. **Starting** waits for the pod to start running the integration.
5. **Deployed** indicates that the integration is running.

During start-up, you can click **View Logs** to display OpenShift logs that provide start-up information.

Integration status after publication

When publishing the integration is complete, the **Running** status appears next to the integration name. The pod is running the integration.

8.2.2. Stopping integrations

Each integration can have exactly one version that is running. A running version is in the **Running** state. At any time, you can stop running an integration.

Prerequisite

The integration that you want to stop is in the **Running** state.

Procedure

1. In the left Fuse Online panel, click **Integrations**.
2. In the list of integrations, identify the entry for the integration that you want to stop running. The entry shows that this integration is **Running**.
3. At the far right of this integration's entry, click ⋮ and select **Stop**.

Result

Fuse Online stops running the integration. **Stopping** and then **Stopped** appears in the integration's entry in the list of integrations.

8.2.3. Starting integrations

The first time that you start an integration, the process is referred to as publishing the integration because Fuse Online has to build the integration runtime before it can run the integration. At any time, you can stop running an integration and then start it again.

Prerequisite

The integration that you want to start is in the **Stopped** state.

Procedure

1. In the left navigation panel, click **Integrations**.
2. In the list of integrations, to the right of the entry for the integration that you want to start, click ⋮ .
3. Select **Start**.

Result

Fuse Online displays **Starting** as the status of that integration version, and then **Running** when the integration is running again.

8.2.4. Restarting older integration versions

You might publish an integration that does not work the way you want it to. In this situation, you can stop the incorrect version and replace it with a version that you published previously and that runs correctly.

Prerequisites

- A version of the integration is running but you want to stop it.

- You have another version of the integration and you want to run that one.

Procedure

1. In the left panel, click **Integrations** to display a list of the the integrations in this environment.
2. Click the entry for the integration for which you want to publish an older version. Fuse Online displays a list of the versions of the integration.
3. In the entry for the version that is running, at the far right, click ⋮ and select **Stop**.
4. Click **OK** to confirm that you want to stop running this version.
5. Wait for **Stopped** to appear to the right of the integration name near the top of the page.
6. To publish the older version as is, skip to the next step. Or, before you publish the older version, you can update it:
   a. In the entry for the integration version that you want to update, at the far right, click ⋮ and select **Replace Draft**.
   b. Update the integration as needed.
   c. When updates are complete, in the upper right, click **Publish**, and then click **Publish** to confirm. This takes the place of the next two steps.
7. To publish the older version as is, in the entry for the integration version that you want to start running again, at the far right, click ⋮ and select **Start**.
8. Click **Start** to confirm that you want to start this version of the integration.

Result

Fuse Online starts the integration, which takes a few minutes. When the integration is running, then **Running version** ***n*** appears to the right of the integration's name.

8.3. LOGGING INFORMATION ABOUT INTEGRATION EXECUTION

For each execution of an integration, for each step in a flow, Fuse Online provides the following activity information:

- The date and time that the step was executed
- How long it took to execute the step
- Whether execution was successful
- The error message if execution was not successful

To view this information in Fuse Online, display the integration's summary and then click the **Activity** tab.

To obtain further details about integration execution, you can log information about the messages that an integration processes by adding a log step to an integration flow. For each message that an integration receives, a log step can provide one or more of the following:

- The message’s context, which provides metadata about the message, including the message’s header.
- The message’s body, which provides the content of the message.
- Text that you specify either explicitly or through evaluation of an Apache Camel Simple language expression.

NOTE

The Log connection that was available in previous releases is no longer available to be added to integrations. Add a log step instead of a log connection.

Prerequisites

You are creating or editing a flow and Fuse Online is prompting you to choose a step. Or, Fuse Online is prompting you to choose a finish connection.

Procedure

1. Click **Log**.
2. On the **Configure Log Step** page, select the content that you want to log. If you select **Custom Text**, then in the text input field, enter one of the following:
   - The text that you want to log
   - A Camel Simple language expression

   If you enter an expression, Fuse Online resolves the expression and logs the resulting text.
3. When log step configuration is complete, click **Done** to add the step to the flow.
4. When the flow is complete, publish the integration to start seeing output from the new log step.

Additional resources

After a flow that has a log step has been executed, output from the log step appears in the integration’s **Activity** tab. See Viewing integration activity information .

8.4. MONITORING INTEGRATIONS

Fuse Online provides various ways for you to monitor the execution of your integrations. See:

- Section 8.4.1, “Viewing integration history”
- Section 8.4.2, “Viewing information about an integration’s activity”
- Section 8.4.3, “Viewing metrics for a particular integration”
- Section 8.4.4, “Viewing metrics for a Fuse Online environment”

8.4.1. Viewing integration history

Fuse Online maintains each version of an integration. You can always view a list of the versions of each integration.

Procedure

1. In the left panel, click **Integrations** to display a list of the integrations in your environment.
2. Click the entry for the integration whose versions you want to see.

Result

In the page that appears, the **History** section lists the versions of the integration. The ✓ icon identifies the current version, which is the most recently, successfully running version. For each version, you can also see the date on which it was last started.

To edit, start, or stop a particular version, click the ⋮ to the right of the version's entry. Select the operation that you want to perform.

8.4.2. Viewing information about an integration's activity

Fuse Online provides activity information for each execution of an integration. This information is part of the integration's log. For each step in a flow, Fuse Online provides:

- The date and time that the step was executed
- How long it took to execute the step
- Whether execution was successful
- The error message if execution was not successful

At any time, you can view this information.

Prerequisites

- There is or was a running integration for which you want to view activity information.
- This integration has been executed at least once.

Procedure

1. In the left panel, click **Integrations**.
2. Click the entry for the integration for which you want to view activity information.
3. In the integration's summary page, click the **Activity** tab.
4. Optionally, enter date and/or keyword filters to limit the executions listed.
5. Click the integration execution for which you want to view activity information.

Additional resources

- To obtain additional information between any two steps, you can add a log step to the integration. A log step provides information about each message it receives and can provide custom text that you specify. If you add a log step, then it appears as one of the integration's steps when you expand the integration execution that you want to view activity information for. You view Fuse Online information for a log step in the same way that you view Fuse Online information for any other step. See Logging information about integration execution .

8.4.3. Viewing metrics for a particular integration

Fuse Online provides the following metrics for each integration:

- **Total Errors** indicates the number of runtime errors that all executions of this integration encountered during the past 30 days.
- **Last Processed** displays the most recent date and time that this integration processed a message. The message might have been successfully processed or there might have been an error.
- **Total Messages** is the number of messages that all executions of this integration processed in the last 30 days. This includes message failures.
- **Uptime** indicates when this integration started running and how long it has been running without an error.

Prerequisite

The integration that you want to view metrics for is running or has been running.

Procedure

1. In the left panel, click **Integrations**.
2. Click the entry for the integration for which you want to view metrics.
3. In the integration's summary page, click **Metrics**.

8.4.4. Viewing metrics for a Fuse Online environment

Metrics for your Fuse Online environment appear on the Fuse Online home page.

Procedure

In the left panel, click **Home**.

Results

Fuse Online updates the following metrics every 5 seconds:

- The number of integrations that are defined in this environment as well as the number of integrations that are running, the number of integrations that are stopped, and the number of integrations that are pending. Fuse Online is either stopping or starting pending integrations. A red cross indicates any integrations that were running but that encountered an error that suspended execution.
- The number of connections that are defined in this environment.
- Total number of messages that have been processed by integrations in this environment in the last 30 days. This includes messages that were processed by integrations that might no longer be running or that might have been deleted from this environment.

- Uptime indicates how long there has been at least one integration that is running. The date and time when uptime started appears as well.

8.5. TESTING INTEGRATIONS

After you create an integration and it is running correctly in a Fuse Online development environment, you might want to run it in a different Fuse Online environment to test it.

Prerequisite

- You have a Fuse Online development environment and a Fuse Online test environment.
- You have an integration that is running correctly in your Fuse Online development environment.

Procedure

1. Learn about copying an integration to another environment .
2. Export the integration from the development environment. See Exporting an integration.
3. Import the integration into the test environment. See Importing an integration.

8.6. TIPS FOR TROUBLESHOOTING INTEGRATION EXECUTION

If an integration stops working, check its activity and history details. See Viewing integration activity information and Viewing integration history.

For a connection to an application that uses OAuth, you might see an error message that indicates that the access token for the application has expired. Sometimes, you might get a less explicit **403 - Access denied** message. The information in the message depends on the application that the integration is connecting to. In this situation, try reconnecting to the application and then republishing the integration:

1. In the left panel, click **Integrations**.
2. In the list of integrations, click the entry for the integration that stopped running.
3. In the integration’s summary page, in the flow visualization, click the icon for the application that you want to reconnect to.
   If this is an API provider integration, then in the integration’s summary page, click its flow icon to display the operations list. Then click the operation whose path contains the connection that is failing and then click the failing connection in the operation’s path visualization.
4. In the connection’s details page, click **Reconnect**.
5. Respond to that application’s OAuth workflow prompts.
   Fuse Online displays a message to indicate that its access to that application has been authorized. For some applications, this takes a few seconds but it can take longer for other applications.
6. After reconnecting to the application, start the integration.

If reconnection is not successful, try this:

1. Re-register Fuse Online as a client of the application. See General procedure for obtaining authorization.

2. Create a new connection.
3. Edit each integration that was using the old connection:
    a. Remove the old connection.
    b. Replace it with the new connection.
4. Publish each updated integration.

8.7. UPDATING INTEGRATIONS

After you create an integration, you might need to update it to add, edit or remove a step.

Prerequisite

In your Fuse Online environment, there is a version of the integration that you want to update.

Procedure

1. In the left Fuse Online panel, click **Integrations**.
2. In the list of integrations, click the entry for the integration that you want to update.
3. On the integration's summary page, in the upper right corner, click **Edit Integration**.
   If this is a simple integration, the left panel displays the integration's flow. If this is an API provider integration, Fuse Online displays the operations list. To update the flow for a particular operation, click the operation to display its flow.
4. Update the flow as needed:
    - To add a step, in the left panel, click the plus sign that is in the location where you want to add it. Then click the card that represents the step that you want to add.
    - To delete a step, in the left panel, click 🗑 to the right of the step that you want to delete. You might need to expand the width of the visualization panel to see these icons.
    - To change the configuration of a step, in the left panel, click the step that you want to update. In the configuration page, update the parameter settings as needed.

8.8. DELETING INTEGRATIONS

After you delete an integration, Fuse Online still has the history of that integration. If you import a version of the deleted integration, then Fuse Online associates the history of the deleted integration with the imported integration.

Procedure

1. In the left Fuse Online panel, click **Integrations**.
2. In the list of integrations, at the right of the entry for the integration that you want to delete, click ⋮ and select **Delete**.
3. In the popup, click **OK** to confirm that you want to delete the integration.

8.9. COPYING AN INTEGRATION TO ANOTHER ENVIRONMENT

To run integrations across development, staging and production environments, you can export and import integrations. The environments can all be on a single OpenShift cluster, or they can be spread out across multiple OpenShift clusters. See the following topics:

- Section 8.9.1, “About copying integrations”
- Section 8.9.2, “Exporting integrations”
- Section 8.9.3, “Importing integrations”

8.9.1. About copying integrations

Each Fuse Online installation is an environment from which you can export an integration. Exporting an integration downloads a zip file that contains the information needed to recreate the integration in a different Fuse Online environment.

In an environment, each integration can have only one **Draft** version.

The result of importing an integration depends on:

- Whether the integration was previously imported
- Whether a connection that the integration uses was previously imported

Fuse Online uses an internal identifier for each integration and each connection to determine whether it already exists in the environment that it is being imported into. If you change the name of an integration or connection, Fuse Online recognizes it as the same integration or connection, which just has a different name.

The following table describes the possible results of importing an integration:

In the importing environment:	What the import operation does:
The integration has not been previously imported.	Creates the integration. The integration is in the **Draft** state.
The integration has been previously imported.	Fuse Online updates the integration. The updated integration is in the **Draft** state. If there was a**Draft** version of this integration, it is lost.
The imported integration uses a connection that did not exist in the environment before the import operation.	Fuse Online creates a connection that has the same settings except for secrets. You must review each new connection. If a connection is not completely configured for its new environment then you must add the missing settings. For example, you might need to obtain secret settings by registering this Fuse Online environment as a client of the application that this connection accesses.

8.9.2. Exporting integrations

When Fuse Online exports an integration it downloads a zip file to your local **Downloads** folder. This zip file contains the information needed to recreate the integration in a different Fuse Online environment.

Exporting an integration is also a way to have a backup of the integration. However, Fuse Online maintains the versions of an integration so exporting an integration is not required for having a backup copy.

Procedure

1. In the left panel of Fuse Online, click **Integrations**.
2. In the list of integrations, identify the entry for the integration that you want to export.
3. At the right of the entry, click ⋮ and select **Export**.

Next step

To import the integration into another Fuse Online environment, open that environment and import the exported zip file.

8.9.3. Importing integrations

In a Fuse Online environment, you can import an integration that was exported from another Fuse Online environment. Exporting an integration downloads the zip file that you upload to import the integration.

Prerequisite

You have a zip file that contains an integration that was exported in another Fuse Online environment.

Procedure

1. Open the Fuse Online environment that you want to import the integration into.
2. In the left panel, click **Integrations**.
3. In the upper right, click **Import**.
4. Drag and drop one or more exported integration zip files, or navigate to a zip file that contains an exported integration and select it.
5. After Fuse Online imports the file(s), click **Done**. Fuse Online displays information about imported integrations.
6. In the left panel, click **Connections**.
   If an imported integration uses a connection that requires configuration, then there is a **Configuration Required** message at the bottom of the connection’s card.
7. For each connection that requires configuration:
   a. Click it to display its details.
   b. Enter or change connection details as needed. It is possible that every field on this page is correct and that only security configuration is required.
   c. If you updated any fields, click **Save**.

   d. In the left panel, click **Settings**.
      The **Settings** page displays entries for applications that use the OAuth protocol.

8. For each connection that requires configuration and that accesses an application that uses the OAuth protocol, register your Fuse Online environment with the application. The steps vary for each application. See the appropriate topic:

   - Registering with Dropbox
   - Registering with Google
   - Registering with a REST API
   - Registering with Salesforce
   - Connecting to SAP Concur
   - Registering with Twitter

9. In the left panel, click **Connections** and confirm that there are no longer any connections that require configuration.

10. In the left panel, click **Integrations**. In the list of integrations, imported integrations have a green triangle in the upper left corner of their entries.

11. In the list of integrations, at the right of the entry for the integration that you imported, click ⋮ and select **Edit**.

12. In the upper right, click **Save** or, if you want to start running the imported integration, click **Publish**. Regardless of whether you save the integration as a draft or you publish the integration, Fuse Online updates the integration to use the updated connections.

CHAPTER 9. CUSTOMIZING FUSE ONLINE

Fuse Online provides many connectors that you can use to connect to common applications and services. There are also a number of built-in steps for processing data in common ways. However, if Fuse Online does not provide a feature that you need, talk with a developer about your requirements. An experienced developer can help you customize integrations by providing any of the following:

- An OpenAPI document that Fuse Online can use to create a connector for a REST API client.
  You upload this schema to Fuse Online and Fuse Online creates a connector according to the schema. You then use the connector to create a connection that you can add to an integration. For example, many retail web sites provide a REST API client interface that a developer can capture in an OpenAPI document.

- An OpenAPI document that defines a REST API service.
  You upload this schema to Fuse Online. Fuse Online makes the API service available and provides the URL for API calls. This lets you trigger integration execution with an API call .

- A **JAR** file that implements a Fuse Online extension. An extension can be any one of the following:

  - One or more steps that operate on integration data between connections

  - A connector for an application or service

  - A library resource such as a JDBC driver for a proprietary SQL database
    You upload this **JAR** file to Fuse Online and Fuse Online makes the custom feature provided by the extension available.

See the following topics for details:

- Section 9.1, “Developing REST API client connectors”

- Section 9.2, “Adding and managing REST API client connectors”

- Section 9.3, “Developing Fuse Online extensions”

- Section 9.4, “Adding and managing extensions”

9.1. DEVELOPING REST API CLIENT CONNECTORS

Fuse Online can create connectors for Representational State Transfer Application Programming Interfaces (REST APIs) that support Hypertext Transfer Protocol (HTTP). To do this, Fuse Online requires a valid OpenAPI 2.0 document that describes a REST API that you want to connect to. If the API service provider does not make an OpenAPI document available then an experienced developer must create the OpenAPI document.

The following topics provide information and instructions for developing REST API connectors:

- Section 9.1.1, “Requirements for REST API client connectors”

- Section 9.1.2, “Guidelines for OpenAPI schemas for REST API client connectors”

- Section 9.1.3, “Provide client credentials in parameters”

- Section 9.1.4, “Automatically refresh access tokens”

9.1.1. Requirements for REST API client connectors

After you upload an OpenAPI schema to Fuse Online, a connector to the REST API becomes available. You can select the connector to create a REST API client connection. You can then create a new integration and add the REST API client connection, or you can edit an existing integration to add the REST API client connection.

Fuse Online connectors support OAuth 2.0 and HTTP Basic Authorization. If access to the REST API requires Transport Layer Security (TLS) then the API needs to use a valid certificate that is issued by a recognized Certificate Authority (CA).

A REST API that uses OAuth must have an authorization URL that takes a client callback URL as input. After Fuse Online creates the connector and before you use the connector to create a connection, you must visit that URL to register your Fuse Online environment as a client of the REST API. This authorizes your Fuse Online environment to access the REST API. As part of registration, you provide the Fuse Online callback URL. The details for doing this are described in Connecting Fuse Online to Applications and Services, Registering with REST APIs.

For a REST API that uses OAuth, Fuse Online supports only the **Authorization Code** grant flow for obtaining authorization. In the OpenAPI document that you upload to create the connector, in the OAuth **securityDefinitions** object, you must set the **flow** attribute to **accessCode**, for example:

```
securityDefinitions:
  OauthSecurity:
    type: oauth2
    flow: accessCode
    authorizationUrl: 'https://oauth.simple.api/authorization'
    tokenUrl: 'https://oauth.simple.api/token'
```

You must not set **flow** to **implicit**, **password**, or **application**.

The OpenAPI schema for a REST API client connector cannot have cyclic schema references. For example, a JSON schema that specifies a request or response body cannot reference itself as a whole nor reference any part of itself through any number of intermediate schemas.

Fuse Online cannot create connectors for REST APIs that support the HTTP 2.0 protocol.

9.1.2. Guidelines for OpenAPI schemas for REST API client connectors

When Fuse Online creates a REST API client connector, it maps each resource operation in the OpenAPI document to a connection action. The action name and action description come from documentation in the OpenAPI document.

The more detail that the OpenAPI document provides, the more support Fuse Online can offer when connecting to the API. For example, the API definition is not required to declare data types for requests and responses. Without type declarations, Fuse Online defines the corresponding connection action as typeless. However, in an integration, you cannot add a data mapping step immediately before or immediately after an API connection that performs a typeless action.

One remedy for this is to add more information to the OpenAPI document. Identify the OpenAPI resource operations that will map to the actions you want the API connection to perform. In the OpenAPI document, ensure that there is a JSON schema that specifies each operation’s request and response types.

After you upload the schema, Fuse Online gives you an opportunity to review and edit it in Apicurito, which is a visual editor for designing APIs based on the OpenAPI document. You can add more detail, save your updates, and Fuse Online creates an API client connector that incorporates your updates.

If the OpenAPI document for the API declares support for **application/json** content type and also **application/xml** content type then the connector uses the JSON format. If the OpenAPI document specifies **consumes** or **produces** parameters that define both **application/json** and **application/xml**, then the connector uses the JSON format.

9.1.3. Provide client credentials in parameters

When Fuse Online tries to obtain authorization to access an OAuth2 application, it uses HTTP basic authentication to provide client credentials. If you need to, you can change this default behavior so that Fuse Online passes client credentials to the provider as parameters instead of using HTTP basic authentication. This affects the use of the **tokenUrl** endpoint that is used to obtain an OAuth access token.

IMPORTANT

This is a Technology Preview feature.

To specify that Fuse Online should pass client credentials as parameters, in the **securityDefinitions** section of the OpenAPI document, add the **x-authorize-using-parameters** vendor extension with a setting of **true**. In the example below, the last line specifies **x-authorize-using-parameters**:

```
securityDefinitions:
  concur_oauth2:
    type: 'oauth2'
    flow: 'accessCode'
    authorizationUrl: 'https://example.com/oauth/authorize'
    tokenUrl: 'https://example.com/oauth/token'
    scopes:
      LIST: Access List API
    x-authorize-using-parameters: true
```

The setting of the **x-authorize-using-parameters** vendor extension is **true** or **false**:

- **true** indicates that client credentials are in parameters.
- **false**, the default, indicates that Fuse Online uses HTTP basic authentication to provide client credentials.

9.1.4. Automatically refresh access tokens

If an access token has an expiration date, then Fuse Online integrations that use that token to connect to an application stop running successfully when the token expires. To obtain a new access token, you must either reconnect to the application or re-register with the application.

If you need to, you can change this default behavior so that Fuse Online automatically requests a new access token in the following situations:

- When the expiration date has been reached.
- When HTTP response status codes that you specify are received.

IMPORTANT

This is a Technology Preview feature.

To specify that Fuse Online should automatically try to obtain a new access token in the situations described, in the **securityDefinitions** section of the OpenAPI document, add the **x-refresh-token-retry-statuses** vendor extension. The setting of this extension is a comma separated list that specifies HTTP response status codes. When an access token's expiration date is reached or when Fuse Online receives a message from an OAuth2 provider and the message has one of these response status codes, then Fuse Online automatically tries to obtain a new access token. In the example below, the last line specifies **x-refresh-token-retry-statuses**:

```
securityDefinitions:
  concur_oauth2:
    type: 'oauth2'
    flow: 'accessCode'
    authorizationUrl: 'https://example.com/oauth/authorize'
    tokenUrl: 'https://example.com/oauth/token'
    scopes:
      LIST: Access List API
    x-refresh-token-retry-statuses: 401,402,403
```

NOTE

Sometimes, an API operation fails and a side effect of that failure is that the access token is refreshed. In this situation, even when obtaining a new access token is successful, the API operation still fails. In other words, Fuse Online does not retry the failed API operation after it receives the new access token.

9.2. ADDING AND MANAGING REST API CLIENT CONNECTORS

Fuse Online can create a REST API client connector from an OpenAPI document. For information about the content of the OpenAPI document, see Developing REST API client connectors .

The following topics provide information and instructions for adding and managing REST API client connectors:

- Section 9.2.1, "Creating REST API client connectors"
- Section 9.2.2, "Updating API client connectors by creating new ones"
- Section 9.2.3, "Deleting API client connectors"

After you create a REST API client connector, for details about using that connector, see Connecting Fuse Online to Applications and Services, Connecting to REST APIs.

9.2.1. Creating REST API client connectors

Upload an OpenAPI document to enable Fuse Online to create a REST API client connector.

Prerequisite

You have an OpenAPI document for the connector that you want Fuse Online to create.

Procedure

1. In the Fuse Online navigation panel, click **Customizations** to display the **API Client Connectors** tab. Any API client connectors that are already available are listed here.

2. Click **Create API Connector**.

3. On the **Create API Client Connector** page, do one of the following:

   - Click **Browse** and select the OpenAPI file that you want to upload.

   - Select **Use a URL** and paste the URL for your OpenAPI document in the input field.

4. Click **Next**. If there is invalid or missing content, Fuse Online displays information about what needs to be corrected. Select a different OpenAPI file to upload or click **Cancel**, revise the OpenAPI file, and upload the updated file.
   If the schema is valid, Fuse Online displays a summary of the actions that the connector provides. This might include errors and warnings about the action definitions.

5. If you are satisfied with the action summary, click **Next**.
   Or, to revise the OpenAPI document, in the lower right, click **Review/Edit** to open the Apicurito editor. Update the schema as needed. In the upper right, click **Save** to incorporate your updates into the new API client connector. Then click **Next** to continue creating the API client connector.

   For details about using Apicurito, see Designing and developing an API definition with Apicurito .

   Sometimes, if you provide a URL for the OpenAPI document, Fuse Online can upload it but cannot open it for editing. Typically, this is caused by settings on the file’s host. To open the schema for editing, Fuse Online requires that the file host has:

   - An **https** URL. An **http** URL does not work.

   - Enabled CORS.

6. Indicate the API’s security requirements by selecting one of the following:

   a. **No Security**

   b. **HTTP Basic Authorization** – Enter the user name and password you want to use to access the API.

   c. **OAuth** – Fuse Online prompts you to enter:

      i. **Authorization URL** is the location for registering Fuse Online as a client of the API. Registration authorizes Fuse Online to access the API. See Connecting Fuse Online to Applications and Services, Registering Fuse Online as a REST API client. The OpenAPI document or other documentation for the API should specify this URL. If it does not then you must contact the service provider to obtain this URL.

      ii. **Access Token URL** is required for OAuth authorization. Again, the OpenAPI document or other documentation for the API should provide this URL. If it does not then you must contact the service provider.

7. Click **Next**. Fuse Online displays the connector’s name, description, host, and base URL as indicated by the OpenAPI document. For connections that you create from this connector,

- Fuse Online concatenates the host and base URL values to define the endpoint for the connection. For example, if the host is **https://example.com** and the base URL is **/api/v1** then the connection endpoint is **https://example.com/api/v1**.
- Fuse Online applies the OpenAPI document to data mapping steps. If the OpenAPI document supports more than one schema then Fuse Online uses the TLS (HTTPS) schema.

8. Review the connector details and optionally upload an icon for the connector. If you do not upload an icon, Fuse Online generates one. You can upload an icon at a later time. When Fuse Online displays the flow of an integration, it displays a connector's icon to represent connections that are created from that connector.
   To override a value obtained from the OpenAPI file, edit the field value that you want to change. After Fuse Online creates a connector, you cannot change it. To effect a change, you need to upload an updated OpenAPI document so that Fuse Online can create a new connector or you can upload the same schema and then edit it in Apicurito. You then continue the process for creating a new API client connector.

9. When you are satisfied with the connector details, click **Create API Connector**. Fuse Online displays the new connector with the other connectors.

Next step

For details about using your new API connector, see Connecting Fuse Online to Applications and Services, Connecting to REST APIs.

9.2.2. Updating API client connectors by creating new ones

When there is an update to an OpenAPI document from which you created an API client connector, and you want your API client connector to use those updates, you must create a new API client connector. You cannot directly update an API client connector. After you create the new API client connector, you use it to create a new connection and then you edit each integration that uses a connection that was created from the out-of-date connector.

Prerequisite

Be prepared to do one of the following:

- Upload the updated OpenAPI document.
- Upload the out-of-date schema again and update it in Apicurito.

Procedure

1. Create a new API client connector based on the updated OpenAPI document. To easily distinguish between the old connector and the new connector, you might want to specify a version number in the connector name or the connector description.
   See Creating REST API client connectors .

2. Create a new connection from the new connector. Again, you want to be able to easily distinguish between connections created from the old connector and connections created from the new connector. A version number in the connection name or connection description is helpful.

3. Edit each integration that uses a connection that was created from the old connector by removing the old connection and adding the new connection.

4. Publish each updated integration.

5. Recommended, but not required: delete the old connector and the old connections.

9.2.3. Deleting API client connectors

You cannot delete a connector when there is a connection that was created from that connector and this connection is being used in an integration. After you delete an API client connector, you cannot use a connection that was created from that connector.

Procedure

1. In the left panel, click **Customizations**.

2. In the **API Client Connectors** tab, to the right of the name of the connector that you want to delete, click **Delete**.

3. Read the confirmation popup to be sure that you want to click **Delete**.

9.3. DEVELOPING FUSE ONLINE EXTENSIONS

If Fuse Online does not provide a feature that is needed to create an integration, then an expert developer can code an extension that provides the needed behavior. The Syndesis extensions repository contains examples of extensions.

A business integrator shares requirements with a developer who codes the extension. The developer provides a **.jar** file that contains the extension. The business integrator uploads the **.jar** file in Fuse Online to make the custom connector, custom step(s), or library resource available for use in Fuse Online.

The Fuse Tooling plugin to Red Hat Developer Studio provides a wizard that helps you develop a step extension or a connector extension. It is a matter of personal preference whether you choose to develop a step extension or a connector extension in Developer Studio or in some other IDE. For information about using the Developer Studio plugin, see Developing extensions for Fuse Online integrations .

In this document, the following topics outline the procedure, describe the requirements, and provide additional examples for developing extensions in an IDE that you choose.

- Section 9.3.1, “General procedure for developing extensions”
- Section 9.3.2, “Description of the kinds of extensions”
- Section 9.3.3, “Overview of extension content and structure”
- Section 9.3.4, “Requirements in an extension definition JSON file”
- Section 9.3.5, “Description of Maven plugin that supports extensions”
- Section 9.3.6, “How to specify data shapes in extensions”
- Section 9.3.7, “Alternatives for developing step extensions”
- Section 9.3.8, “Example of developing a connector extension”
- Section 9.3.9, “How to develop library extensions”

- Section 9.3.10, “Creating JDBC driver library extensions”

9.3.1. General procedure for developing extensions

Before you start to develop an extension, become familiar with the tasks that you will need to accomplish.

Prerequisites

- Familiarity with Maven
- Familiarity with Camel if you are developing an extension that provides a connector or that provides an integration step that operates on data between connections
- Experience programming

CAUTION

An integration pod runs in a Java process with a flat classpath. To avoid version clashes, make sure that the dependencies that an extension uses are aligned with the imported bill of materials (BOM) from all of these sources:

- **org.springframework.boot:spring-boot-dependencies:$SPRING_BOOT_VERSION**
- **org.apache.camel:camel-spring-boot-dependencies:$CAMEL_VERSION**
- **io.syndesis.integration:integration-bom:$SYNDESIS_VERSION**

If there are additional dependencies that are not part of the imported BOMs, you must:

- Package them in the extension JAR file that is in the **lib** directory.
- Omit them from the **dependencies** property of the extension’s JSON descriptor file.

Procedure

1. Obtain an understanding of what the extended feature must do. Talk to your business colleague to understand the feature requirements.
2. Determine whether you need to develop a step extension, a connector extension, or a library extension.
3. Set up the Maven project in which to develop the extension.
4. If you are developing a step extension:
    a. Decide whether to implement it as a Camel route or implement it by using the Syndesis **Step** API. Information for the Syndesis API is at http://javadoc.io/doc/io.syndesis.extension/extension-api.
    b. If you choose to implement the extension as a Camel route, decide whether to implement XML fragments, a **RouteBuilder** class, or a bean.
    c. In your Maven project, specify the required metadata, such as the **schemaVersion**, extension **name**, **extensionId**, and so on.

5. Code the classes that implement the feature.

6. Add dependencies to the project’s **pom.xml** file.

7. For connector and library extensions, and for step extensions that you implement in XML, create the JSON file that defines the extension.
   For step extensions that you implement in Java, Maven can generate the JSON extension definition file for you when you specify corresponding data structure values in your Maven project.

8. Run Maven to build the extension and create the extension’s JAR file.

9. Test the extension by uploading the JAR file to a Fuse Online development environment.

10. Provide the JAR file that packages the extension to your business colleague, who uploads it to a Fuse Online production environment. When you provide the JAR file, let your business colleague know about any configuration settings that require information beyond what appears in the Fuse Online web interface.

9.3.2. Description of the kinds of extensions

An extension defines one of the following:

- One or more custom steps that operate on integration data between connections. Each custom step performs one action. This is a step extension.

- A library resource that an integration runtime uses. For example, a library extension can provide a JDBC driver for connecting to a proprietary SQL database, such as Oracle.

- A single custom connector for creating connections to a particular application or service that you want to integrate. This is a connector extension.

> **NOTE**
>
> Fuse Online can use an OpenAPI document to create a connector for a REST API client. See Develop a REST API client connector .

A business integrator shares requirements with a developer who codes the extension. The developer provides a **.jar** file that contains the extension. The business integrator uploads the **.jar** file in Fuse Online to make the custom connector, custom step(s), or library resource available for use within Fuse Online.

An extension **.jar** file that you upload to Fuse Online always contains exactly one extension.

For an example of uploading and using an extension that provides a step that operates on data between connections, see the AMQ to REST API sample integration tutorial .

9.3.3. Overview of extension content and structure

An extension is a collection of classes, dependencies, and resources that are packaged in a **.jar** file.

Fuse Online uses Spring Boot to load an extension. Consequently, you must package an extension according to Spring Boot’s executable JAR format. For example, ensure that you use the **ZipEntry.STORED()** method to save a nested JAR file.

The structure of a **.jar** file that packages an extension is as follows:

```
extension.jar
|
+- META-INF
|  |
|  +- syndesis
|     |
|     +- syndesis-extension-definition.json 1
|
+- mycompany
|  |
|  +-project
|    |
|    +-YourClasses.class 2
|
+- lib 3
  |
  +-dependency1.jar
  |
  +-dependency2.jar
```

1 A JSON schema file that specifies the data structures that define the extension. This is referred to as the extension definition JSON file.

2 The Java classes that implement the behavior that the extension provides.

3 Additional dependencies that are required to build and execute the custom feature.

9.3.4. Requirements in an extension definition JSON file

Each extension must have a **.json** file that defines the extension by specifying values for data structures such as name, description, supported actions, and dependencies. For each extension type, the following table indicates whether Maven can generate the extension definition JSON file and which data structures are required.

Extension Type	Maven Can Generate Extension Definition	Required Data Structures
Step extension in Java	Yes	**schemaVersion** **name** **description** **version** **extensionId** **extensionType** **actions** **dependencies** *

Extension Type	Maven Can Generate Extension Definition	Required Data Structures
Step extension in XML	No	**schemaVersion** **name** **description** **version** **extensionId** **extensionType** **actions** **dependencies** *
Connector extension	No	**schemaVersion** **name** **description** **version** **extensionId** **extensionTypeproperties** **actions** **dependencies** * **componentScheme** **connectorCustomizers** **connectorFactory**
Library extension	No	**schemaVersion** **name** **description** **version** **extensionId** **extensionType** **dependencies** *

*While specification of **dependencies** is not strictly required, in practice, there are almost always dependencies that you need to specify.

Typically, an extension definition file has the following layout:

```
{
  "schemaVersion": "v1",
  "name": "",
  "description": "",
  "version": "",
  "extensionId": "",
  "extensionType": "",
  "properties": {
  },
  "actions": [
  ],
  "dependencies": [
  ],
}
```

- **schemaVersion** defines the version of the extension definition schema. Internally, Syndesis uses **schemaVersion** to determine how to map the extension definition to the internal model. This allows extensions that were developed against an old version of Syndesis to be deployed on newer versions of Syndesis.
- **name** is the name of the extension. When you upload an extension to Fuse Online, this name appears.
- **description** is any useful information that you want to specify. Fuse Online does not operate on this value.
- **version** is for your convenience to help you distinguish updates to an extension. Fuse Online does not operate on this value.
- **extensionId** defines a unique ID for the extension. This should be unique at least across a Syndesis environment.
- **extensionType** indicates to Syndesis what the extension provides. As of Syndesis version 1.3, the following extension types are supported:
  - **Steps**
  - **Connectors**
  - **Libraries**
- **properties** defines the global configuration options that are supported by the extension. Only connector extensions use properties that you specify. For example:

```
"propertyName": {
  "deprecated": true|false,
  "description": "",
  "displayName": "",
  "group": "",
  "kind": "",
  "label": "",
  "required": true|false,
  "secret": true|false,
  "javaType": "",
  "type": "",
  "defaultValue": "",
  "enum": {
  }
}
```

- **actions** defines the operations that a connector can perform or the operation that a step between connections can perform. Only connector and step extensions use actions that you specify. The format for an action specification looks like this:

```
{
    "id": "",
    "name": "",
    "description": "",
    "actionType": "step|connector",
    "descriptor": {
    }
}
```

  - **id** is a unique ID for the action. This should be unique at least within a Syndesis environment.
  - **name** is the action name that appears in Fuse Online. An integrator sees this value as the name of a connection action or as the name of a step that operates on integration data between connections.
  - **description** is the action description that appears in Fuse Online. Use this field to help the integrator understand what the action does.
  - **actionType** indicates whether the action is performed by a connection or a step that is between connections.
  - **descriptor** specifies nested attributes such as **kind**, **entrypoint**, **inputDataType**, **outputDatatype** and more.

- **dependencies** defines the resources that this extension requires Fuse Online to provide. Define a dependency as follows:

```
{
  "type": "MAVEN",
  "id"   : "org.apache.camel:camel-telegram:jar:2.21.0"
}
```

  - **type** indicates the type of the dependency. Specify **MAVEN**. (It is expected that other types will be supported in the future.)
  - **id** is the ID of the Maven dependency, which is a Maven GAV.

9.3.5. Description of Maven plugin that supports extensions

The **extension-maven-plugin** supports extension development by packaging the extension as a valid Spring Boot module. For step extensions that you implement in Java, this plugin can generate the extension definition JSON file.

In your Maven project's`pom.xml` file, add the following plugin declaration:

```
<plugin>
    <groupId>io.syndesis.extension</groupId>
    <artifactId>extension-maven-plugin</artifactId>
    <version>${syndesis.version}</version>
    <executions>
        <execution>
        <goals>
            <goal>generate-metadata</goal>
            <goal>repackage-extension</goal>
        </goals>
        </execution>
    </executions>
</plugin>
```

The **extension-maven-plugin** defines the following goals:

- **generate-metadata** generates the JSON extension definition file that will be in the generated JAR file as follows:

  a. Maven starts with the data structure specifications that are in the **META-INF/syndesis/syndesis-extension-definition.json** file, if there is one.
     If you are coding in XML, then you must define the extension definition JSON file yourself and it must specify all required data structures.

     If you are developing a connector or library extension, then you must define the extension definition JSON file yourself and it must specify all required data structures.

     If you are developing a step extension in Java, you can:

     - Create the extension definition JSON file yourself.
     - In your Java code, specify annotations that define all required data structures. You do not create an extension definition JSON file.
     - Create an extension definition JSON file and specify some but not all data structures.

  b. For step extensions that you develop in Java, Maven obtains missing specifications from code annotations

  c. Maven adds the dependencies list, which specifies dependencies that are provided with a scope of **provided** and that are managed through the **extension-bom**.

- **repackage-extension** packages the extension.
  - Dependencies and related transitive dependencies that are not managed through the **extension-bom** are in the **lib** folder of the generated JAR.
  - For library extensions, dependencies whose scope is **system** are in the **lib** folder of the generated JAR.

For example, suppose your Maven project has the following **pom.xml** file:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.company</groupId>
  <artifactId>my-extension</artifactId>
  <version>1.0.0</version>
  <name>MyExtension</name>
  <description>A Sample Extension</description>
  <packaging>jar</packaging>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>io.syndesis.extension</groupId>
        <artifactId>extension-bom</artifactId>
        <version>1.3.10</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>

  <dependencies>
    <dependency>
      <groupId>io.syndesis.extension</groupId>
      <artifactId>extension-api</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>com.github.lalyos</groupId>
      <artifactId>jfiglet</artifactId>
      <version>0.0.8</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.7.0</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
      <plugin>
        <groupId>io.syndesis.extension</groupId>
        <artifactId>extension-maven-plugin</artifactId>
        <version>1.3.10</version>
        <executions>
          <execution>
            <goals>
              <goal>generate-metadata</goal>
              <goal>repackage-extension</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

Based on this **pom.xml** file, the generated extension definition JSON file looks like this:

```
{
  "name": "MyExtension",
  "description": "A Sample Extension",
  "extensionId": "com.company:my-extension",
  "version": "1.0.0",
  "dependencies": [ {
    "type": "MAVEN",
    "id": "io.syndesis.extension:extension-api:jar:1.3.10"
  } ],
  "extensionType": "Libraries",
  "schemaVersion": "v1"
}
```

The generated archive has this structure and content:

```
my-extension-1.0.0.jar
|
+- lib
|  |
|  + jfiglet-0.0.8.jar
|
+- META-INF
  |
  +- MANIFEST.MF
     |
     +- syndesis
        |
        +- syndesis-extension-definition.json
```

9.3.6. How to specify data shapes in extensions

A data shape holds data type metadata for use by the data mapper. The data mapper transforms this metadata into internal documents that it uses to display the source and target data fields in the data mapper user interface. In an extension definition JSON file for a connector or for a custom step, each action specification defines an input data shape (**inputDataShape**) and an output data shape (**outputDataShape**).

When you are developing an extension, it is important to specify data shape properties that allow the data mapper to correctly handle and display the source and target fields. The following data shape properties affect data mapper behavior:

- **kind**
- **type**
- **specification**
- **name**
- **description**

About the kind property

The data shape **kind** property is represented by the **DataShapeKinds** enum. The possible values for the **kind** property are:

- **java** indicates that the data type is represented by a Java class. Follow the **"kind": "java"** declaration by specifying a fully qualified class name for the **type** property. For example:

  ```
  "outputDataShape": {
       "kind": "java",
       "type": "org.apache.camel.component.telegram.model.IncomingMessage"
  },
  ```

- **json-schema** indicates that the data type is represented by a JSON schema. When **kind** is set to **json-schema**, specify a JSON schema as the value of the data shape's **specification** property. For example:

  ```
  "inputDataShape": {
    "description": "Person data",
    "kind": "json-schema",
    "name": "Person",
    "specification": "{\"$schema\":\"http://json-schema.org/draft-
  04/schema#\",\"title\":\"Person\",\"type\":\"object\",\"properties\":{\"firstName\":{...}}}"
  }
  ```

  The code for the SAP Concur connector contains examples of data shapes that are specified by JSON schemas.

- **json-instance** indicates that the data type is represented by a JSON instance. When **kind** is set to **json-instance**, specify a JSON instance as the value of the data shape's **specification** property. For example:

  ```
  "inputDataShape": {
    "description": "Person data",
    "kind": "json-instance",
    "name": "Person",
    "specification": "{\"firstName\":\"John\",...}"
  }
  ```

- **xml-schema** indicates that the data type is represented by an XML schema. When **kind** is set to **xml-schema**, specify an XML Schema as the value of the data shape's **specification** property. For example:

  ```
  "inputDataShape": {
    "description": "Person data",
    "kind": "xml-schema",
    "name": "Person",
    "specification": "<?xml version=\"1.0\" encoding=\"UTF-8\" ?><xs:schema
  xmlns:xs=\"http://www.w3.org/2001/XMLSchema\">...</xs:schema>"
  }
  ```

- **xml-instance** indicates that the data type is represented by an XML instance. When **kind** is set to **xml-instance**, specify an XML instance as the value of the data shape's **specification** property. For example:

  ```
  "inputDataShape": {
    "description": "Person data",
    "kind": "xml-instance",
    "name": "Person",
    "specification": "<?xml version=\"1.0\" encoding=\"UTF-8\" ?><Person>
  <firstName>Jane</firstName></Person>"
  }
  ```

- **any** indicates that the data type is not structured. For example, it might be a byte array or free format text. The data mapper ignores a data shape when its **kind** property is set to **any**. In other words, the data does not appear in the data mapper and therefore you cannot map any fields to or from this data.

However, for a custom connector, when its **kind** property is set to **any**, Fuse Online prompts you to specify input and/or output data types when you configure a connection that you have created from the custom connector. This happens when you add a connection to an integration. You can specify the kind of the data shape's schema, an appropriate document for the kind of schema that you specify, and a name for the data type.

- **none** indicates that there is no data type. For an input data shape, this indicates that the connection or step does not read data. For an output data shape, this indicates that the connection or step does not modify data. For example, when an input message body is being transferred to an output message body, setting the **kind** property to **none** indicates that the data is only passing through. The data mapper ignores data shapes when **kind** is set to **none**. In other words, the data does not appear in the data mapper and therefore you cannot map any fields to or from this data.

About the **type** property

When the value of the **kind** property is **java**, the **"kind": "java"** declaration is followed by a **type** declaration that specifies a fully qualified Java class name. For example:

```
"outputDataShape": {
    "kind": "java",
    "type": "org.apache.camel.component.telegram.model.IncomingMessage"
},
```

When the **kind** property is set to anything other than **java** then any setting for the **type** property is ignored.

About the **specification** property

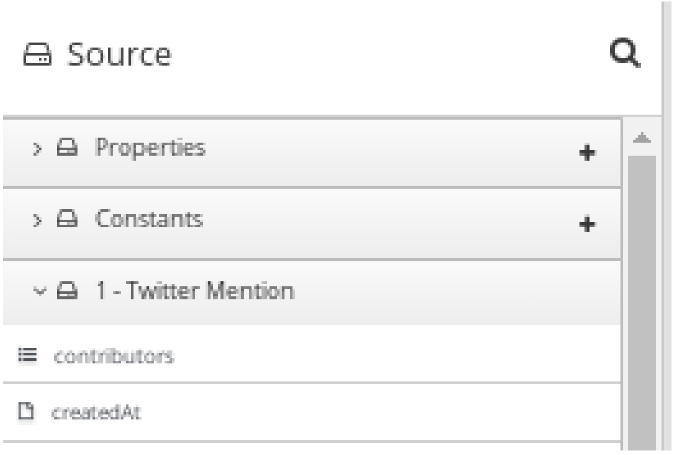
The setting of the **kind** property determines the setting of the **specification** property, as shown in the following table.

kind property setting	**specification** property setting
java	Java inspection result. For each extension that you write in Java, use **extension-maven-plugin** to at least obtain the Java inspection result. The plugin inserts the Java inspection result in the JSON extension definition file as the setting of the **specification** property. This is the only way to obtain the Java inspection result, which is required for data mapping in Fuse Online. As a reminder, for step extensions written in Java, **extension-maven-plugin** generates the JSON extension definition file and populates it with required content. For connector extensions, while **extension-maven-plugin** inserts the Java inspection result in the JSON extension definition file, you will need to manually add the required content that the plugin does not insert.

kind property setting	**specification** property setting
json-schema	An actual JSON schema document. The setting cannot be a reference to a document and the JSON schema cannot point to other JSON schema documents by means of references.
json-instance	An actual JSON document that contains example data. The data mapper derives the data types from the example data. The setting cannot be a reference to a document.
xml-schema	An actual XML schema document. The setting cannot be a reference to a document and the XML schema cannot point to other XML schema documents by means of references.
xml-instance	An actual XML instance document. The setting cannot be a reference to a document.
any	The **specification** property is not required. Any setting is ignored.
none	The **specification** property is not required. Any setting is ignored.

About the **name** property

The data shape **name** property specifies a human readable name for the data type. The data mapper displays this name in its user interface as the label for the data fields. In the following image, **Twitter Mention** is an example of where you would see the value of the **name** property.



This name also appears in data type indicators in the Fuse Online flow visualization panel.

About the **description** property

The data shape **description** property specifies text that appears as a tooltip when the cursor hovers over the data type name in the data mapper user interface.

9.3.7. Alternatives for developing step extensions

A step extension implements one or more custom steps. Each custom step implements one action for processing integration data between connections. The following examples demonstrate the alternatives for developing step extensions:

- Section 9.3.7.1, “Example of developing a Camel route with XML fragments”
- Section 9.3.7.2, “Example of developing a Camel route with RouteBuilder”
- Section 9.3.7.3, “Example of using a Camel bean”
- Section 9.3.7.4, “Example of using the Syndesis Step API”

Syndesis provides custom Java annotations that you can use in conjunction with the **syndesis-extension-plugin**. When you implement a step extension or a connector extension in Java, you can specify annotations that enable Maven to add action definitions to the extension definition JSON file. To enable annotation processing, add the following dependency to your Maven project:

```
<dependency>
  <groupId>io.syndesis.extension</groupId>
  <artifactId>extension-annotation-processor</artifactId>
  <optional>true</optional>
</dependency>
```

9.3.7.1. Example of developing a Camel route with XML fragments

To develop a custom step, you can implement the action as an XML fragment that is a Camel route that has an input such as **direct**. The Syndesis runtime invokes this route in the same way that it invokes any other Camel route.

For example, suppose that you want to create a step that logs the body of a message with an optional prefix. The following XML defines a Camel route that does this.

```
<?xml version="1.0" encoding="UTF-8"?>
<routes xmlns="http://camel.apache.org/schema/spring"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
      http://camel.apache.org/schema/spring
      http://camel.apache.org/schema/spring/camel-spring.xsd">

  <route id="log-body-with-prefix">
    <from uri="direct:log"/>
    <choice>
      <when>
        <simple>${header.prefix} != ''</simple>
        <log message="${header.prefix} ${body}"/>
      </when>
      <otherwise>
        <log message="Output ${body}"/>
      </otherwise>
    </choice>
  </route>

</routes>
```

When you develop an extension in XML, you must create the extension definition JSON file yourself. For this XML fragment, the **src/main/resources/META-INF/syndesis/syndesis-extension-definition.json** file could define the action as follows:

```
{
  "actionType": "step",
  "id": "log-body-with-prefix",
  "name": "Log body with prefix",
  "description": "A simple body log with a prefix",
  "descriptor": {
    "kind": "ENDPOINT", 1
    "entrypoint": "direct:log", 2
    "resource": "classpath:log-body-action.xml", 3
    "inputDataShape": {
      "kind": "none"
    },
    "outputDataShape": {
      "kind": "none"
    },
  "propertyDefinitionSteps": [ {
    "description": "extension-properties",
    "name": "extension-properties",
    "properties": { 4
      "prefix": {
        "componentProperty": false,
        "deprecated": false,
        "description": "The Log body prefix message",
        "displayName": "Log Prefix",
        "javaType": "String",
        "kind": "parameter",
        "required": false,
        "secret": false,
        "type": "string"
      }
    }
  } ]
  }
}
```

1 The type of the action is set to **ENDPOINT**. The runtime invokes a Camel endpoint to execute the action provided by this custom step.

2 The Camel endpoint to invoke is **direct:log**. This is the **from** specification in the route.

3 This is the location of the XML fragment.

4 These are the properties that the action defined in this custom step exposes to the integrator who will be adding this step to an integration. In Fuse Online, each value that the integrator specifies in the user interface gets mapped to a message header that has the same name as the property. In this example, the integrator will see one input field, with the **Log Prefix** display name.

WARNING

Syndesis does not support full Camel XML configuration. Syndesis supports only the <routes> tag.

9.3.7.2. Example of developing a Camel route with RouteBuilder

You can implement a custom step by developing an action as a Camel route with the support of the **RouteBuilder** class. Such a route has an input such as **direct**. Syndesis invokes this route in the same way that it invokes any other Camel route.

To implement the example that creates a step that logs the body of a message with an optional prefix, you can write something like this:

```
import org.apache.camel.builder.RouteBuilder;

import io.syndesis.extension.api.annotations.Action;
import io.syndesis.extension.api.annotations.ConfigurationProperty;

@Action( ①
    id = "log-body-with-prefix",
    name = "Log body with prefix",
    description = "A simple body log with a prefix",
    entrypoint = "direct:log")
public class LogAction extends RouteBuilder {
    @ConfigurationProperty( ②
        name = "prefix",
        description = "The Log body prefix message",
        displayName = "Log Prefix",
        type = "string")
    private String prefix;

    @Override
    public void configure() throws Exception {
        from("direct::start") ③
            .choice()
                .when(simple("${header.prefix} != ""))
                    .log("${header.prefix} ${body}")
                .otherwise()
                    .log("Output ${body}")
            .endChoice();
    }
}
```

① The **@Action** annotation indicates the action definition.

② The **@ConfigurationProperty** annotation indicates the property definition.

③ This is the action implementation.

This Java code uses Syndesis annotations, which means that the **extension-maven-plugin** can automatically generate the action definition. In the extension definition JSON file, the action definition will look like this:

```
{
  "id": "log-body-with-prefix",
  "name": "Log body with prefix",
  "description": "A simple body log with a prefix",
  "descriptor": {
    "kind": "ENDPOINT", ❶
    "entrypoint": "direct:log", ❷
    "resource": "class:io.syndesis.extension.log.LogAction", ❸
    "inputDataShape": {
      "kind": "none"
    },
    "outputDataShape": {
      "kind": "none"
    },
    "propertyDefinitionSteps": [ {
      "description": "extension-properties",
      "name": "extension-properties",
      "properties": { ❹
        "prefix": {
          "componentProperty": false,
          "deprecated": false,
          "description": "The Log body prefix message",
          "displayName": "Log Prefix",
          "javaType": "java.lang.String",
          "kind": "parameter",
          "required": false,
          "secret": false,
          "type": "string",
          "raw": false
        }
      }
    } ]
  },
  "actionType": "step"
}
```

❶ The type of action is **ENDPOINT**. The runtime invokes a Camel endpoint to execute the action that this step implements.

❷ This is the Camel endpoint to invoke. It is the **from** specification in the route.

❸ This is the class that implements **RoutesBuilder**.

❹ These are the properties that the action defined in this custom step exposes to the integrator who will be adding this step to an integration. In Fuse Online, each value that the integrator specifies in the user interface gets mapped to a message header that has the same name as the property. In this example, the integrator will see one input field, with the **Log Prefix** display name.

9.3.7.3. Example of using a Camel bean

You can implement a custom step by developing an action as a Camel bean processor. To implement the example that creates a step that logs the body of a message with an optional prefix, you can write something like this:

```
import org.apache.camel.Body;
import org.apache.camel.Handler;
import org.apache.camel.Header;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import io.syndesis.extension.api.annotations.Action;
import io.syndesis.extension.api.annotations.ConfigurationProperty;

@Action(
  id = "log-body-with-prefix",
  name = "Log body with prefix",
  description = "A simple body log with a prefix")
public class LogAction  {
  private static final Logger LOGGER = LoggerFactory.getLogger(LogAction.class);

  @ConfigurationProperty(
    name = "prefix",
    description = "The Log body prefix message",
    displayName = "Log Prefix",
    type = "string")
  private String prefix;

  @Handler 1
  public void process(@Header("prefix") String prefix, @Body Object body) {
    if (prefix == null) {
      LOGGER.info("Output {}", body);
    } else {
      LOGGER.info("{} {}", prefix, body);
    }
  }
}
```

1 This is the function that implements the action.

This Java code uses Syndesis annotations, which means that the **extension-maven-plugin** can automatically generate the action definition. In the extension definition JSON file, the action definition will look like this:

```
{
 "id": "log-body-with-prefix",
 "name": "Log body with prefix",
 "description": "A simple body log with a prefix",
 "descriptor": {
  "kind": "BEAN", 1
  "entrypoint": "io.syndesis.extension.log.LogAction::process", 2
  "inputDataShape": {
   "kind": "none"
  },
  "outputDataShape": {
   "kind": "none"
  },
  "propertyDefinitionSteps": [ {
   "description": "extension-properties",
   "name": "extension-properties",
   "properties": {
    "prefix": { ❸
     "componentProperty": false,
     "deprecated": false,
     "description": "The Log body prefix message",
     "displayName": "Log Prefix",
     "javaType": "java.lang.String",
     "kind": "parameter",
     "required": false,
     "secret": false,
     "type": "string",
     "raw": false
    }
   }
  } ]
 },
 "actionType": "step"
}
```

❶ The type of the action is **BEAN**. The runtime invokes a Camel bean processor to execute the action in this custom step.

❷ This is the Camel bean to invoke.

❸ These are the properties that the action defined in this custom step exposes to the integrator who will be adding this step to an integration. In Fuse Online, each value that the integrator specifies in the user interface gets mapped to a message header that has the same name as the property. In this example, the integrator will see one input field, with the **Log Prefix** display name.

When you use beans, you might find it convenient to inject user properties into the bean instead of retrieving them from the exchange header. To do this, implement getter and setter methods for the properties that you want to get injected. The action implementation would look like this:

```
import org.apache.camel.Body;
import org.apache.camel.Handler;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import io.syndesis.extension.api.annotations.Action;
import io.syndesis.extension.api.annotations.ConfigurationProperty;

@Action(
  id = "log-body-with-prefix",
  name = "Log body with prefix",
  description = "A simple body log with a prefix")
public class LogAction  {
  private static final Logger LOGGER = LoggerFactory.getLogger(LogAction.class);

  @ConfigurationProperty(
    name = "prefix",
        description = "The Log body prefix message",
        displayName = "Log Prefix",
        type = "string")
    private String prefix;

    public void setPrefix(String prefix) { 1
        this.prefix = prefix;
    }

    public String getPrefix() { 2
        return prefix;
    }

    @Handler
    public void process(@Body Object body) {
        if (this.prefix == null) {
            LOGGER.info("Output {}", body);
        } else {
            LOGGER.info("{} {}", this.prefix, body);
        }
    }
}
```

1 This is the property setter method.

2 This is the property getter method.

9.3.7.4. Example of using the Syndesis Step API

You can implement a custom step by using the Syndesis **Step** API. This provides a way to interact with runtime route creation. You can use any method provided by the **ProcessorDefinition** class and you can create more complex routes. Information for the Syndesis API is at http://javadoc.io/doc/io.syndesis.extension/extension-api.

Here is an example of a step extension that uses the Syndesis **Step** API to implement a split action:

```
import java.util.Map;
import java.util.Optional;

import io.syndesis.extension.api.Step;
import io.syndesis.extension.api.annotations.Action;
import io.syndesis.extension.api.annotations.ConfigurationProperty;
import org.apache.camel.CamelContext;
import org.apache.camel.model.ProcessorDefinition;
import org.apache.camel.util.ObjectHelper;
import org.apache.camel.Expression;
import org.apache.camel.builder.Builder;
import org.apache.camel.processor.aggregate.AggregationStrategy;
import org.apache.camel.processor.aggregate.UseOriginalAggregationStrategy;
import org.apache.camel.spi.Language;

@Action(id = "split", name = "Split", description = "Split your exchange")
public class SplitAction implements Step {
    @ConfigurationProperty(
        name = "language",
        displayName = "Language",
        description = "The language used for the expression")
    private String language;

    @ConfigurationProperty(
        name = "expression",
        displayName = "Expression",
        description = "The expression used to split the exchange")
    private String expression;

    public String getLanguage() {
        return language;
    }

    public void setLanguage(String language) {
        this.language = language;
    }

    public String getExpression() {
        return expression;
    }

    public void setExpression(String expression) {
        this.expression = expression;
    }

    @Override
    public Optional<ProcessorDefinition> configure(
            CamelContext context,
            ProcessorDefinition route,
            Map<String, Object> parameters) { 1

        String languageName = language;
        String expressionDefinition = expression;

        if (ObjectHelper.isEmpty(languageName) && ObjectHelper.isEmpty(expressionDefinition)) {
            route = route.split(Builder.body());
        } else if (ObjectHelper.isNotEmpty(expressionDefinition)) {

            if (ObjectHelper.isEmpty(languageName)) {
                languageName = "simple";
            }

            final Language splitLanguage = context.resolveLanguage(languageName);
            final Expression splitExpression = splitLanguage.createExpression(expressionDefinition);
            final AggregationStrategy aggreationStrategy = new UseOriginalAggregationStrategy(null,
false);

            route = route.split(splitExpression).aggregationStrategy(aggreationStrategy);
        }

        return Optional.of(route);
    }
}
```

1 This is the implementation of the action that the custom step performs.

This Java code uses Syndesis annotations, which means that the **extension-maven-plugin** can automatically generate the action definition. In the extension definition JSON file, the action definition will look like this:

```
{
  "id": "split",
  "name": "Split",
  "description": "Split your exchange",
  "descriptor": {
    "kind": "STEP", 1
    "entrypoint": "io.syndesis.extension.split.SplitAction", 2
    "inputDataShape": {
      "kind": "none"
    },
    "outputDataShape": {
      "kind": "none"
    },
    "propertyDefinitionSteps": [ {
      "description": "extension-properties",
      "name": "extension-properties",
      "properties": {
        "language": {
          "componentProperty": false,
          "deprecated": false,
          "description": "The language used for the expression",
          "displayName": "Language",
          "javaType": "java.lang.String",
          "kind": "parameter",
          "required": false,
          "secret": false,
          "type": "string",
          "raw": false
        },
        "expression": {
          "componentProperty": false,
          "deprecated": false,
          "description": "The expression used to split the exchange",
          "displayName": "Expression",
          "javaType": "java.lang.String",
          "kind": "parameter",
          "required": false,
          "secret": false,
          "type": "string",
          "raw": false
        }
      }
    } ]
  },
  "tags": [],
  "actionType": "step"
}
```

1 The type of the action is **STEP**.

2 This is the class that is implementing the **Step** interface.

9.3.8. Example of developing a connector extension

If Fuse Online does not provide a connector for the application or service that you want to connect to in an integration, an experienced developer can code an extension that contributes a new connector to Fuse Online.

IMPORTANT

For connector extensions, it is not yet possible to automatically generate the extension definition JSON file from Java code.

A connector is essentially a proxy for a Camel component. A connector configures the underlying component and creates endpoints according to options that are defined in the extension definition and in user-supplied options that the Fuse Online web interface collects.

The connector extension definition extends the extension definition that is required for step extensions with the following additional data structures:

- **componentScheme**
  Defines the Camel component that the connector uses. For example, **telegram**.

- **connectorCustomizers**
  Specifies a list of classes that implement the ComponentProxyCustomizer class. Each class customizes the behavior of a connector. For example, a class might manipulate properties before they are applied to the underlying component/endpoint, or a class might add pre/post endpoint logic. For each class, specify the full class name of the implementation, for example, **com.mycomponent.MyCustomizer**.

- **connectorFactory**
  Defines the class that implements the ComponentProxyFactory class, which creates and/or configures the underlying component/endpoint. Specify the full class name of the implementation.

You can set the above data structures at the action level or at the global level. Settings at the action level have precedence over the same items defined at global level.

Customizer example

The following customizer example sets up a **DataSource** from individual options:

```
public class DataSourceCustomizer implements ComponentProxyCustomizer, CamelContextAware {
    private final static Logger LOGGER = LoggerFactory.getLogger(DataSourceCustomizer.class);

    private CamelContext camelContext;

    @Override
    public void setCamelContext(CamelContext camelContext) { 1
        this.camelContext = camelContext;
    }
    @Override
    public CamelContext getCamelContext() { 2
        return this.camelContext;
    }

    @Override
    public void customize(ComponentProxyComponent component, Map<String, Object> options) {
        if (!options.containsKey("dataSource")) {
            if (options.containsKey("user") && options.containsKey("password") &&
options.containsKey("url")) {
                try {
                    BasicDataSource ds = new BasicDataSource();

                    consumeOption(camelContext, options, "user", String.class, ds::setUsername); 3
                    consumeOption(camelContext, options, "password", String.class, ds::setPassword); 4
                    consumeOption(camelContext, options, "url", String.class, ds::setUrl); 5

                    options.put("dataSource", ds);
                } catch (@SuppressWarnings("PMD.AvoidCatchingGenericException") Exception e) {
                    throw new IllegalArgumentException(e);
                }
            } else {
                LOGGER.debug("Not enough information provided to set-up the DataSource");
            }
        }
    }
}
```

1 2 By implementing **CamelContextAware**, Syndesis injects the Camel context and then invoking the customize method.

3 4 5 Processes options and then removes them from the options map.

Example of injecting properties

If the customizer respects Java bean conventions, you can also inject the properties, as shown in this revision of the previous example:

```
public class DataSourceCustomizer implements ComponentProxyCustomizer, CamelContextAware {
    private final static Logger LOGGER = LoggerFactory.getLogger(DataSourceCustomizer.class);

    private CamelContext camelContext;
    private String userName;
    private String password;
    private String url;

    @Override
    public void setCamelContext(CamelContext camelContext) { 1
        this.camelContext = camelContext;
    }

    @Override
    public CamelContext getCamelContext() { 2
        return this.camelContext;
```

```
  }

  public void setUserName(String userName) { ❸
   this.userName = userName;
  }

  public String getUserName() { ❹
   return this.userName;
  }

  public void setPassword(String password) { ❺
   this.password = password;
  }

  public String getPassword() { ❻
   return this.password;
  }

  public void setUrl(String url) { ❼
   this.url = url;
  }

  public String getUrl() { ❽
   return this.url;
  }

  @Override
  public void customize(ComponentProxyComponent component, Map<String, Object> options) {
    if (!options.containsKey("dataSource")) {
      if (userName != null && password != null && url != null) {
        try {
          BasicDataSource ds = new BasicDataSource();
          ds.setUserName(userName);
          ds.setPassword(password);
          ds.setUrl(url);

          options.put("dataSource", ds);
        } catch (@SuppressWarnings("PMD.AvoidCatchingGenericException") Exception e) {
          throw new IllegalArgumentException(e);
        }
      } else {
        LOGGER.debug("Not enough information provided to set-up the DataSource");
      }
    }
  }
}
```

❶ ❷ ❸ By implementing **CamelContextAware**, Syndesis injects the Camel context and then invokes the customize method. This sample code overrides the **setCamelContext()** and **getCamelContext()** methods, and sets the user name.

❹ ❺ ❻ ❼ ❽ Starting here, the sample code processes the injected options and automatically removes them from the options map.

Using a customizer to configure before/after logic

You can use a customizer to configure before/after logic as shown in this example:

```
public class AWSS3DeleteObjectCustomizer implements ComponentProxyCustomizer {
  private String filenameKey;

  public void setFilenameKey(String filenameKey) {
    this.filenameKey = filenameKey;
  }

  public String getFilenameKey() {
     return this.filenameKey;
  }

  @Override
  public void customize(ComponentProxyComponent component, Map<String, Object> options) {
    component.setBeforeProducer(this::beforeProducer);
  }

  public void beforeProducer(final Exchange exchange) throws IOException {
    exchange.getIn().setHeader(S3Constants.S3_OPERATION, S3Operations.deleteObject);

    if (filenameKey != null) {
      exchange.getIn().setHeader(S3Constants.KEY, filenameKey);
    }
  }
}
```

Customizing behavior of **ComponentProxyComponent**

To customize the behavior of the ComponentProxyComponent class, you can override any of the following methods:

- **createDelegateComponent()**
  Syndesis invokes this method when the proxy starts and it is used to eventually create a dedicated instance of the component with the scheme defined by the **componentScheme** option.

  The default behavior of this method is to determine if any of the connector/action options applies at the component level. If the same option cannot be applied at the endpoint, and only in this case, the method creates a custom component instance and configures it according to the applicable options.

- **configureDelegateComponent()`**
  Syndesis invokes this method only if a custom component instance has been created to configure additional behavior of the delegated component instance.

- **createDelegateEndpoint()**
  Syndesis invokes this method when the proxy creates the endpoint and by default creates the endpoint by using Camel catalog facilities.

- **configureDelegateEndpoint()**
  After the delegated endpoint has been created, Syndesis invokes this method to configure additional behavior of the delegated endpoint instance, for example:

```
public class IrcComponentProxyFactory implements ComponentProxyFactory {

    @Override
    public ComponentProxyComponent newInstance(String componentId, String
componentScheme) {
        return new ComponentProxyComponent(componentId, componentScheme) {
            @Override
            protected void configureDelegateEndpoint(ComponentDefinition definition, Endpoint
endpoint, Map<String, Object> options) throws Exception {
                if (!(endpoint instanceof IrcEndpoint)) {
                    throw new IllegalStateException("Endpoint should be of type IrcEndpoint");
                }

                final IrcEndpoint ircEndpoint = (IrcEndpoint)endpoint;
                final String channels = (String)options.remove("channels");

                if (ObjectHelper.isNotEmpty(channels)) {
                    ircEndpoint.getConfiguration().setChannel(
                        Arrays.asList(channels.split(","))
                    );
                }
            }
        };
    }
}
```

9.3.9. How to develop library extensions

A library extension provides a resource that an integration requires at runtime. A library extension does not contribute steps or connectors to Fuse Online.

Support for library extensions is evolving. In this release, in the Fuse Online web interface:

- When you create an integration, you cannot select which library extension(s) an integration should include.

- When you add a database connection to an integration, Fuse Online adds all extensions that have the **jdbc-driver** tag to the integration runtime.

A library extension does not define any actions. Here is a sample definition for a library extension:

```
{
  "schemaVersion" : "v1",
  "name" : "Example JDBC Driver Library",
  "description" : "Syndesis Extension for adding a custom JDBC Driver",
  "extensionId" : "io.syndesis.extensions:syndesis-library-jdbc-driver",
  "version" : "1.0.0",
  "tags" : [ "jdbc-driver" ],
  "extensionType" : "Libraries"
}
```

Other than the lack of actions, the structure of a library extension is the same as the structure of a step or connector extension.

In a Maven project that creates a library extension, to add dependencies that are not available from a Maven repository, specify a **system** dependency, for example:

```
<dependency>
    <groupId>com.company</groupId>
    <artifactId>jdbc-driver</artifactId>
    <version>1.0</version>
    <scope>system</scope>
    <systemPath>${project.basedir}/lib/jdbc-driver-1.0.jar</systemPath>
</dependency>
```

9.3.10. Creating JDBC driver library extensions

To connect to a SQL database other than Apache Derby, MySQL, and PostgreSQL, you can create a library extension that wraps a JDBC driver for the database you want to connect to. After uploading this extension to Fuse Online, the Fuse Online-provided **Database** connector can access the driver to validate and create connections to the proprietary database. You do not create a new connector for your particular database.

The Syndesis open source community provides a project for creating an extension that wraps a JDBC driver.

Package one driver only in an extension. This makes it easier to manage the extension as part of managing your particular database. However, it is possible to create a library extension that wraps more than one driver.

Prerequisites

To use the Syndesis project, you must have a GitHub account.

Procedure

1. Ensure access to the JDBC driver for the database you want to connect to by doing one of the following:

    a. Confirm that the driver is in a Maven repository.

    b. Download the driver.

2. In a browser tab, go to https://github.com/syndesisio/syndesis-extensions

3. Fork the **syndesis-extensions** repository to your GitHub account.

4. Create a local clone from your fork.

5. In your **syndesis-extensions** clone:

    a. If the driver is not in a Maven repository, copy the driver into the **syndesis-library-jdbc-driver/lib** folder.

    b. Edit the **syndesis-library-jdbc-driver/pom.xml** file:

        i. Update the value of the **Name** element to be a name that you choose for this extension.

        ii. Update the value of the **Description** element to provide helpful information about this extension.

iii. If the driver is in a Maven repository, ensure that a reference to that Maven repository is in the **pom.xml** file.

iv. Examine the rest of the content of the **pom.xml** file and change any relevant metadata as needed.

c. In the **syndesis-library-jdbc-driver** folder, execute **mvn clean package** to build the extension.

The generated **.jar** file is in the **syndesis-library-jdbc-driver/target** folder. Provide this **.jar** file for uploading to Fuse Online.

NOTE

Fuse Online does not yet offer a way to select which library extension(s) an integration should include. When you add a database connection to an integration, Fuse Online adds all extensions that have the **jdbc-driver** tag to integration runtime. This is expected to improve in a future release.

9.4. ADDING AND MANAGING EXTENSIONS

Extensions let you add customizations to Fuse Online so you can integrate applications the way you want to. After you start using customizations provided in extensions, you can identify the integrations that use those customizations. This is helpful to do before you update or delete an extension.

The following topics provide details:

- Section 9.4.1, “Making custom features available”
- Section 9.4.2, “Identifying integrations that use extensions”
- Section 9.4.3, “Updating extensions”
- Section 9.4.4, “Deleting extensions”

9.4.1. Making custom features available

To make a custom feature available for use in an integration, upload the extension to Fuse Online.

Prerequisite

A developer has provided a **.jar** file that contains a Fuse Online extension.

Procedure

1. In the left Fuse Online panel, click **Customizations**.
2. At the top, click **Extensions**.
3. Click **Import Extension**.
4. Drag and drop, or choose, the **.jar** file that contains the extension that you want to upload. Fuse Online immediately tries to validate that the file contains an extension. If there is a problem, Fuse Online displays a message about the error. You must coordinate with the extension developer to obtain an updated **.jar** file, which you can then try to upload.

5. Review the extension details.
   After Fuse Online validates the file, it extracts and displays the extension's name, ID, description, and type. The type indicates whether the extension defines a custom connector, or one or more custom steps for operating on data between connections, or a JDBC driver for a proprietary database. An extension that provides a JDBC driver is referred to as a library extension.

   For a connector extension, Fuse Online displays the actions that are available to a connection that is created from this custom connector. In the extension, the developer might have provided an icon that Fuse Online can use to represent the application connections created from this connector. While you do not see this icon in the extension details page, it appears when you create connections from the custom connector. If the extension developer did not provide an icon in the extension, then Fuse Online generates an icon.

   For a step extension, Fuse Online displays the name of each custom step that the extension defines.

   For a library extension, if this extension contains a newer version of a JDBC driver that you previously uploaded, then you must remove the older version from your classpath. Ensure that your classpath has only one version of this driver and that the reference is to the newer driver you are uploading. Integrations that are running and that use connections based on the older driver are not affected. New connections that you create will use the new driver. If you start an integration that has a connection that was created with the older driver, Fuse Online automatically uses the new driver instead.

6. Click **Import Extension**. Fuse Online makes the custom connector or custom step(s) available and displays the extension's details page.

Additional resources

- Creating a connection from a custom connectors .
- Adding a custom step
- Connecting to a proprietary database

9.4.2. Identifying integrations that use extensions

Before you update or delete an extension, you should identify the integrations that use customizations that are provided by that extension.

Procedure

1. In the left Fuse Online panel, click **Customizations**.
2. At the top, click **Extensions**.
3. In the list of extensions, find the entry for the extension that you want to update or delete and click it.

Result

Fuse Online displays details about the extension including a list of any integrations that use a customization provided by the extension.

9.4.3. Updating extensions

When a developer updates an extension, you can upload the updated **.jar** file to implement the updates in your integrations.

Prerequisite

A developer has provided you with an updated **.jar** file for an extension that you previously uploaded.

Procedure

1. In Fuse Online, in the left panel, click **Customizations**.
2. Click the **Extensions** tab.
3. At the right of the entry for the extension that you want to update, click **Update**.
4. Click **Browse**, select the updated **.jar** file, and click **Open**.
5. Confirm that the extension details are correct and click **Update**.
6. In the details page for the updated extension, determine which integrations use the connector or custom step(s) defined in the extension.

It is up to you to know exactly what is required to update each integration that uses a custom connector or a custom step from the updated extension. At the very least, you must republish each integration that uses a customization defined in the updated extension.

In some cases, you might need to edit the integration to change or add configuration details for a customization. You must communicate with the extension developer to understand how to update integrations.

9.4.4. Deleting extensions

You can delete an extension even if a running integration uses a step that is provided by that extension or uses a connection that was created from a connector that was provided by that extension. After you delete an extension, you cannot start an integration that uses a customization that was provided by that extension.

Procedure

1. In the left Fuse Online panel, click **Customizations**.
2. At the top, click **Extensions**.
3. In the list of extensions, find the entry for the extension that you want to delete and click **Delete**, which appears at the right of the entry.

Results

There might be stopped or draft integrations that use a customization provided by an extension that you delete. To run one of these integrations, you will need to edit the integration to remove the customization.

Additional resources

- Identifying extension use
- Updating an extension

CHAPTER 10. INSTALLING AND MANAGING FUSE ONLINE ON OPENSHIFT CONTAINER PLATFORM

You can install and manage Fuse Online on OpenShift Container Platform (OCP) on-site. When Fuse Online is running on OCP on-site, additional features are available beyond the features that are provided when Fuse Online is running on OpenShift Online or on OpenShift Dedicated. See the following topics for details:

- Section 10.1, "Installation of Fuse Online on OCP"
- Section 10.2, "Bypassing the OpenShift access authorization page"
- Section 10.3, "Configuring Fuse Online to enable 3scale discovery of APIs"
- Section 10.4, "Monitoring Fuse Online integrations on OCP with Prometheus"
- Section 10.5, "How to invoke Fuse Online public REST API endpoints"
- Section 10.6, "Using external tools to export/import integrations for CI/CD"
- Section 10.7, "Upgrading Fuse Online on OCP"
- Section 10.8, "Uninstalling Fuse Online from an OCP project"
- Section 10.9, "Deleting an OCP project that contains Fuse Online"
- Section 10.10, "Fuse Online public REST API endpoints reference"
- Section 10.11, "Enabling Apache Camel K to speed up Fuse Online deployments on OCP"

10.1. INSTALLATION OF FUSE ONLINE ON OCP

The following topics provide details about how to install Fuse Online on OCP on-site:

- Section 10.1.1, "Overview of steps for installing Fuse Online on OCP"
- Section 10.1.2, "Registering a custom resource definition for deploying Fuse Online resources"
- Section 10.1.3, "Decisions to make before you install Fuse Online on OCP"
- Section 10.1.4, "Installing Fuse Online on OCP"

10.1.1. Overview of steps for installing Fuse Online on OCP

To install Fuse Online on OCP on-site, the main steps are:

1. A user with cluster administration permissions:
    a. Downloads the installation script.
    b. Registers a custom resource definition (CRD) at the cluster level.
    c. Grants permission for a user to install Fuse Online in their projects.
2. A user who is granted permission to install Fuse Online:

a. Ensures that all prerequisites are met.

b. Decides how to install Fuse Online with regard to the OpenShift project to install into, the OpenShift route for Fuse Online, and the level of accessibility of OpenShift logs.

c. Downloads the installation script.

d. Invokes the installation script with the command that implements decisions.

e. Confirms that Fuse Online is running.

10.1.2. Registering a custom resource definition for deploying Fuse Online resources

To enable installation of Fuse Online, a cluster administrator registers a custom resource definition. The administrator needs to do this only once for the OpenShift cluster. The administrator also grants permission for installing Fuse Online to the appropriate users. Each user can then install Fuse Online in their projects.

Prerequisites

- You must have cluster administration permissions.
- You must be connected to the OCP cluster into which Fuse Online will be installed.

Procedure

1. Download the package containing the Fuse Online installation scripts from the following location:
   https://github.com/syndesisio/fuse-online-install/releases/tag/1.6

   Unpack the downloaded archive at a convenient location on your file system. The **fuse-online-install-1.6** directory contains the scripts and supporting files for installing Fuse Online.

2. To verify that you are properly connected and can list custom resource definitions, invoke the following command:
   $ oc get crd

3. To register the custom resource definition at the cluster level, go to the **fuse-online-install-1.6** directory and invoke the following command:
   $ bash install_ocp.sh --setup

4. Grant installation permission to each user who needs to install Fuse Online. For example, suppose you want to grant permission to an account with the user name **developer**. The following command grants permission to **developer** to install Fuse Online into any project that the **developer** account can access on the currently connected cluster:
   $ bash install_ocp.sh --grant developer --cluster

   Repeat this command for each user account that needs permission to install Fuse Online.

10.1.3. Decisions to make before you install Fuse Online on OCP

To correctly specify the Fuse Online installation command, decide on the answers to these questions:

- **Do you want to install Fuse Online into the current OpenShift project or do you want to specify the project into which you want to install Fuse Online in the command that you invoke to do the installation?**
  The default is that the installation script installs Fuse Online into the current project.

  You can install Fuse Online into a project that you specify. If this project does not yet exist, then the script creates it. If this project exists then the installation script prompts you to confirm that it is okay to delete the project's content. To continue, you must confirm. The installation script then deletes the project and re-creates it.

- **Do you want the installation script to calculate the OpenShift route by which Fuse Online can be reached or do you want to specify the OpenShift route in the installation command?**
  The default is that the installation script calculates the route.

- **Do you want to be able to access OpenShift log information for Fuse Online integrations by clicking a link in the Fuse Online user interface?**
  While you can always access OpenShift logs manually, you might want to enable direct access by means of a link in the Fuse Online user interface. The default is that the installation script does not enable this link. To enable this link, you specify the URL for your OpenShift console when you invoke the installation script.

10.1.4. Installing Fuse Online on OCP

To install Fuse Online on OCP on-site, download the installation package, run it, and confirm that Fuse Online is installed.

Prerequisites

- You are running OCP on-site.

- You installed the **oc** client tool and it is connected to the OCP cluster in which you want to install Fuse Online.

- A user with cluster administration permissions gave you permission to install Fuse Online in any project that you have permission to access in the cluster.

- You decided whether to perform the default installation or specify options to customize the installation.

- You have a Red Hat developer account for which you know your user name and password. The installation script prompts you for these credentials so it can authenticate you against **https://developers.redhat.com**. For details about creating an account, see Accessing and Configuring the Red Hat Registry.

Procedure

1. Download the package containing the Fuse Online installation scripts from the following location:
   https://github.com/syndesisio/fuse-online-install/releases/tag/1.6

   Unpack the downloaded archive at a convenient location on your file system. The **fuse-online-install-1.6** directory contains the scripts and supporting files for installing Fuse Online.

2. Log in to OpenShift with an account that has permission to install Fuse Online. For example:

   ```
   $ oc login -u developer -p developer
   ```

3. Ensure that the current project is the project into which you want to install Fuse Online. To view the current project:
   $ oc project

4. In the directory in which you downloaded the installation script, invoke the installation script:

   - To perform the default installation, invoke:
     $ bash install_ocp.sh

   - To specify installation options, you might invoke something like the following command, which specifies three options:

     ```
     $ bash install_ocp.sh \
        --project my-project \
        --route my-project.6a63.fuse-online.openshiftapps.com \
        --console https://console.fuse-online.openshift.com/console
     ```

     - **--project** specifies the name of the OpenShift project in which you want to install Fuse Online. You need to specify this option only when you want to install Fuse Online into an OpenShift project other than the current OpenShift project.

     - **--route** specifies the URL that your Fuse Online environment will have. This URL will be where you access your Fuse Online environment. This URL specifies the project in which you are installing Fuse Online followed by the domain that is specific to your OpenShift cluster.
       In other words, specify the project name followed by the hostname that is associated with the OpenShift route that is used to access Fuse Online. The DNS resolution of the specified hostname must point to the OpenShift Router.

     - **--console** specifies the full URL for your OpenShift console.

   To learn more about the installation script options, invoke the **$ bash install_ocp.sh --help** command.

5. Confirm that installation was successful:

   a. Display the OpenShift OAuth proxy log-in page at **https://openshift-route**.
      If you specified the **--route** option when you ran the installation script, replace **openshift-route** with the route name that you specified. If you chose to let the installation script calculate the OpenShift route, then the script displays the calculated route near the end of its execution. Replace **openshift-route** with the value that the script provided.

   b. If you are not already logged in to the OpenShift console, its log-in page appears. Enter your OpenShift user name and password to log in.

   The Fuse Online home page appears either immediately or after you log in to the OpenShift console.

10.2. BYPASSING THE OPENSHIFT ACCESS AUTHORIZATION PAGE

By default, when users access Fuse Online, they need to log in to OpenShift and then authorize access from Fuse Online to their OpenShift account. If you want to, you can configure OpenShift so that users bypass the page that prompts for access authorization.

To do this, a cluster administrator creates an **OAuthClient** resource. Then the administrator, or any user with permission to modify objects in the namespace in which Fuse Online is installed, updates the

syndesis-oauthproxy **DeploymentConfig** resource. The update changes the container arguments to refer to the OAuth client ID and secret in the **OAuthClient** resource.

Prerequisites

- Fuse Online is installed on OCP on-site.
- A user with an OpenShift account that has the privileges that are associated with members of the **system:cluster-admins** group must create an **OAuthClient** resource. To obtain a list of the accounts that have this ability, run:

  ```
  oc policy who-can create OAuthClient
  ```

Procedure

1. Create an **OAuthClient** resource with the following content and format:

   ```
   apiVersion: oauth.openshift.io/v1
   kind: OAuthClient
   grantMethod: auto
   metadata:
     name: ${CLIENT_ID}
   redirectURIs:
   - https://${ROUTE_HOSTNAME}/oauth/callback
   secret: ${SECRET}
   ```

2. In the **OAuthClient** resource:

   a. Replace **${CLIENT_ID}** with a unique string that distinguishes a particular Fuse Online installation. For example, you might prefix the OpenShift namespace in which Fuse Online is installed: **fuse-online-proj123456**.
   Or, you might specify something like **fuse-online-1**. Typically, a name that indicates where the resource is used is most helpful. Then when an administrator lists OAuth clients, the purpose of each client is clear.

   b. Replace **${ROUTE_HOSTNAME}** with the fully qualified hostname that is configured for the Fuse Online route. For example, **fuse-online.mycorp.com**.

   c. Replace **${SECRET}** with a sequence of random alphanumeric characters that will be used to authenticate Fuse Online requests to OpenShift.
   For example:

   ```
   apiVersion: oauth.openshift.io/v1
   kind: OAuthClient
   grantMethod: auto
   metadata:
     name: fuse-online-1
   redirectURIs:
   - https://fuse-online.mycorp.com/oauth/callback
   secret: IFCbZTwrTOdZi1h86azxoixXkN7wnsUf3gjoSKbb
   ```

3. Run the following command to update the **syndesis-oauthproxy** **DeploymentConfig** resource:

   ```
   oc patch dc syndesis-oauthproxy --type json --patch "[{\"op\": \"replace\", \"path\":
   ```

```
\"/spec/template/spec/containers/0/args/1\", \"value\": \"--client-id=$CLIENT_ID\"}, {\"op\":
\"replace\", \"path\": \"/spec/template/spec/containers/0/args/2\", \"value\": \"--client-
secret=$CLIENT_SECRET\"}]"
```

When you installed Fuse Online, the installation script created the **syndesis-oauthproxy** **DeploymentConfig** resource. This **oc patch** command updates the resource so that its container arguments refer to the OAuth client ID and the OAuth secret defined in the new **OAuthClient** resource. For example, the updated resource looks like this:

```
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  name: syndesis-oauthproxy
spec:
  template:
    spec:
      containers:
      - args:
        - --client-id=fuse-online-1
        - --client-secret=lFCbZTwrTOdZi1h86azxoixXkN7wnsUf3gjoSKbb
```

Result

When users try to display the Fuse Online console, they log in to OpenShift and the Fuse Online console displays in the browser. The user does not have to authorize access from Fuse Online to OpenShift.

10.3. CONFIGURING FUSE ONLINE TO ENABLE 3SCALE DISCOVERY OF APIS

If you create an API provider integration, you might want to enable discovery of the API for that integration in 3scale. For Fuse Online environments that are installed on OCP, a user with cluster administration permissions can enable 3scale discovery of APIs by setting a Fuse Online server configuration environment variable.

The default behavior is that APIs are not exposed for automatic discovery in 3scale. Turning on 3scale service discovery applies to all API provider integrations in your Fuse Online environment. You cannot choose which APIs are discoverable.

When 3scale service discovery is turned on:

- Fuse Online does not provide an external URL for an API provider integration that is running.
- The API is accessible only through 3scale. Configuration in 3scale is required to expose the endpoint. For details, see Red Hat 3scale API Management, Service Discovery .

Prerequisites

- Fuse Online is installed on OCP on-site.
- You have cluster administration permissions.

Procedure

Edit the **syndesis-server** **DeploymentConfig** object to set the **CONTROLLERS_EXPOSE_VIA3SCALE** environment variable to **true**. To do this, enter the following command on one line:

```
$ oc patch dc syndesis-server -p '{"spec":{"template":{"spec":{"containers":[{"name":"syndesis-server","env":[{"name":"CONTROLLERS_EXPOSE_VIA3SCALE","value":"true"}]}]}}}}'
```

10.4. MONITORING FUSE ONLINE INTEGRATIONS ON OCP WITH PROMETHEUS

When you install Fuse Online on OCP on-site, the environment includes an instance of Prometheus in the OpenShift project in which you installed Fuse Online. You can access the Prometheus dashboard to monitor Fuse Online integrations.

Prerequisites

- You are running OCP on-site.
- Fuse Online is installed in an OpenShift project.

Procedure

1. Log in to the OpenShift console.
2. Open the project in which you installed Fuse Online.
3. In the left pane of the OpenShift console, select **Applications** > **Routes**.
4. Click the **syndesis-prometheus** Hostname URL to open the Prometheus dashboard in a new browser tab or window.

Additional resources

- For information about getting started with Prometheus, go to: https://prometheus.io/docs/prometheus/latest/getting_started/
- For information about how to configure an existing external Prometheus instance to monitor Fuse Online, see the section on Prometheus in Managing Fuse.

10.5. HOW TO INVOKE FUSE ONLINE PUBLIC REST API ENDPOINTS

When you are running Fuse Online on OCP, each Fuse Online environment can expose public REST API endpoints. External tools can invoke these endpoints to operate on the resources that are in that Fuse Online environment. The command that invokes an API endpoint specifies the Fuse Online environment on which the endpoint operates.

To invoke Fuse Online public REST API endpoints, you must expose the endpoints in each Fuse Online environment and you must have an understanding of how to specify a command that invokes an endpoint. See the following topics for details:

- Section 10.5.1, “Exposing Fuse Online public REST APIs for use by external tools”
- Section 10.5.2, “Description of base URL for Fuse Online public REST API endpoints”
- Section 10.5.3, “Obtaining a secret token for calling a public REST API endpoint”

- Section 10.5.4, “How to find integration IDs”
- Section 10.5.5, “Format for specifying **curl** commands to invoke Fuse Online public endpoints”

10.5.1. Exposing Fuse Online public REST APIs for use by external tools

When you are running Fuse Online on OCP on-site, you might want to use an external tool to copy Fuse Online integrations from one Fuse Online environment to another Fuse Online environment. An external tool might be a Jenkins job, an Ansible playbook, a **cron**-based shell script, or something else. For example, an Ansible playbook can export an integration from a Fuse Online development environment and import it into a Fuse Online testing environment.

To enable this, you must expose Fuse Online public REST API endpoints for each Fuse Online environment. In other words, you must repeat the procedure here for each installation of Fuse Online. For Fuse Online installations on the same cluster, while you can use the same service account to expose the public APIs, the recommendation is to use a different service account for each installation for better security.

Prerequisites

- You have an OCP project in which Fuse Online is installed.
- You use an external CI/CD tool and you want it to copy tagged integrations from one Fuse Online environment to another.
- The user who creates the OpenShift public OAuth proxy application must have cluster administration privileges.

Procedure

1. Create an OpenShift service account that has a name that you specify. For example, the following command creates the **cicd-client** service account:

   ```
   $ oc create serviceaccount cicd-client
   ```

2. Give your new service account permission to access the Fuse Online public API. For example, if **cicd-client** is the name of your new service account, then you would invoke the following command:

   ```
   $ oc policy add-role-to-user edit system:serviceaccount:syndesis:cicd-client
   ```

3. Invoke the following command to create an OpenShift template for exposing Fuse Online public REST API endpoints:

   ```
   $ oc create -f https://raw.githubusercontent.com/syndesisio/fuse-online-install/1.6.x/resources/syndesis-public-oauth-proxy.yml
   ```

   External tools can use these endpoints to export and import Fuse Online integrations across Fuse Online clusters. These integrations must have been marked for CI/CD pipelines.

   > NOTE
   >
   > In the template, the name of the resource is **syndesis-public-oauth-proxy**, while the name of the template itself is **syndesis-public-oauthproxy**. As you can see, the resource name hyphenates **oauth** and **proxy** but the template file name does not.

4. Invoke the following command to create an OpenShift application that runs a Fuse Online public OAuth proxy process:

   ```
   $ oc new-app --template=syndesis-public-oauthproxy \
       -p PUBLIC_API_ROUTE_HOSTNAME=EXTERNAL_HOSTNAME \
       -p OPENSHIFT_PROJECT=$(oc project -q) \
       -p OPENSHIFT_OAUTH_CLIENT_SECRET=$(oc sa get-token syndesis-oauth-client) \
       -p SAR_PROJECT=$(oc project -q)
   ```

   Replace **EXTERNAL_HOSTNAME** with the public address of your Fuse Online environment. For example, the address might be something like **public-fuse-online.127.0.0.1.nip.io**.

   This command uses the **syndesis-public-oauthproxy** template, which you created in the previous step, to create an OAuth proxy process that exposes the Fuse Online public REST API. This OAuth proxy allows access to the Fuse Online public API at a new public network address by using an OpenShift service account token.

Result

Fuse Online backend server public REST API endpoints are available to be called by external CI/CD tools.

Next steps

- Mark integrations for export to other Fuse Online environments.
- Configure external tools to call the Fuse Online public REST API endpoints.

10.5.2. Description of base URL for Fuse Online public REST API endpoints

The base URL for Fuse Online public REST API endpoints is something like this:

https://public-syndesis.192.168.64.42.nip.io/api/v1/public

The first part of the base URL is different for each Fuse Online environment. When you create the OpenShift application that runs the Fuse Online public OAuth proxy that enables access to public REST API endpoints, you specify the public address of your Fuse Online environment. This address is the first part of the base URL for calling endpoints that operate in that Fuse Online environment. For example:

https://public-syndesis.192.168.64.42.nip.io

The second part of the base URL is the same for all Fuse Online environments:

/api/v1/public

The Fuse Online public REST API provides endpoints that operate on three resources:

- **/integrations** are the integrations that are in the Fuse Online environment that is identified in the base URL.
- **/connections** are the connections that are in the Fuse Online environment that is identified in the base URL.
- **/environments** is the set of environment labels that are in the Fuse Online environment that is identified in the base URL.

10.5.3. Obtaining a secret token for calling a public REST API endpoint

A command that calls a Fuse Online public REST API endpoint must specify a secret token. This token is for the service account that you created when you exposed the Fuse Online public REST API in a given Fuse Online environment.

Prerequisites

- You are running Fuse Online on OCP on-site.
- You exposed the public REST API that is provided by a Fuse Online environment in which you want to invoke endpoints.

Procedure

1. Obtain the names of the secret tokens for the service account that you created when you exposed the public REST API for this Fuse Online environment. For example, if **cicd-client** is the name of the service account, you would invoke the following command:
   $ oc describe serviceaccount cicd-client

   This displays a list of information about the **cicd-client** service account including the names of its two tokens, which looks something like this:

   ```
   Tokens:      cicd-client-token-gxb25
                cicd-client-token-gxdnv
   ```

2. Display the content of either one of the tokens. For example:
   $ oc describe secret cicd-client-token-gxb25

   This displays a list of information, including a **Data** section that displays **token:** followed by a long series of random characters. This is one of the service account's two secret tokens.

3. Copy the secret token, paste it into a file, and save it.

Result

In a **curl** command, specification of the secret token looks like this:

```
-H 'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJ
uZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJzeW5kZXNpcyIsImt1YmVybmV0ZXMuaW
8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJzeW5kZXNpcy1jZC1jbGllbnQtdG9rZW4tMnZjNmwi
LCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoic3luZGVzaX
MtY2QtY2xpZW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQudW
kIjoiNjUxMjYxNGMtMmYwMS0xMWU5LTk3OWEtNDI1YWNlMzY3MTcyIiwic3ViIjoic3lzdGVtOnNlcnZpY
2VhY2NvdW50OnN5bmRlc2lzOnN5bmRlc2lzLWNkLWNsaWVudCJ9.uKsri0JSKJDbgHoQwAhBJSNuW
KwJgjegf2QlrCkhxVssSK1zIMZQaF9P5a4R7ZcWRnrZ_345UTqxYVeRlfHWVH0PqBkD-
n7PAS9dcKJIFdS1jUHOmL1FTGgc3YW-bz1SlWT93tvK1EhorZ4_-
EBfXhSAP4Uumi5qAg3_QUTMDstq233NSwBKYtFOw3Pp1ys3p3y0hcaiLMimeCH60vR4iWvptqqzc5Q
DigHiPySZNWxs_5awZlwdoIDvR-nSj690aC-
49UKFgyEEdzdFU4bI2W4hOyDyhN9fVaIAZQKeJUrJBU-
lnFTHI_NAd2OwzOEBpWZuj31Za5w9fU4kf4UDGA'
```

Next step

Copy the token from your saved file into a command that calls a public REST API endpoint in the given Fuse Online environment.

10.5.4. How to find integration IDs

In a command that invokes a Fuse Online public REST API endpoint that operates on only a particular integration, you must specify the ID of the integration that you want the endpoint to operate on. Specify one of the following:

- The integration's name
  You must specify it exactly as it appears in the Fuse Online console, for example, **timer-to-log**. If the integration name has any spaces or special characters, then you must specify HTML escape characters.

- The internal integration ID
  This ID is in the Fuse Online console URL when you view an integration's summary. To view an integration's summary, in the left navigation panel, click **Integrations**. In the list of integrations, click the entry for the integration whose ID you need.

  With the integration summary visible in the browser, you would see something like this at the end of the URL: **/integrations/i-Lauq5ShznJ4LcuWwiwcz**. This integration's ID is **i-Lauq5ShznJ4LcuWwiwcz**.

10.5.5. Format for specifying curl commands to invoke Fuse Online public endpoints

A **curl** command that calls a Fuse Online public REST API endpoint has the following format:

```
curl [options] \
    -H "Content-Type: <media-type>" \
    -H "SYNDESIS-XSRF-TOKEN: awesome" \
    -H `Authorization: Bearer <token>` \
    <base-url><endpoint> \
    [--request <HTTP-method>] \
    [-d <data>] \
    [-o <filename>]
```

[options]	Specify **curl** options that you choose.
<media-type>	For the export endpoint, specify **multipart/form-data**. For all other endpoints, specify **application/json**.
<token>	Specify a secret token for the OpenShift service account that you created when you exposed the public REST API.
<base-url>	Specify the base URL for the Fuse Online environment that has the integration, connection, or environment label, that you want the endpoint to operate on.
<endpoint>	Specify the endpoint that you want to call.
[--request <HTTP-method>]	Optionally, specify the HTTP method, for example, **--request POST**.

[-d <data>]	Optionally, depending on the endpoint that is being called, specify arguments that the endpoint requires. For example, to change an integration’s environment label to **test**, you would specify**-d `test`**
[-o <filename>]	Optionally, if you need to specify the name of a file that contains output, specify the **-o curl** option with the name of a file, for example,**-o export.zip**.

The following **curl** command invokes the Fuse Online public API endpoint that marks an integration for one or more environments that you specify:

```
curl -v -k -L -H "Content-Type: application/json" -H "SYNDESIS-XSRF-TOKEN: awesome" -H
'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJ
uZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJzeW5kZXNpcyIsImt1YmVybmV0ZXMuaW
8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJzeW5kZXNpcy1jZC1jbGllbnQtdG9rZW4tMnZjNmwi
LCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoic3luZGVzaX
MtY2QtY2xpZW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQudW
kIjoiNjUxMjYxNGMtMmYwMS0xMWU5LTk3OWEtNDI1YWNlMzY3MTcyIiwic3ViIjoic3lzdGVtOnNlcnZpY
2VhY2NvdW50OnN5bmRlc2lzOnN5bmRlc2lzLWNkLWNsaWVudCJ9.uKsri0JSKJDbgHoQwAhBJSNuW
KwJgjegf2QlrCkhxVssSK1zIMZQaF9P5a4R7ZcWRnrZ_345UTqxYVeRlfHWVH0PqBkD-
n7PAS9dcKJIFdS1jUHOmL1FTGgc3YW-bz1SlWT93tvK1EhorZ4_-
EBfXhSAP4Uumi5qAg3_QUTMDstq233NSwBKYtFOw3Pp1ys3p3y0hcaiLMimeCH60vR4iWvptqqzc5Q
DigHiPySZNWxs_5awZlwdoIDvR-nSj690aC-
49UKFgyEEdzdFU4bI2W4hOyDyhN9fVaIAZQKeJUrJBU-
lnFTHI_NAd2OwzOEBpWZuj31Za5w9fU4kf4UDGA'
https://public-syndesis.192.168.64.42.nip.io/api/v1/public/integrations/timer-to-log/tags -d
'["test","staging"]' --request PUT
```

In the sample **curl** command:

- The URL at the end of the command identifies the Fuse Online environment whose endpoint you are invoking.
- **timer-to-log** indicates that you are marking the **timer-to-log** integration for the specified environments.
- Specification of **test** and **staging** marks the **timer-to-log** integration for those environments.

10.6. USING EXTERNAL TOOLS TO EXPORT/IMPORT INTEGRATIONS FOR CI/CD

When you are running Fuse Online on OCP on-site, you might have Continuous Integration/Continuous Deployment (CI/CD) pipelines that you want to act on certain integrations. Implementing this requires the completion of these tasks:

- In the Fuse Online console, mark integrations for CI/CD pipelines.
- In OpenShift, expose the Fuse Online public API.
- Use external tools to invoke Fuse Online public API endpoints that export and import integrations.

See the following topics for details:

- Section 10.6.1, “About marking integrations for CI/CD”
- Section 10.6.2, “Marking an integration for CI/CD”
- Section 10.6.3, “Invoking the Fuse Online public API export endpoint”
- Section 10.6.4, “Invoking the Fuse Online public API import endpoint”

10.6.1. About marking integrations for CI/CD

When you are running Fuse Online on OCP on-site, to identify an integration for pipelines, mark the integration for a CI/CD environment that you specify. This applies a time-stamped label to the integration. Backend CI/CD Fuse Online public APIs use the label and its timestamp to filter integrations to find an integration that a pipeline needs to act on.

For example, suppose that in Fuse Online you mark an integration for the **test1** environment. You can then invoke the Fuse Online public API export endpoint to export integrations that have the **test1** environment label. The endpoint packages **test1** integrations into an export file and returns that file. To copy **test1** integrations to a Fuse Online test environment, you would invoke the public API import endpoint and provide the file that contains the exported **test1** integrations.

Now suppose that you iteratively update and publish an integration that you previously marked for the **test1** environment. You now have a new version of the integration and you want to export the updated version to the **test1** environment. You must mark the integration again, even though it is already marked for the **test1** environment. Marking the integration again updates the timestamp on the **test1** environment label. This indicates to external tools that the integration has been updated and it is ready to be exported.

To mark an integration again, that is, to refresh the timestamp on an environment label that is already assigned to an integration, start to follow the procedure for Marking integrations for CI/CD. In the CI/CD dialog, you just need to click **Save** since the checkbox for the desired environment should already be selected.

10.6.2. Marking an integration for CI/CD

When you are running Fuse Online on OCP on-site, to identify an integration for pipelines, mark the integration for a CI/CD environment that you specify. This applies a label to the integration.

Prerequisite

You have an OCP project in which Fuse Online is installed.

Procedure

1. In the Fuse Online navigation panel on the left, click **Integrations**.
2. In the list of integrations, at the right of the entry for the integration that you want to mark, click ⋮ to display a popup menu and click **CI/CD**.
3. In the dialog that appears, do one or more of the following:
    - If the environment for which you want to mark this integration already appears, select the checkbox to the left of that environment.

- If the environment for which you want to mark this integration is already selected, leave it selected.
- If you need to create a label for an environment, then in the input field, enter a name for the environment and then click **Add**. For example, enter **test1**.

Fuse Online applies the selected environment label(s) to the integration. You can apply any number of environment labels to an integration.

4. Click **Save**.

Result

Fuse Online tags the integration for release in the selected environment. Backend APIs can filter integrations to find the integrations that have, for example, the **test1** label.

Next step

You must expose the Fuse Online public API endpoints before pipelines can act on integrations that are marked for a particular environment,

10.6.3. Invoking the Fuse Online public API export endpoint

Before you can use external tools to copy Fuse Online integrations from one Fuse Online environment to another, the following tasks must have been done:

- In Fuse Online, an integration that you want to export for a CI/CD pipeline must have been marked for a particular environment. See Marking an integration for CI/CD .
  There is an exception to this requirement. When you want to export all integrations from a Fuse Online environment in one export operation, it does not matter whether or not an integration has already been marked for a particular environment.
- You exposed the Fuse Online public API. See Exposing Fuse Online public API endpoints for external tools.

Endpoint for exporting integrations for a particular environment

To export integrations that have been marked for a particular environment, Fuse Online provides the following **GET** method endpoint:

/public/integrations/{env}/export.zip

Replace **{env}** with a CI/CD environment label that you created in the Fuse Online console or by calling a Fuse Online public API endpoint for marking an integration. When an integration is marked for a particular environment, Fuse Online maintains a timestamp that indicates when it was marked. The endpoint exports an integration only if it has not already been exported since it was marked. For example, to export integrations that have been marked for the **test1** environment, the endpoint is:

/public/integrations/test1/export.zip

This endpoint exports each integration that has the **test1** environment label and that has been marked for the **test1** environment since the last time that it was exported. The endpoint packages the integrations in the **export.zip** file and returns that file.

If no integration that is marked for the specified environment has been marked since the last time that the integration was exported, then the endpoint returns an HTTP **204** response to indicate that there is nothing to return.

Endpoint for exporting all integrations

You can invoke the export endpoint so that it exports all integrations in one Fuse Online environment. This makes it easy to duplicate all integrations in another Fuse Online environment. To do this, add the **all=true** option to the invocation of the export endpoint:

/public/integrations/{env}/export.zip?all=true

Replace **{env}** with a CI/CD environment label. It does not matter whether or not you already created this label in the Fuse Online console. The endpoint assigns the specified environment label to each integration that is not already marked for that environment.

When you add the **all=true** option, you must also explicitly specify that you want the exported integrations to be packaged in the **export.zip** file. For example:

/public/integrations/test1/export.zip?all=true -o export.zip

This invocation of the endpoint:

- Marks each integration for the **test1** environment.
- Returns all integrations in the **export.zip** file.
  If you do not specify the **-o export.zip** option, then the endpoint returns a file whose name is **export.zip?all=true**.

Custom headers required by export endpoint

A command that invokes the export endpoint must specify these custom headers:

- **-H "Content-Type: multipart/form-data"**
- **-H "SYNDESIS-XSRF-TOKEN: awesome"**
  You must specify this custom header exactly as it appears above. The Fuse Online public API requires this header to authenticate the request.
- **-H 'Authorization: Bearer <token>'**
  Replace **<token>** with the secret token that you copied into a file when you created the OpenShift service account that is used to expose the Fuse Online public API.

Sample curl command that exports integrations

Following is an example of a **curl** command that invokes a Fuse Online API endpoint that exports integrations:

```
curl -v -k -L -H "Content-Type: multipart/form-data" -H "SYNDESIS-XSRF-TOKEN: awesome" -H
'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJ
uZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJzeW5kZXNpcyIsImt1YmVybmV0ZXMuaW
8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJzeW5kZXNpcy1jZC1jbGllbnQtdG9rZW4tMnZjNmwi
LCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoic3luZGVzaX
MtY2QtY2xpZW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQudW
kIjoiNjUxMjYxNGMtMmYwMS0xMWU5LTk3OWEtNDI1YWNlMzY3MTcyIiwic3ViIjoic3lzdGVtOnNlcnZpY
2VhY2NvdW50OnN5bmRlc2lzOnN5bmRlc2lzLWNkLWNsaWVudCJ9.uKsri0JSKJDbgHoQwAhBJSNuW
KwJgjegf2QlrCkhxVssSK1zIMZQaF9P5a4R7ZcWRnrZ_345UTqxYVeRlfHWVH0PqBkD-
n7PAS9dcKJIFdS1jUHOmL1FTGgc3YW-bz1SlWT93tvK1EhorZ4_-
EBfXhSAP4Uumi5qAg3_QUTMDstq233NSwBKYtFOw3Pp1ys3p3y0hcaiLMimeCH60vR4iWvptqqzc5Q
DigHiPySZNWxs_5awZlwdoIDvR-nSj690aC-
```

```
49UKFgyEEdzdFU4bI2W4hOyDyhN9fVaIAZQKeJUrJBU-
lnFTHI_NAd2OwzOEBpWZuj31Za5w9fU4kf4UDGA'
https://public-syndesis.192.168.64.42.nip.io/api/v1/public/integrations/dev1/export.zip
```

In the command:

- The URL at the end of the command identifies the Fuse Online environment from which to export integrations.
- Specification of the **dev1** environment label indicates that you want to export integrations that have been marked for the **dev1** environment and that have not already been exported since they were marked.

10.6.4. Invoking the Fuse Online public API import endpoint

You can obtain one or more integrations by invoking a Fuse Online public API export endpoint. To copy exported integrations to another Fuse Online environment, invoke the Fuse Online public API import endpoint.

Endpoint for importing an integration

To import integrations, Fuse Online provides the following **POST** method endpoint:

/public/integrations

In the following example, the endpoint imports the integrations that are in the **export.zip** file and tags them for the **testing** environment:

/public/integrations -F data=@export.zip -F environment=testing

An import endpoint always imports the supplied integrations. That is, even if an integration has not changed since the last time it was imported, the endpoint still imports it.

Custom headers required by import endpoint

A command that invokes the import endpoint must specify these custom headers:

- **-H "Content-Type: application/json"**
- **-H "SYNDESIS-XSRF-TOKEN: awesome"**
  You must specify this custom header exactly as it appears above. The Fuse Online public API requires this header to authenticate the request.
- **-H 'Authorization: Bearer <token>'**
  Replace **<token>** with the secret token that you copied into a file when you created the OpenShift service account that is used to expose the Fuse Online public API.

Sample curl command that imports integrations

Following is an example of a **curl** command that invokes a Fuse Online API endpoint that imports integrations:

```
curl -v -k -L -H "Content-Type: application/json" -H "SYNDESIS-XSRF-TOKEN: awesome" -H
'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJ
uZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJzeW5kZXNpcyIsImt1YmVybmV0ZXMuaW
8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJzeW5kZXNpcy1jZC1jbGllbnQtdG9rZW4tMnZjNmwi
```

```
LCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoic3luZGVzaX
MtY2QtY2xpZW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQudW
kIjoiNjUxMjYxNGMtMmYwMS0xMWU5LTk3OWEtNDI1YWNlMzY3MTcyIiwic3ViIjoic3lzdGVtOnNlcnZpY
2VhY2NvdW50OnN5bmRlc2lzOnN5bmRlc2lzLWNkLWNsaWVudCJ9.uKsri0JSKJDbgHoQwAhBJSNuW
KwJgjegf2QlrCkhxVssSK1zIMZQaF9P5a4R7ZcWRnrZ_345UTqxYVeRlfHWVH0PqBkD-
n7PAS9dcKJIFdS1jUHOmL1FTGgc3YW-bz1SlWT93tvK1EhorZ4_-
EBfXhSAP4Uumi5qAg3_QUTMDstq233NSwBKYtFOw3Pp1ys3p3y0hcaiLMimeCH60vR4iWvptqqzc5Q
DigHiPySZNWxs_5awZlwdoIDvR-nSj690aC-
49UKFgyEEdzdFU4bI2W4hOyDyhN9fVaIAZQKeJUrJBU-
lnFTHI_NAd2OwzOEBpWZuj31Za5w9fU4kf4UDGA'
https://public-syndesis.192.168.64.45.nip.io/api/v1/public/integrations -F data=@export.zip -F
environment=testing
```

In this command:

- The URL at the end of the command identifies the Fuse Online environment to import integrations into.
- The **export.zip** file contains the integrations to be imported.
- Specification of **environment=testing** causes the endpoint to mark each imported integration for the **testing** environment.

10.7. UPGRADING FUSE ONLINE ON OCP

To upgrade Fuse Online on OCP on-site, download the latest Fuse Online release and run the update script.

From time to time, fresh application images, which incorporate patches and security fixes, are released for Fuse Online. You are notified of these updates through Red Hat's errata update channel. You can then upgrade your Fuse Online images.

The upgrade procedure for the following upgrades is the same:

- From Fuse Online 7.2 to Fuse Online 7.3
- From a Fuse Online 7.3 version to a newer Fuse Online 7.3 version

Prerequisites

- You installed and are running version 7.2 of Fuse Online on OCP on-site. **OR**, you installed and are running a version of 7.3 of Fuse Online on OCP on-site and you want to upgrade to fresh application images.
  If you are running version 7.1 of Fuse Online on OCP, then you must upgrade to 7.2 and then you can upgrade to 7.3.
- You installed the **oc** client tool and it is connected to the OCP cluster in which Fuse Online is installed.
- A user with cluster administration permissions gave you permission to install or upgrade Fuse Online in any project that you have permission to access in the cluster.
- If you are upgrading from 7.2 to 7.3, you have a Red Hat developer account for which you know your user name and password. The upgrade script prompts you for these credentials so it can authenticate you against **https://developers.redhat.com**. For details about creating an account, see Accessing and Configuring the Red Hat Registry .

Procedure

1. Download the package containing the Fuse Online installation scripts from the following location:
   https://github.com/syndesisio/fuse-online-install/releases/tag/1.6

   Unpack the downloaded archive at a convenient location on your file system. The **fuse-online-install-1.6** directory contains the scripts and supporting files for upgrading Fuse Online.

2. Change to the directory that contains the extracted archive. For example:
   $ cd fuse-online-install-1.6

3. Log in to OpenShift with an account that has permission to upgrade Fuse Online. For example:
   $ oc login -u developer

4. Invoke the following command, which returns the name of the current project, to ensure that the current project is the project where Fuse Online is installed:
   $ oc project

   If you need to switch to the project where Fuse Online is installed then invoke the following command with the name of the OpenShift project that contains Fuse Online:

   $ oc project *project-name*

5. To check which version you are about to upgrade to, run the update script with the **--version** option, as follows:
   $ bash update_ocp.sh --version

6. Invoke the update script as follows:
   $ bash update_ocp.sh

   To learn more about the script, invoke **$ bash update_ocp.sh --help**.

   During and after an infrastructure upgrade, existing integrations continue to run with the *older* versions of Fuse Online libraries and dependencies.

7. Upgrade Fuse Online integrations that are running as follows:

   a. In Fuse Online, select the integration that you want to upgrade.

   b. Select **Edit**.

   c. Select **Publish** to republish the integration.

   Republishing the integration forces a rebuild that uses the latest Fuse Online dependencies.

10.8. UNINSTALLING FUSE ONLINE FROM AN OCP PROJECT

You can uninstall Fuse Online from an OCP project without deleting the project nor anything else in that project. After uninstalling Fuse Online, integrations that are running continue to run but you can no longer edit or republish them.

Prerequisite

- You have an OCP project in which Fuse Online is installed.

- You exported any integrations that you might want to use in some other OpenShift project in which Fuse Online is installed. If necessary, see Export integrations.

Procedure

Invoke the following command:

```
$ oc delete syndesis app
```

This command deletes the Fuse Online infrastructure.

10.9. DELETING AN OCP PROJECT THAT CONTAINS FUSE ONLINE

Deleting an OpenShift project in which Fuse Online is installed deletes everything in the project. This includes all integrations that have been defined as well as all integrations that are running.

Prerequisites

- You have an OCP project in which Fuse Online is installed.

- You exported any integrations that you might want to use in some other OpenShift project in which Fuse Online is installed. If necessary, see Exporting integrations.

Procedure

Invoke the **oc delete project** command. For example, to delete an OpenShift project whose name is **fuseonline**, enter the following command:

```
$ oc delete project fuseonline
```

10.10. FUSE ONLINE PUBLIC REST API ENDPOINTS REFERENCE

This section provides reference information for each Fuse Online public REST API endpoint. For additional information, see How to invoke Fuse Online public REST API endpoints .

The OpenAPI document that defines the public REST API endpoints is available in your Fuse Online environment at **https://<fuse-online-host>/api/v1/swagger.json**. However, this document defines the **tags** object as having three tags: **public-api**, **extensions**, and **integration-support**. Only the **public-api** tag is accessible when using the OpenShift public OAuth proxy process for Fuse Online. You should ignore the other two tags.

- Section 10.10.1, “Endpoint for obtaining the state of an integration”

- Section 10.10.2, “Endpoint for obtaining a list of an integration’s environment labels”

- Section 10.10.3, “Endpoint for marking an integration and keeping unspecified tags”

- Section 10.10.4, “Endpoint for marking an integration and removing unspecified tags”

- Section 10.10.5, “Endpoint for publishing an integration”

- Section 10.10.6, “Endpoint for stopping an integration”

- Section 10.10.7, “Endpoint for exporting integrations”

- Section 10.10.8, “Endpoint for importing integrations”

10.10.1. Endpoint for obtaining the state of an integration

This endpoint returns the state of the specified integration. The state is **Running**, **Stopped**, **Pending**, or **Error**.

Method and endpoint

GET /public/integrations/{id}/state

Table 10.1. Parameters

Name	Type	Description
Header Parameters		
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Path Parameter		
{id}	string	Required path parameter. Name or internal ID of the integration whose state you want to obtain. See How to find integration IDs.

Request example

In the following example, the endpoint returns the state of the **timer-to-log** integration:

/public/integrations/timer-to-log/state

Produces

application/json

Response example

{"currentState":"Unpublished","stateDetails":{"id":"i-Lc0JLrsUFtBJfR_ylfEz:5","integrationId":"i-Lc0JLrsUFtBJfR_ylfEz","deploymentVersion":5,"detailedState":{"value":"BUILDING","currentStep":2,"totalSteps":4},"namespace":"syndesis","podName":"i-timer-to-log-5-build","linkType":"LOGS"}}`

10.10.2. Endpoint for obtaining a list of an integration's environment labels

This endpoint returns the environment labels (tags) that have been applied to the specified integration.

Method and endpoint

GET /public/integrations/{id}/tags

Table 10.2. Parameters

Name	Type	Description
Header Parameters		
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Path Parameter		
{id}	string	Required path parameter. Name or internal ID of the integration whose environment labels you want to obtain. How to find integration IDs.

Request example

In the following example, the endpoint returns the environment labels for the **timer-to-log** integration:

/public/integrations/timer-to-log/tags

Produces

application/json

Response example

{"test":{"name":"test","releaseTag":"i-Lc5WI16UFtBJfR_ylggz","lastTaggedAt":1554887553159,"lastExportedAt":1554887330152,"lastImportedAt":1554888047271},"staging":{"name":"staging","releaseTag":"i-Lc5WI16UFtBJfR_ylgfz","lastTaggedAt":1554887553159}}

10.10.3. Endpoint for marking an integration and keeping unspecified tags

This endpoint uses the **PATCH** method to mark the specified integration for the specified environment(s). If the integration is already marked for a specified environment, the endpoint updates the timestamp for that environment label. If the integration was previously marked for an environment that is not specified in a new request, the endpoint leaves that tag in place and does not update its timestamp.

This **PATCH** endpoint is a convenience method for CI/CD tools because it adds tags without the need to remove any other existing tags. This is in contrast to the **PUT** endpoint, which marks the integration for specified environments and removes any tags for environments that are not specified in the request.

Method and endpoint

PATCH /public/integrations/{id}/tags

Table 10.3. Parameters

Name	Type	Description
Header Parameters		
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Path Parameter		
{id}	string	Required. Name or internal ID of the integration that you want to mark for the specified environment(s). See How to find integration IDs.
Additional Parameters		
--request PATCH		Specify the **PATCH** method.
-d [env{,...}]	string	Required. Specify one or more, comma-separated, environment labels, that you want to add to the specified integration. If the environment label does not already exist, the endpoint creates it. See Marking integrations for CI/CD.

Request example

In the following example, the endpoint marks the **timer-to-log** integration for the **test2** and **test3** environments:

```
public/integrations/timer-to-log/tags --request PATCH -d '["test2","test3"]'
```

Produces

```
application/json
```

Response example

```
{"test2":{"name":"test2","releaseTag":"i-LcXydouUFtBJfR_ylgrz","lastTaggedAt":1555365010746},"test3":{"name":"test3","releaseTag":"i-LcXydouUFtBJfR_ylgsz","lastTaggedAt":1555365010746},"test":{"name":"test","releaseTag":"i-Lc5WI16UFtBJfR_ylggz","lastTaggedAt":1554887553159,"lastExportedAt":1554887330152,"lastImportedAt":1554888047271},"staging":{"name":"staging","releaseTag":"i-Lc5WI16UFtBJfR_ylgfz","lastTaggedAt":1554887553159}}
```

10.10.4. Endpoint for marking an integration and removing unspecified tags

This endpoint uses the **PUT** method to mark the specified integration for the specified environment(s). If the integration was previously marked for an environment that is not specified in the new request, the endpoint removes that environment label from the integration.

To mark an integration without removing unspecified environment labels, call the **PATCH** method endpoint instead.

Method and endpoint

```
PUT /public/integrations/{id}/tags
```

Table 10.4. Parameters

Name	Type	Description
Header Parameters		
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Path Parameter		
{id}	string	Required. Name or internal ID of the integration that you want to mark. See How to find integration IDs.
Additional Parameters		
--request PUT		Specify the **PUT** method.

Name	Type	Description
-d [env{,...}]	string	Required. Specify one or more, comma-separated, environment labels. The endpoint marks the specified integration for these environments. If the environment label does not already exist, the endpoint creates it. See Marking integrations for CI/CD.

Request example

In the following example, the endpoint marks the **timer-to-log** integration for the **test2** and **test3** environments. If the integration was previously marked for any other environments, the endpoint removes those tags from the integration.

public/integrations/timer-to-log/tags --request PUT -d '["test2","test3"]'

Produces

application/json

Response example

{"test2":{"name":"test2","releaseTag":"i-LcXyw7GUFtBJfR_ylgtz","lastTaggedAt":1555365085713},"test3":{"name":"test3","releaseTag":"i-LcXyw7GUFtBJfR_ylguz","lastTaggedAt":1555365085713}}

10.10.5. Endpoint for publishing an integration

This endpoint publishes the specified integration. If the integration is already running, then the endpoint stops the integration and re-publishes it.

Method and endpoint

POST /public/integrations/{id}/deployments

Table 10.5. Parameters

Name	Type	Description
Header Parameter		
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.

Name	Type	Description
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Path Parameters		
{id}	string	Required. Name or internal ID of the integration that you want to publish. See How to find integration IDs.

Request example

In the following example, the endpoint publishes the **timer-to-log** integration.

/public/integrations/timer-to-log/deployments

Produces

application/json

Response example

In this example, the ellipsis indicates the omission of some of the response.

```
{"id":"i-
Lc0JLrsUFtBJfR_ylfEz:8","version":8,"createdAt":1555365135324,"updatedAt":1555365135324,"u
serId":"system:serviceaccount:syndesis:syndesis-cd-
client","currentState":"Pending","targetState":"Published","integrationId":"i-
Lc0JLrsUFtBJfR_ylfEz", . .2c+PC9zdmc+","description":"Trigger events based on an interval or a
cron expression","isDerived":false},"stepKind":"endpoint"},{"id":"-
Lc0I7wqEVfKCDDHC8Jv","configuredProperties":
{"bodyLoggingEnabled":"true","contextLoggingEnabled":"true"},"metadata":
{"configured":"true"},"stepKind":"log","name":"Log"}]}],"continuousDeliveryState":{"test2":
{"name":"test2","releaseTag":"i-
LcXyw7GUFtBJfR_ylgtz","lastTaggedAt":1555365085713},"test3":
{"name":"test3","releaseTag":"i-LcXyw7GUFtBJfR_ylguz","lastTaggedAt":1555365085713}}}}
```

10.10.6. Endpoint for stopping an integration

This endpoint stops the specified integration.

Method and endpoint

PUT /public/integrations/{id}/deployments/stop

Table 10.6. Parameters

Name	Type	Description
Header Parameters		

Name	Type	Description
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Path Parameter		
{id}	string	Required. Name or internal ID of the integration that you want to stop. See How to find integration IDs.

Request example

In the following example, the endpoint stops the **timer-to-log** integration.

/public/integrations/timer-to-log/deployments stop

Produces

application/json

Response example

No content with a **204** status code

10.10.7. Endpoint for exporting integrations

This endpoint exports integrations that are marked for the specified environment and that have never been exported or that have not been exported since the last time that they were marked for that environment. See also: Invoking the Fuse Online public REST API export endpoint .

Method and endpoint

GET /public/integrations/{env}/export.zip

Table 10.7. Parameters

Name	Type	Description
Header Parameters		
-H "Content-Type: <media-type>"	**multipart/form-data**	Media type that the endpoint requires.

Name	Type	Description
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Path Parameter		
{env}	string	Required. Environment label that you created in the Fuse Online console. The endpoint exports the integrations that are tagged for this environment.
Query Parameter		
all=true	string	Optional. Specify this option to export all integrations that are in the Fuse Online environment. The endpoint exports the current version of each integration. If an integration is not already marked for the specified environment, then the endpoint adds the specified environment label to the integration. The specified environment label does not need to have been created in the Fuse Online console.
Additional Parameter		
-o export.zip	string	Required if you specify the query parameter. Without this option, the exported integrations are in a file named **export.zip?all=true**.

Request example

In this example, the endpoint exports integrations that have been tagged for the **test1** environment:

/public/integrations/test1/export.zip

In the following example, the endpoint ensures that each integration is marked for the **test1** environment and returns all integrations in the **export.zip** file.

/public/integrations/test1/export.zip?all=true -o export.zip

Produces

application/octet-stream

Response

The **export.zip** file that contains the exported integration(s). The endpoint returns an HTTP status of **204** if there are no integrations to export.

10.10.8. Endpoint for importing integrations

This endpoint imports the integrations that are in the provided file. See also: Invoking the Fuse Online public REST API import endpoint.

Method and endpoint

POST /public/integrations

Table 10.8. Parameters

Name	Type	Description
Header Parameters		
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Additional Parameters		
data=@export.zip	string	Required. This is the file that contains the integrations that you want to import. You must have previously invoked the export endpoint to obtain this file.
environment={env}	string	Required. Environment label that you want to add to each imported integration.

Request example

In the following example, the endpoint imports the integrations that are in the **export.zip** file and marks them for the **testing** environment.

/public/integrations -F data=@export.zip -F environment=testing

Produces

application/json

Response example

The response is a list of the imported resources, which includes integrations and connections. In the following example, an ellipsis indicates that part of the response is omitted here.

```
{"lastImportedAt":1554888047271,"results":[{"id":"i-
Lc0JLrsUFtBJfR_ylfEz","version":5,"createdAt":1554800274935,"updatedAt":0,"tags":
["timer"],"name":"timer-to-log","flows":[{"id":"-Lc0I5AZEVfKCDDHC8Jv","steps":[{"id":"-
Lc0I5jnEVfKCDDHC8Jv","configuredProperties":{"period":"900000"},"metadata":...
"description":"Trigger events based on an interval or a cron
expression","isDerived":false},"stepKind":"endpoint"},{"id":"-
Lc0I7wqEVfKCDDHC8Jv","configuredProperties":
{"bodyLoggingEnabled":"true","contextLoggingEnabled":"true"},"metadata":
{"configured":"true"},"stepKind":"log","name":"Log"}]}],"continuousDeliveryState":{"staging":
{"name":"staging","releaseTag":"i-
Lc5WI16UFtBJfR_ylgfz","lastTaggedAt":1554887553159},"test":{"name":"test","releaseTag":"i-
Lc5WI16UFtBJfR_ylggz","lastTaggedAt":1554887553159,"lastExportedAt":1554887330152,"lastIm
portedAt":1554887859824}}}]}
```

10.10.9. Endpoint for removing an environment label from a particular integration

This endpoint removes the specified environment label from the specified integration. The environment label itself continues to exist but it no longer marks the specified integration.

Method and endpoint

DELETE /public/integrations/{id}/tags/{env}

Table 10.9. Parameters

Name	Type	Description
Header Parameters		
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Path Parameters		
{id}	string	Required path parameter. Name or internal ID of the integration that you want to unmark. See How to find integration IDs.
{env}	string	Required path parameter. Environment label that you want to remove from the specified integration.
Additional Parameter		

Name	Type	Description
--request DELETE		Specify the **DELETE** method.

Request example

In the following example, the endpoint removes the **dev1** environment label from the **timer-to-log** integration.

/public/integrations/timer-to-log/tags/dev1 --request DELETE

Response example

No content with a **204** status code

10.10.10. Endpoint for obtaining a list of environment labels

This endpoint returns a list of environment labels that exist in the Fuse Online environment. You create environment labels in the Fuse Online console. See Marking integrations for CI/CD.

Method and endpoint

GET /public/environments

Table 10.10. Parameters

Name	Type	Description
Header Parameters		
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.

Produces

application/json

Sample response

["test2", "test3"]

10.10.11. Endpoint for changing an environment label

This endpoint changes an environment label. Integrations that were marked for the original environment label are now marked for the new environment label.

Method and endpoint

PUT /public/environments/{env}

Table 10.11. Parameters

Name	Type	Description
Header Parameters		
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Path Parameter		
{env}	string	Required. Environment label that you want to change.
Additional Parameters		
-d {env}	string	Required. Specify the new label for the environment.
--request PUT		Specify the **PUT** method.

Request example

In the following example, the endpoint changes the **dev1** environment label to **dev2**:

/public/environments/dev1 -d ‘dev2’ --request PUT

Integrations that were marked for the **dev1** environment no longer have that tag. Those integrations are now marked for the **dev2** environment.

Response example

No content with response status code **204**

10.10.12. Endpoint for removing an environment label from all integrations

This endpoint removes the specified environment label from each integration to which it has been applied.

Method and endpoint

DELETE /public/environments/{env}

Table 10.12. Parameters

Name	Type	Description
Header Parameters		
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Path Parameter		
{env}	string	Required. Environment label that you want to remove from integrations that have it.
Additional Parameter		
--request DELETE		Specify the **DELETE** method.

Request example

In the following example, the endpoint removes the **dev1** tag from any integrations that have it:

/public/environments/dev1 --request DELETE

Response example

No content with response status code **204**

10.10.13. Endpoint for changing a connection's properties

This endpoint changes the properties of the specified connection. This is often useful after you import an integration that has connections that require configuration. For example, you might need to change the credentials that a connection uses.

Method and endpoint

POST /public/connections/{id}/properties

Table 10.13. Parameters

Name	Type	Description
Header Parameters		

Name	Type	Description
-H "Content-Type: <media-type>"	**application/json**	Media type that the endpoint requires.
-H "SYNDESIS-XSRF-TOKEN: awesome"	Custom	Required for authentication.
-H 'Authorization: Bearer <token> '	Custom	Secret token for the OpenShift service account that exposes the Fuse Online public REST API. See Obtaining a secret token.
Path Parameter		
{id}	string	Required. Replace **{id}** with the ID of the connection whose properties you want to change. For the connection ID, specify one of the following: ● The connection's name, for example: **PostgresDB**. If the connection name has any spaces or special characters, then you must specify HTML escape characters. ● The internal connection ID, which is in the Fuse Online console URL when you view a connection's details. To view a connection's details, in the left navigation panel, click **Connections**. Then click the connection whose details you want to view. When the connection details are visible in the browser, you would see something like this at the end of the URL: /**connections/i-Laupl8XznJ4LcuWwiwaz**. This connection's ID is **i-Laupl8XznJ4LcuWwiwaz**.

Request example

The following example changes the properties of the **PostgresDB** connection. The new value of the **user** property is **myuser**, and the new value of the **password** property is **mypassword**:

/public/connections/PostgresDB/properties --request POST -d '{ "user": "myuser", "password": "mypassword" }'

Produces

application/json

Sample response

In this example, there is an ellipsis that indicates the omission of a large part of the response.

```
{"uses":0,"id":"i-LaOziUGpQE45nua4pfCz","name":"TODO app","configuredProperties":
{"password":"»ENC:c2cb731046372a275b76beabc92aefa061f79b43fb791fb599d9e85ec0235a7e","
basePath":"/api","host":"http://todo-syndesis.my-minishift.syndesis.io/","specification":...
"userId":"admin","lastUpdated":1555365796629,"createdDate":1553066813379,"board":{"id":"i-
Lbj4-vqUFtBJfR_ylfCz","metadata":{"connector-id":"i-LaOzcPZpQE45nua4pfBz","connector-
version-latest":"1","connector-version-connection":"1"},"messages":
[{"level":"WARN","code":"SYNDESIS007"}],"createdAt":1554494263030,"updatedAt":15544942637
27,"targetResourceId":"i-
LaOziUGpQE45nua4pfCz","notices":0,"warnings":1,"errors":0},"isDerived":false}
```

10.11. ENABLING APACHE CAMEL K TO SPEED UP FUSE ONLINE DEPLOYMENTS ON OCP

You can install Apache Camel K Technology Preview as an optional add-on to Fuse Online to speed up deployment of integrations on OCP.

IMPORTANT

Camel K is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview.

The following topics introduce Apache Camel K and explain how to install it with Fuse Online:

- Section 10.11.1, “About Apache Camel K”
- Section 10.11.2, “Installing Apache Camel K with Fuse Online on OCP”

10.11.1. About Apache Camel K

Apache Camel K is a lightweight cloud integration platform based on the Apache Camel framework for Enterprise Integration Patterns. Camel K runs natively on Kubernetes, OpenShift, and Knative, and is specifically designed and optimized for serverless and microservice architectures.

Camel K provides automation and performance optimizations when running integrations in the cloud. It uses the Kubernetes Operator SDK to deploy integrations (for example, automatically creating services and routes on OCP).

Camel K is a subproject of the Apache Camel open source community. For more details, see https://github.com/apache/camel-k.

10.11.2. Installing Apache Camel K with Fuse Online on OCP

The Camel K Operator is available in Fuse Online to speed up turnaround times when deploying and redeploying integrations on OCP (for example, from 1-2 minutes to a few seconds). This section explains how to install and configure Camel K with Fuse Online from the Syndesis open source project.

Prerequisites

- You must be connected to the OCP cluster in which you want to install Fuse Online, or to a local cluster created for development.
- You must have cluster administration privileges.
- You must have the Docker client installed and the Docker daemon running to extract the Camel K client.
- You must authenticate in Docker using the following command:

  ```
  $ docker login registry.redhat.io
  ```

Procedure

1. Download and unzip the latest Fuse Online 1.6.x release: https://github.com/syndesisio/fuse-online-install/releases/tag/1.6
2. Set up Fuse Online to use Camel K:

   ```
   $ bash install_ocp.sh --setup --camel-k
   ```

   This installs the Fuse Online and Camel K Operators in your cluster.
3. Install Fuse Online and Camel K in your cluster:

   ```
   $ bash install_ocp.sh --camel-k
   ```

   By default, this installs Fuse Online and the Camel K runtime engine in the current OCP project, and the script calculates the route by which Fuse Online is accessed. Only Camel K is installed if Fuse Online is already installed. For more details, see https://github.com/syndesisio/fuse-online-install.
4. Update the **ConfigMap** object for **syndesis-server-config** to enable Camel K:

   ```
   $ oc get configmap syndesis-server-config -o yaml | sed 's/controllers:/controllers:\n
   integration: camel-k/' | oc apply -f -
   ```

   Alternatively, you can manually edit this **ConfigMap** and add the following setting in the embedded **application.yml** section:

   ```
   controllers:
     integration: camel-k
   ```
5. Restart the **syndesis-server**:

   ```
   $ oc delete pod -l syndesis.io/component=syndesis-server
   ```

   #### TIP

   If existing integrations are running, it is best to stop them all in Fuse Online before restarting the server.

Limitations

The Camel K runtime in the Technology Preview does not support the following Fuse Online features:

- REST API client connectors
- API provider integrations
- Webhooks
- Extensions

NOTE

When **camel-k** is configured as the runtime engine, it replaces the classic Camel integration runtime (**spring-boot**).

Further information

For more details, see Section 10.1, “Installation of Fuse Online on OCP” .