Red Hat Fuse 7.0

Integrating Applications with Ignite

User's guide to integrating applications with Ignite

Last Updated: 2018-08-27

Red Hat Fuse 7.0 Integrating Applications with Ignite

User's guide to integrating applications with Ignite

Legal Notice

Abstract

Ignite provides integration as a service.

Table of Contents

PREFACE

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

- Chapter 1, *Understanding Ignite*
- Chapter 2, *Logging in to Ignite*
- Chapter 3, *Connecting to applications that you want to integrate*
- Chapter 4, *Adding and managing customizations*
- Chapter 5, *Creating integrations*
- Chapter 6, *Mapping data to fields for the next connection*
- Chapter 7, *Managing integrations*
- Chapter 8, *Installing Ignite on OpenShift Container Platform*

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

In this release, consider the names Red Hat Fuse Online and Ignite as interchangeable.

To obtain support, in Ignite, in the upper right, click  and then select **Support**.

CHAPTER 1. UNDERSTANDING IGNITE

Ignite is an integration platform. With Ignite, 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.

For a high-level overview of Ignite, see:

1.1. HOW IGNITE WORKS

Ignite lets you enable data transfer between different applications. For example, each time a customer is mentioned in Twitter, you might want to capture that mention in your Salesforce account for that customer. Another example is a service that makes stock trade recommendations. You can capture recommendations of interest and forward them to a service that automates stock transfers.

Ignite 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.

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.
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.

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

1.2. WHO IGNITE IS FOR

Ignite 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.

1.3. BENEFITS OF IGNITE

With Ignite, you can:

- Integrate data from different applications or services without writing code.
- Run the integration on OpenShift in the public cloud or 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 Ignite 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. UNDERSTANDING IGNITE TERMS

When you integrate applications with Ignite, you create an integration by working with connectors, connections, actions, and steps. See the following sections to learn about each of these constructs.

- Section 1.4.1, “About integrations”
- Section 1.4.2, “About Ignite connectors”
- Section 1.4.3, “About Ignite connections”
- Section 1.4.4, “About actions”
- Section 1.4.5, “About steps”

1.4.1. About integrations

To set up data integration between applications, you create an integration. An integration is set of ordered steps. This set includes:

- A step that connects to an application to start the integration. This connection provides the initial source data that the integration operates on. A subsequent connection can provide additional source 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 source 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.

In an integration, each step can operate on the data that is output from the previous steps. To determine the steps that you need in an integration, see Section 1.5, “Planning integrations”.

1.4.2. About Ignite connectors

Ignite 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.

If Ignite does not provide a connector you need, a developer can create the needed connector.

1.4.3. About Ignite 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 create an AMQ connection, you select the AMQ connector and then follow 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. While you can create any number of connections from one connector, doing so does not always make sense. For example, you can use the AMQ connector to create three AMQ connections where each connection accesses a different broker. However any connector for an application that uses OAuth requires authorization from that application for access to a particular user account. Consequently, each connection that you create from that connector would have the same configuration. There is no need for more than one connection to an application that uses OAuth.

To create an integraton, you select a connection to start the integration, a connection to end the integration, and optionally one or more connections for accessing additional applications. Any number of integrations can use the same connection.

For details, see Chapter 3, *Connecting to applications that you want to integrate*.

1.4.4. About actions

In an integration, each connection performs one action. As you create an integration, you choose a connection to add to the integration and then you choose the action that the connection performs. For example, when you create an integration that uses a Salesforce connection, 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 Ignite prompts you for this information if it is needed.

1.4.5. About steps

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

To operate on data between connections, Ignite provides steps for:

- Mapping data fields in one application to data fields in another application.
- Filtering data so that the integration continues only when it meets criteria that you define.
- Logging information in addition to the default logging that Ignite automatically provides.

See also Section 5.3, “Adding steps between connections” and Section 1.6, “Workflow for creating an integration”.

To operate on data between connections in a way that is not built into Ignite, you can upload an extension that provides a custom step. See Section 4.2, “Adding extensions”.

1.5. PLANNING INTEGRATIONS

Some planning so that you have answers to the following questions is helpful before you create an integration.

To start the integration:

- Which application should the integration 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 Ignite use to access this application?

To finish the 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 Ignite use to access this application?

Between the start and finish applications:

- Do you need to access any other applications? For any other applications the integration accesses:
  - Which application does the integration need to connect to?
  - What action should the integration perform?
  - What are the data fields of interest?
  - What credentials should the integration use to connect to this application?
- Does the integration need to operate on the data between connections? For example:
  - Should the integration 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 integration need to operate on the data in some customized way?

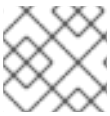
1.6. WORKFLOW FOR CREATING AN INTEGRATION

After you log in to the Ignite console, the general steps for integrating applications are:

1. For each application that you want to integrate and that uses the OAuth protocol, register Ignite as a client of that application.
2. For each application that you want to integrate, create a connection.
3. Create the integration:
    a. Select the start connection. This connection starts the integration by accessing the application you want to obtain data from.
    b. Select the action that you want the start connection to perform.
    c. Optionally, depending on the connection, enter some configuration information, for example, you might indicate whether to operate on a Salesforce contact or a Salesforce lead.
    d. Select the finish connection. This connection completes the integration by accessing the application that uses the data from the start connection.
    e. Select the action you want the finish connection to perform.
    f. Optionally, depending on the connection, enter some configuration details.
    g. Optionally, between the start connection and the finish connection, add one or more connections to other applications.
    h. Optionally, between connections, add additional steps, such as filtering data, mapping data fields, or logging that is in addition to the automatically-provided logging. Typical integrations require data mapping.
4. Click **Publish** to start running your integration.

1.7. HOW YOU USE IGNITE

The best way to learn about how to use Ignite is to create the sample integrations by following the instructions in the sample integration tutorials. Following is an abbreviated description of one of the samples to provide an overview of how you use Ignite. These steps omit details so you should not try to follow them.



NOTE

If you already created a sample integration then you can skip this section.

1. Register your installation of Ignite as an application that can access Salesforce. You need to do this only once to be able to create any number of integrations that connect to Salesforce.
2. Create a Salesforce connection. To configure this connection, Ignite prompts you to log in to the Salesforce account you used to register Ignite. You can use the same Salesforce connection in any number of integrations.

3. Choose your Salesforce connection as the connection that starts the integration.

4. Choose the action that you want the Salesforce connection to perform. In the sample integration, you choose the **On create** action for the **Lead** object. After connecting to Salesforce, the integration watches for notifications that a Salesforce lead was created. When the integration finds such a notification, it passes the new lead's data to the next step in the integration. However, before you can add the next step, you must choose the integration's finish connection.

5. Choose the **PostgresDB** connection as the connection that completes the integration.

6. Choose the action that you want the **PostgresDB** connection to perform. In the sample integration, you choose **add_lead** as the procedure you want to invoke. This is a provided stored procedure that runs in the sample database. This procedure determines the requirements for mapping Salesforce data fields to database fields.

7. Add a step between the Salesforce connection and the database connection. This step maps Salesforce data fields to database fields.

8. Give the integration a name. Optionally, enter a description of what the integration does.

9. Click **Publish** to start running the integration.

10. On the Ignite dashboard, confirm that the Salesforce to database integration is designated as **Published**, which means that it is running.

11. Confirm that the integration is working as expected by creating a new lead in Salesforce.

12. For this sample integration, in a browser, insert `todo-` in front of the URL for your Ignite installation. This displays the notification that a new lead was created in the database.

1.8. ABOUT THE PRODUCT NAME

Ignite is Red Hat's web-based integration platform. Syndesis is the open source project for Ignite.

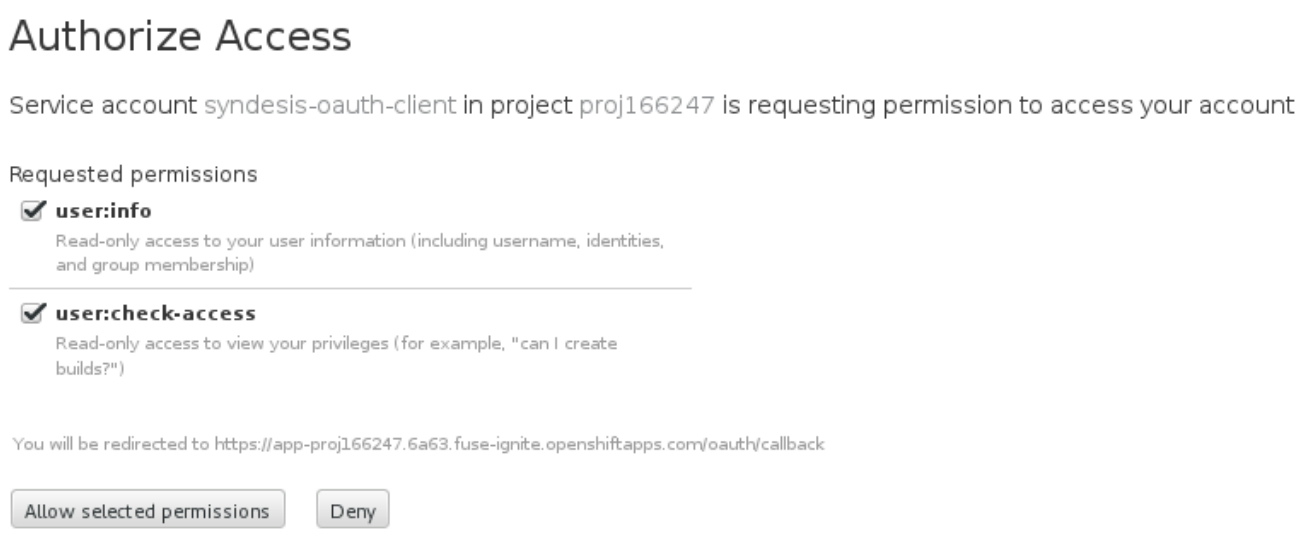
Ignite runs in two environments:

Product Name	Host	Installation
Fuse Online	OpenShift Online	Red Hat installs and provisions Ignite on Red Hat infrastructure.
Ignite	OpenShift Container Platform	Customer installs and manages.

In user documentation, consider the names Fuse Online and Ignite as interchangeable.

CHAPTER 2. LOGGING IN TO IGNITE

In this release, you receive a link for accessing Ignite. 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 Ignite to access to your account:



Click **Allow selected permissions**. You need to do this only once. The next time you log in, Ignite immediately appears.

Red Hat supports using Ignite in the following browsers:

- Chrome
- Firefox
- Microsoft Edge

CHAPTER 3. CONNECTING TO APPLICATIONS THAT YOU WANT TO INTEGRATE

When you create an integration you add a connection to each application that you want to integrate. The following topics provide details for creating connections and adding them to integrations:

3.1. OVERVIEW OF CREATING CONNECTIONS

You must create a connection to each application that you want to integrate. The procedure for creating a connection varies for each application. The following topics provide an overview of the workflow:

3.1.1. About creating connections

To create a connection, you select the connector for the application that you want to connect to and respond to prompts to configure the 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.

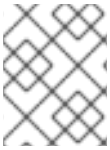
For examples, see:

- Section 3.4.1, “Create an AMQ connection”
- Section 3.8.1, “Create a connection to an HTTP or HTTPS endpoint”
- Section 3.12.1, “Create a Slack connection”

3.1.2. Obtaining authorization to access applications

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 Ignite for access to that application. Registration authorizes all connections from your Ignite installation to a given application. For example, if you register your Ignite installation with Salesforce, all connections from your Ignite installation to Salesforce use the same client ID and the same client secret.

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



NOTE

For information about using custom connectors that let you access applications that use the OAuth protocol, see Section 3.1.5, “Creating connections from custom connectors”.

For details, see the following topics:

- Section 3.1.2.1, “General procedure for obtaining authorization”
- Section 3.1.2.2, “About connection validation”

3.1.2.1. General procedure for obtaining authorization

To integrate applications that use OAuth, you must register with that application before you can create a connection to the application. For example, after you register your installation of Ignite as an application that can access Salesforce, then you can create a Salesforce connection.

While the specific steps vary for each OAuth application that you want to connect to, registration always provides your installation of Ignite 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, Ignite provides an entry in its **Settings** page that makes it easy to register with the application. To see which applications this applies to, in the left panel of Ignite, click **Settings**.

For an application that has an entry in the Ignite **Settings** page, to register Ignite with that application, the main steps are:

1. In the Ignite **OAuth Application Management** page, in the entry for the application with which you want to register Ignite, 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 required to obtain a client ID and secret. One of these steps requires you to enter the callback URL for your installation of Ignite. Paste the URL that you copied to the clipboard in the second step.

4. On your Ignite installation **Settings** page, paste the client ID and client secret and save the settings.

For examples, see

- Section 3.11.1, “Register Ignite as a Salesforce client”
- Section 3.14.1, “Register Ignite as a Twitter client”

For an example of registering with an application that does not have an entry in the Ignite **Settings** page, see: Section 3.6.1, “Register Ignite as a Dropbox client”.

3.1.2.2. About connection validation

After obtaining authorization for Ignite 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, Ignite 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 data. If the access token expires, you can reconnect to the application to obtain a new acess 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 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 Ignite settings for the application. See the instructions in this guide 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, republish the integration. To republish an integration, stop it and restart it.

If validation fails and reconnection fails but everything appears to be valid at the service provider, then try reregistering your Ignite environment with the application and then recreate the connection. Ignite 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.

3.1.3. About adding connections to integrations

When you add a connection to an integration, Ignite 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. 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.

3.1.4. Viewing and editing connection information

After you create a connection, Ignite assigns an internal identifier to the connection. This identifier does not change. You can change the connection's name, description, or configuration values and Ignite 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 details. In the **Integration Summary** page, in the flow diagram of the integration, click a connection icon to view that connection's details.

On the **Connection Details** page, for the connection 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 by stopping it and publishing it again.

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.

3.1.5. Creating connections from custom connectors

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 Ignite-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 installation of Ignite 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 installation of Ignite vary for each application.

For example, suppose the custom connector is for creating connections to Yammer. You would need to register your installation of Ignite by creating a new application within Yammer. Registration provides a Yammer client ID for Ignite and a Yammer client secret value for Ignite A connection from your Ignite installation 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 installation of Ignite, 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.

For more information, see Section 3.1, “Overview of creating connections”.

3.2. SPECIFYING CONNECTION INPUT OR OUTPUT TYPES

To process data from the start connection through the finish connection, sometimes you need to specify input/output types when you configure a connection’s action. Type specifications let Ignite alert you when a data mapping step is required. A data mapping step ensures that the next integration step can process the data it receives.

After you configure an Amazon S3, AMQ, AMQP, Dropbox, or FTP/SFTP connection, Ignite prompts you to specify input and/or output data types as follows:

1. In the **Select Type** field, if the data type does not need to be known, accept **Type specification not required** and then, at the bottom, click **Done**. You do not need to follow the rest of these instructions.
   Otherwise, select one of the following as the schema type:

   - **JSON schema** is a document that describes the structure of JSON data. The document’s media type is **`application/schema+json`**.
   - **JSON instance** is a document that contains JSON data. The document’s media type is **`application/json`**.
   - **XML schema** is a document that describes the structure of XML data. The document’s file extension is **`.xsd`**.
   - **XML instance** is a document that contains XML data. The document’s file extension is **`.xml`**.

2. In the **Definition** input box, paste a definition that conforms to the schema type you selected. For example, if you select **JSON schema** then you would paste the content of a JSON schema file, which has a media type of **`application/schema+json`**.

3. In the **Data Type Name** field, enter a name that you choose for the data type. For example, suppose you are specifying a JSON schema for vendors. You can specify **`Vendor`** as the data type name.
   You will see this data type name when you are creating or editing an integration that uses the connection for which you are specifying this type. Ignite displays the type name in the integration visualization panel and in the data mapper.

4. In the **Data Type Description** field, provide information that helps you distinguish this type. This description appears in the data mapper when you hover over the step that processes this type.

5. Click **Done**.

3.3. CONNECTING TO AMAZON S3

In an integration, to retrieve data from an Amazon S3 bucket or copy data into an Amazon S3 bucket, you create an Amazon S3 connection. You then add that Amazon S3 connection to an integration. For details, see:

3.3.1. Prerequisites for creating an Amazon S3 connection

To create an Amazon S3 connection, you must know the following:

- Amazon S3 access key ID that is associated with the Amazon Web Services (AWS) account that created, or will create, the bucket that you want the connection to access.
  You can create a connection that accesses a bucket that does not yet exist. In this case, when the integration starts running then it use the AWS account associated with this access key ID to try to create the bucket.
- Amazon S3 secret access key that is associated with the AWS account that created or will try to create (when the integration starts running) the bucket that you want the connection to access.
- Name of the bucket that you want to access or its Amazon Resource Name (ARN).
  If the bucket you specify does not yet exist then the connection tries to create a bucket with the name that you specify. Because S3 allows a bucket to be used as a URL that can be accessed publicly, the bucket name that you specify must be globally unique. Also, it must meet S3 bucket naming requirements.

  If the bucket you specify does not exist in the AWS account that is associated with the Amazon S3 access key ID, but it does exist in another AWS account, then the connection does not create the bucket and an integration that uses this connection cannot start running.
- Region in which the bucket is located or the region in which you want the connection to create the bucket.

A user with the login credentials for the AWS account that created or will create the bucket obtains the Amazon S3 keys as follows:

1. Go to https://aws.amazon.com/s3/.
2. Sign in to the console with the AWS account that created the bucket that you want to access or with the account that you want the connection to use to create the bucket.
3. In the console, in the upper right, click the down arrow next to the user name and click **My Security Credentials**.
4. Expand **Access Keys** and click **Create New Access Keys**.
5. Follow the prompts to obtain the keys.

3.3.2. Create an Amazon S3 connection

To create an Amazon S3 connection:

1. In Ignite, in the left panel, click **Connections** to display any available connections.
2. In the upper right, click **Create Connection** to display Ignite connectors.
3. Click the **Amazon S3** connector.

4. In the **Access Key** field, enter the Amazon S3 access key ID, provided by AWS, for the AWS account that created the bucket that you want this connection to access. If the bucket you want the connection to access does not already exist then when Ignite tries to start running the integration, it uses the AWS account associated with this access key to create the bucket. However, if the bucket already exists in some other AWS account, then the connection cannot create the bucket and the integration cannot start.

5. In the **Bucket Name or Amazon Resource Name** field, enter the name of the bucket that you want this connection to access or enter the bucket's ARN. If the bucket does not already exist in either the AWS account being used or in any other AWS account, then the connection creates it. For details about bucket name requirements, see Section 3.3.1, "Prerequisites for creating an Amazon S3 connection".

6. In the **Region** field, select the AWS region in which the bucket resides. If the connection creates the bucket, then it creates it in the selected region.

7. In the **Secret Key** field, enter the Amazon S3 secret access key, provided by AWS, for the account that created, or will create, the bucket that you want this connection to access.

8. Click **Validate**. Ignite immediately tries to validate the connection and displays a message that indicates whether or not validation is successful. If validation fails, revise the configuration details as needed and try again.

9. When validation is successful, click **Next**.

10. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections. For example, enter **`Obtain S3 Data`**.

11. In the **Description** field, optionally enter any information that is helpful to know about this connection. For example, enter **`Sample S3 connection that obtains data from the northeast bucket.`**

12. In the upper right, click **Create** to see that the connection you created is now available. If you entered the example name, you would see that **Obtain S3 Data** is now available.

3.3.3. Adding an Amazon S3 connection to an integration

You must create an Amazon S3 connection before you can add an Amazon S3 connection to an integration. If you did not already create an Amazon S3 connection, see Section 3.3.1, "Prerequisites for creating an Amazon S3 connection".

The procedure for adding an Amazon S3 connection to an integration varies according to whether you want to use the S3 connection to start the integration, finish the integration, or access data in the middle of the integration. See the following topics:

- Section 3.3.3.1, "Starting an integration by obtaining data from Amazon S3"
- Section 3.3.3.2, "Finishing an integration by adding data to Amazon S3"
- Section 3.3.3.3, "Adding data to Amazon S3 in the middle of an integration"

3.3.3.1. Starting an integration by obtaining data from Amazon S3

To start an integration by obtaining data from an Amazon S3 bucket, add an Amazon S3 connection as the start connection:

1. In the Ignite panel on the left, click **Integrations**.
2. Click **Create Integration**.
3. On the **Choose a Start Connection** page, click the Amazon S3 connection that you want to use to start the integration.
4. On the **Choose an Action** page, click one of these actions:
   - **Get Object** to obtain a file from the bucket that the connection accesses. To configure this action:
     a. In the **File Name** field, enter the name of the file that you want to obtain.
     b. To obtain a file and then delete it from the bucket, select **Delete After Read**.
   - **Poll an Amazon S3 Bucket** to periodically obtain files from the bucket that the connection accesses. To configure this action:
     a. In the **Delay** field, accept the default of 500 milliseconds as the time that elapses between polls. Or, to specify a different polling interval, enter a number and select its time unit.
     b. In the **Maximum Objects to Retrieve** field, enter the largest number of files that one poll operation can obtain. The default is 10.
     To have no limit on the number of files that can be obtained, specify `0` or a negative integer. When **Maximum Objects to Retrieve** is unlimited, the poll action obtains all files in the bucket.

     If the bucket contains more than the specified maximum number of files then the action obtains the files that were most recently modified or created.
     c. In the **Prefix** field, optionally specify a regular expression that evaluates to a string. If you specify a prefix then this action retrieves a file only when its name starts with that string.
     d. Indicate whether you want to **Obtain files and then delete them from the bucket**.
5. Click **Done** to specify the action's output type. See Section 3.2, "Specifying connection input or output types".

3.3.3.2. Finishing an integration by adding data to Amazon S3

To finish an integration by copying data to Amazon S3, add an Amazon S3 connection as the finish connection:

1. Start creating the integration.
2. Add and configure the start connection.
3. On the **Choose a Finish Connection** page, click the Amazon S3 connection that you want to use to finish the integration.
4. On the **Choose an Action** page, click the **Copy Object** action to copy one or more objects into the bucket that the connection accesses. The integration obtains the objects to be added to the bucket from the previous integration step(s).

5. Click **Next** to specify the action's input type. See Section 3.2, "Specifying connection input or output types".

3.3.3.3. Adding data to Amazon S3 in the middle of an integration

In the middle of an integration, to add data to Amazon S3, add an Amazon S3 connection between the start and finish connections:

1. Add the start and finish connections.
2. In the left panel, hover over the plus sign that is in the location where you want to add the Amazon S3 connection.
3. In the popup, click **Add a Connection**.
4. Click the Amazon S3 connection that you want to use as a middle connection in the integration.
5. Click the **Copy Object** action. The integration obtains the objects to be added to the bucket from the previous integration step(s).
6. Click **Next** to specify the action's input type. See Section 3.2, "Specifying connection input or output types".

3.4. CONNECTING TO AMQ

In an integration, you can obtain messages from an ApacheMQ (AMQ) broker or publish messages to an AMQ broker. AMQ uses the OpenWire protocol for communication between clients and message brokers. To communicate with the following broker types, use the AMQ connector to create a connection to the broker of interest:

- Apache ActiveMQ broker that does not support AMQP
- AMQ 6 broker

To communicate with one of the following broker types, use the AMQP connector to create a connection to the broker of interest:

- Apache ActiveMQ broker that supports AMQP
- Apache ActiveMQ Artemis
- AMQ 7 broker
- EnMasse, which is an open source messaging platform

See Section 3.5, "Connecting to AMQP".

To use the AMQ connector, see:

- Section 3.4.1, "Create an AMQ connection"
- Section 3.4.2, "Adding an AMQ connection to an integration"

3.4.1. Create an AMQ connection

To create an AMQ connection:

1. In Ignite, in the left panel, click **Connections** to display any available connections.
2. In the upper right, click **Create Connection** to display connectors.
3. Click the **AMQ** connector.
4. Configure the connection by entering:
    a. In the **Broker URL** field, enter the location that you want to send data to or obtain data from, for example, **`tcp://localhost:61616`**.
    b. In the **User Name** field, enter the user name for the account that you want to use to access this broker.
    c. In the **Password** field, enter the password for the account that you want to use to access this broker.
    d. In the **Client ID** field, enter the ID that allows connections to close and reopen without missing messages. The destination type must be a topic.
    e. If this connection will be used in a development environment, you can save some time by disabling **Check Certificates**. Disabling the checking of certificates is a convenience for development environments. For secure production environments, always enable **Check Certificates**.
    f. In the **Broker Certificate** field, paste the broker's PEM certificate text. This is required except when you disable checking the certificates.
    g. In the **Client Certificate** field, paste the client's PEM certificate text. Content in this field is always optional.
5. Click **Validate**. Ignite immediately tries to validate the connection and displays a message that indicates whether validation is successful. If validation fails, revise the configuration details as needed and try again.
6. If validation is successful, click **Next**.
7. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections. For example, you might enter **`AMQ 1`**.
8. In the **Description** field, optionally enter any information that is helpful to know about this connection. For example, enter **`Sample AMQ connection that uses a provided broker.`**
9. In the upper right, click **Create** to see that the connection you created is now available. If you entered the example name, you would see that **AMQ 1** is now available.

3.4.2. Adding an AMQ connection to an integration

You must create an AMQ connection before you can add it to an integration. If you did not already create an AMQ connection, see Section 3.4.1, "Create an AMQ connection".

The procedure for adding an AMQ connection to an integration varies according to whether you want to use the AMQ connection to start the integration, finish the integration, or publish messages in the middle of an integration. See the following topics:

- Section 3.4.2.1, "Starting an integration based on receiving AMQ messages"

- Section 3.4.2.2, “Finishing an integration by publishing AMQ messages”
- Section 3.4.2.3, “Publishing AMQ messages in the middle of an integration”

3.4.2.1. Starting an integration based on receiving AMQ messages

To trigger execution of an integration based on receiving a message from an AMQ broker, add an AMQ connection as the start connection:

1. In the Ignite panel on the left, click **Integrations**.
2. Click **Create Integration**.
3. On the **Choose a Start Connection** page, click the AMQ connection that you want to use to start the integration.
4. On the **Choose an Action** page, click the **Subscribe for messages** action to receive messages from the queue or topic you specify.
5. To configure the action:
   a. In the **Destination Name** field, enter the name of the queue or topic to receive data from.
   b. For the **Destination Type**, accept **Queue** or select **Topic**.
   c. In the **Durable Subscription ID** field, to allow connections to close and reopen without missing messages, enter the durable subscription ID. The destination type must be a topic.
   d. In the **Message Selector** field, if you want to receive only data that satisfies a particular condition, enter a filter expression.
6. Click **Next** to specify the action’s output type. See Section 3.2, “Specifying connection input or output types”.

3.4.2.2. Finishing an integration by publishing AMQ messages

To finish an integration by publishing messages to an AMQ broker, add an AMQ connection as the finish connection:

1. Start creating the integration by adding and configuring the start connection.
2. On the **Choose a Finish Connection** page, click the AMQ connection that you want to use to finish the integration.
3. On the **Choose an Action** page, click the **Publish messages** action to publish messages to the queue or topic you specify.
4. In the **Destination Name** field, enter the name of the queue or topic to send messages to.
5. For the **Destination Type**, accept **Queue** or select **Topic**.
6. Select **Persistent** to guarantee message delivery even if a connection fails.
7. Click **Next** to specify the action’s input/output type. See Section 3.2, “Specifying connection input or output types”.

3.4.2.3. Publishing AMQ messages in the middle of an integration

In the middle of an integration, to publish messages to an AMQ broker, add an AMQ connection between the start and finish connections. You must be creating or editing an integration. The integratation's start and finish connections must have already been added.

To add an AMQ connection as a middle connection:

1. In the integration visualization panel on the left, click the plus sign that is in the location where you want to add the connection.
2. Click **Add a connection**.
3. On the **Choose a Connection** page, click the AMQ connection that you want the integration to use after the start connection and before the finish connection.
4. On the **Choose an Action** page, select one of the following actions:
   - **Publish messages** action to publish messages to the queue or topic you specify. To configure this action:
     a. In the **Destination Name** field, enter the name of the queue or topic to send messages to.
     b. For the **Destination Type**, accept **Queue** or select **Topic**.
     c. Select **Persistent** to guarantee message delivery even if a connection fails.
   - **Request response using messages** to send messages to the JMS destination you specify and receive a response. To configure this action:
     a. In the **Destination Name** field, enter the name of the queue or topic to send messages to.
     b. For the **Destination Type**, accept **Queue** or select **Topic**.
     c. In the **Message Selector** field, if you want to receive only responses that satisfy a particular condition, enter a filter expression.
     d. In the **Named Reply To** field, enter the name of a queue or topic. The destination sends its response to this queue or topic.
     e. Select **Persistent** to guarantee message delivery even if a connection fails.
     f. In the **Response Time Out** field, specify the number of milliseconds that this connection waits for a response message before throwing a runtime exception. The default is 5000 milliseconds (5 seconds).
5. Click **Next** to specify the action's input type and then the action's output type. See Section 3.2, "Specifying connection input or output types".

3.5. CONNECTING TO AMQP

In an integration, you can obtain messages from or publish messages to an Advanced Message Queue Protocol (AMQP) broker. AMQP defines communication between clients and message brokers. To communicate with the following broker types, use the AMQP connector to create a connection to the broker of interest:

- Apache ActiveMQ broker that supports AMQP

- Apache ActiveMQ Artemis
- AMQ 7 broker
- EnMasse, which is an open source messaging platform

To communicate with one of the following broker types, use the AMQ connector to create a connection to the broker of interest:

- Apache ActiveMQ broker that does not support AMQP
- AMQ 6 broker

See Section 3.4, “Connecting to AMQ”.

NOTE

It is possible to use the AMQP connector to create a connection to an Apache ActiveMQ broker that does not support AMQP or to an AMQ 6 broker. Doing this requires transport configuration in the broker. For information about configuring the broker, see Red Hat JBoss A-MQ Managing and Monitoring Brokers, Adding Client Connection Points. For information about the configuration values to specify, see Red Hat JBoss A-MQ Connection Reference, Advanced Message Queueing Protocol (AMQP).

To use the AMQP connector, see:

- Section 3.5.1, “Create an AMQP connection”
- Section 3.5.2, “Adding an AMQP connection to an integration”

3.5.1. Create an AMQP connection

To create an AMQP connection:

1. In Ignite, in the left panel, click **Connections** to display any available connections.
2. In the upper right, click **Create Connection** to display connectors.
3. Click the **AMQP** connector.
4. Configure the connection by entering:
    a. In the **Connection URI** field, enter the location you want to send data to or obtain data from.
    b. In the **User Name** field, enter the user name for the account that you want to use to access this broker.
    c. In the **Password** field, enter the password for the account that you want to use to access this broker.
    d. In the **Client ID** field, enter the ID that allows connections to close and reopen without missing messages. The destination type must be a topic.
    e. If this connection will be used in a development environment, you can save some time by disabling **Check Certificates**. Disabling the checking of certificates is a convenience for development environments. For secure production environments, always enable **Check**

Certificates.

f. In the **Broker Certificate** field, paste the broker's PEM certificate text. This is required except when disable checking the certificates.

g. In the **Client Certificate** field, paste the client's PEM certificate text. Content in this field is always optional.

5. Click **Validate**. Ignite immediately tries to validate the connection and displays a message that indicates whether validation is successful. If validation fails, revise the configuration details as needed and try again.

6. If validation is successful, click **Next**.

7. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections. For example, you might enter **`AMQP 1`**.

8. In the **Description** field, optionally enter any information that is helpful to know about this connection. For example, enter **`Sample AMQP connection`**

9. In the upper right, click **Create** to see that the connection you created is now available. If you entered the example name, you would see that **AMQP 1** is now available.

3.5.2. Adding an AMQP connection to an integration

You must create an AMQP connection before you can add an AMQP connection to an integration. If you did not already create an AMQP connection, see Section 3.5.1, "Create an AMQP connection".

The procedure for adding an AMQP connection to an integration varies according to whether you want to use the connection to start an integration, finish an integration, or publish messages in the middle of an integration. See the following topics:

- Section 3.5.2.1, "Starting an integration based on receiving AMQP messages"
- Section 3.5.2.2, "Finishing an integration by publishing AMQP messages"
- Section 3.5.2.3, "Publishing messages to AMQP in the middle of an integration"

3.5.2.1. Starting an integration based on receiving AMQP messages

To trigger execution of an integration based on receiving messages from an AMQP broker, add an AMQP connection as the start connection:

1. In the Ignite panel on the left, click **Integrations**.

2. Click **Create Integration**.

3. On the **Choose a Start Connection** page, click the AMQP connection that you want to use to start the integration.

4. On the **Choose an Action** page, click the **Subscribe for messages** action to receive messages from the queue or topic you specify.

5. To configure the action:

a. In the **Destination Name** field, enter the name of the queue or topic to receive data from.

b. For the **Destination Type**, accept **Queue** or select **Topic**.

c. In the **Durable Subscription ID** field, to allow connections to close and reopen without missing messages, enter the durable subscription ID. The destination type must be a topic.

d. In the **Message Selector** field, if you want to receive only data that satisfies a particular condition, enter a filter expression.

6. Click **Next** to specify the action's output type. See Section 3.2, "Specifying connection input or output types".

3.5.2.2. Finishing an integration by publishing AMQP messages

To finish an integration by publishing messages to an AMQP broker, add an AMQP connection as the finish connection. You must be creating or editing an integration. You must have already added the start connection. Follow these instructions:

1. On the **Choose a Finish Connection** page, click the AMQP connection that you want to use to finish the integration.
2. On the **Choose an Action** page, click **Publish messages** to publish messages to the queue or topic you specify.
3. In the **Destination Name** field, enter the name of the queue or topic to send messages to.
4. For the **Destination Type**, accept **Queue** or select **Topic**.
5. Select **Persistent** to guarantee message delivery even if a connection fails.
6. Click **Next** to specify the action's input and output type. See Section 3.2, "Specifying connection input or output types".

3.5.2.3. Publishing messages to AMQP in the middle of an integration

In the middle of an integration, to publish messages to an AMQP broker, add an AMQP connection between the start and finish connections. You must be creating or editing an integration. You must have already added the start and finish connections to the integration. Follow these instructions:

1. In the integration visualization panel on the left, click the plus sign in the location where you want to add the connection.
2. Click **Add a connection**.
3. On the **Choose a Connection** page, click the AMQP connection that you want the integration to use after the start connection and before the finish connection.
4. On the **Choose an Action** page, select one of the following actions:
   - **Publish messages** to publish messages to the queue or topic you specify. To configure this action:
     a. In the **Destination Name** field, enter the name of the queue or topic to send messages to.
     b. For the **Destination Type**, accept **Queue** or select **Topic**.
     c. Select **Persistent** to guarantee message delivery even if a connection fails.

- **Request response using messages** to send messages to the JMS destination you specify and receive a response. To configure this action:

  a. In the **Destination Name** field, enter the name of the queue or topic to send messages to.

  b. For the **Destination Type**, accept **Queue** or select **Topic**.

  c. In the **Message Selector** field, if you want to receive only responses that satisfy a particular condition, enter a filter expression.

  d. In the **Named Reply To** field, enter the name of a queue or topic. The destination sends its response to this queue or topic.

  e. Select **Persistent** to guarantee message delivery even if a connection fails.

  f. In the **Response Time Out** field, specify the number of milliseconds that this connection waits for a response message before throwing a runtime exception. The default is 5000 milliseconds (5 seconds).

5. Click **Next** to specify the action's input and output type. See Section 3.2, "Specifying connection input or output types".

3.6. CONNECTING TO DROPBOX

In an integration, you can download files from Dropbox or upload files to Dropbox. The following topics provide the details:

- Section 3.6.1, "Register Ignite as a Dropbox client"
- Section 3.6.2, "Create a Dropbox connection"
- Section 3.6.3, "Adding a Dropbox connection to an integration"

3.6.1. Register Ignite as a Dropbox client

You must register your installation of Ignite as an application that can access Dropbox. This lets you create any number of integrations that connect to Dropbox. In other words, you need to register a particular installation of Ignite with Dropbox only once.

In each Ignite environment, there can be only one registration of Ignite as a Dropbox client. However, while each Dropbox connection uses the same registration, it can use different user credentials.

Perform these steps:

1. In Ignite:

   a. In the left panel, click **Settings**.

   b. Near the top of the page, in the sentence that starts with **`During registration, enter this callback URL:`**, copy the URL at the end of that sentence to the clipboard. For example, the URL is something like this: **`https://app-proj9128.7b63.fuse-ignite.openshiftapps.com/api/v1/credentials/callback`**.

2. In another browser tab, go to **`https://www.dropbox.com`** and do the following:

   a. Sign in to the Dropbox account that has the data that you want to access in an integration.

b. After signing in, go to https://www.dropbox.com/developers/apps.

c. Click **Create App**.

d. Select **Dropbox API**.

e. Choose whether Ignite can access a single folder or all of the folders and files.

f. Specify a name for your Dropbox app. For example, you might specify **`Ignite Access From Aslan LLC`**. The name you specify must be unique in the set of Dropbox app names.

g. Check the box to indicate that you agree to Dropbox API terms and conditions.

h. Click **Create App**.

i. In the Dropbox **Settings** page for your new app, in the input field for **OAuth2 Redirect URIs**, paste your Ignite URL, which you copied to the clipboard at the beginning of this procedure.

j. Click **Add**.

Your installation of Ignite is now registered as a Dropbox client, which means that Ignite can access content in the Dropbox account that you signed into.

3.6.2. Create a Dropbox connection

A connection to Dropbox requires registration of Ignite as an application that can access Dropbox. If you did not already register Ignite, see Section 3.6.1, “Register Ignite as a Dropbox client”.

Follow the instructions below to create a Dropbox connection. You can use the same Dropbox connection in multiple integrations.

To create a Dropbox connection:

1. In a new browser tab, go to **`https://www.dropbox.com`** and do the following:

   a. Sign in to the Dropbox account in which you created the app that registers access from your Ignite installation.

   b. Go to https://www.dropbox.com/developers/apps.

   c. Click the Ignite app to display its settings.

2. In another browser tab, in Ignite, do the following:

   a. In the left panel, click **Connections** to display any available connections.

   b. In the upper right, click **Create Connection** to display the available connectors.

   c. Click the **Dropbox** connector.

3. Go back to the Dropbox settings display for your app and do the following:

   a. Scroll down to see **Generated Access Token**.

   b. Click **Generate**.

   c. Copy the generated access token to the clipboard.

4. Back in Ignite, in the **Configure Connection** page, in the **Access Token** field, paste the generated access token.
5. In the **Client Identifier** field, enter the name that you specified when you created the Dropbox app.
6. Click **Validate**. Ignite displays a message that indicates whether it can validate this connection. If validation fails, try again and be sure to enter the correct values.
7. When validation is successful, in the upper right, click **Next**.
8. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections. For example, enter **`Dropbox Connect 1`**.
9. In the **Description** field, optionally enter any information that is helpful to know about this connection. For example, enter **`Sample Dropbox connection that can access all content in our company Dropbox account.`**
10. In the upper right, click **Create** to see that the connection you created is now available. If you entered the example name, you would see that **Dropbox Connect 1** is now available.

3.6.3. Adding a Dropbox connection to an integration

You must create a Dropbox connection before you can add a Dropbox connection to an integration. If you did not already create a Dropbox connection, see Section 3.6.2, “Create a Dropbox connection”.

You must be creating an integration or updating an integration to add a connection to that integration. If you need to, see Section 5.2, “Procedure for creating an integration” or Section 7.10, “Updating integrations”.

The procedure for adding a Dropbox connection to an integration varies according to whether you want to use the connection to start an integration, finish an integration, or access Dropbox in the middle of an integration. See the following topics:

- Section 3.6.3.1, “Starting an integration by obtaining files from Dropbox”
- Section 3.6.3.2, “Finishing an integration by adding files to Dropbox”
- Section 3.6.3.3, “Accessing Dropbox in the middle of an integration”

3.6.3.1. Starting an integration by obtaining files from Dropbox

To start an integration by downloading files from Dropbox, add a Dropbox connection as the start connection:

1. In the Ignite panel on the left, click **Integrations**.
2. Click **Create Integration**.
3. On the **Choose a Start Connection** page, click the Dropbox connection that you want to use to start the integration.
4. On the **Choose an Action** page, click the **Download from Dropbox** action to obtain one or more files from the Dropbox account that this connection accesses.

5. To configure the action, in the **Folder or file name path to download** field, specify the filename path for the content that you want the integration to obtain. In this release, you can download only a single file.

6. Click **Next** to specify the action's output type. See Section 3.2, "Specifying connection input or output types".

3.6.3.2. Finishing an integration by adding files to Dropbox

To finish an integration by uploading files to Dropbox, add a Dropbox connection as the finish connection:

1. Start creating the integration, add and configure the start connection.

2. On the **Choose a Finish Connection** page, click the Dropbox connection that you want to use to finish the integration.

3. On the **Choose an Action** page, click the **Upload a file to Dropbox** action to add the current integration data to the Dropbox account that this connection accesses.

4. In the **Remote Path** field, enter the local filename path for file that you want to upload. Dropbox stores the file with the same path and name. In this release, you can upload only a single file.

5. For the **Upload mode**,

   - Select **Add** to upload a file only when a file with the same name is not already in the same Dropbox folder. If a file with the same name is already in the same Dropbox folder, then the file is not uploaded and the integration continues. This is the behavior regardless of whether the content in the file you are trying to upload has been updated.

   - Select **Force** to ensure that the file is uploaded even if a file with the same name is present in the same Dropbox folder. Dropbox overwrites the file that it already has with the file that you are uploading.

6. Click **Next** to specify the action's input type. See Section 3.2, "Specifying connection input or output types".

3.6.3.3. Accessing Dropbox in the middle of an integration

To download or upload Dropbox files in the middle of an integration, add a Dropbox connection between the start and finish connections. You must be creating or editing an integration. You must add the start and finish connections first. Follow these instructions:

1. In the integration visualization panel on the left, hover over the plus sign that is in the location where you want to add a Dropbox connection.

2. In the popup, click **Add a connection**.

3. On the **Choose a Connection** page, click the Dropbox connection that you want the integration to use.

4. On the **Choose an Action** page, select one of the following actions:

   - **Upload a file to Dropbox** to add the current integration data to the Dropbox account that this connection accesses. To configure this action:

     a. In the **Remote Path** field, specify the local path and file name of the file you want to upload. Dropbox stores the file with the same path and name. In this release, you can

upload. Dropbox stores the file with the same path and name. In this release, you can upload only a single file.

b. For the **Upload mode**,

- Select **Add** to upload a file only when a file with the same name is not already in the same Dropbox folder. If a file with the same name is already in the same Dropbox folder, then the file is not uploaded and the integration continues. This is the behavior regardless of whether the content in the file you are trying to upload has been updated.
- Select **Force** to ensure that the file is uploaded even if a file with the same name is present in the same Dropbox folder. Dropbox overwrites the file that it already has with the file that you are uploading.

- **Download from Dropbox** to obtain one file from the Dropbox account that this connection accesses. To configure this action, in the **Folder or filename path to download** field, enter the Dropbox filename path of the file you want to obtain. In this release, you can download only a single file.

5. Click **Next** to specify the action's input and output type. See Section 3.2, "Specifying connection input or output types".

3.7. CONNECTING TO AN FTP OR SFTP SERVER

In an integration, you can connect to an FTP or SFTP server to download or upload files by creating an FTP or SFTP connection. You can then add this connection to any number of integrations. The following topics provide details:

- Section 3.7.1, "Creating an FTP or SFTP connection"
- Section 3.7.2, "Adding an FTP or SFTP connection to an integration"

3.7.1. Creating an FTP or SFTP connection

To create a connection to an FTP server or an SFTP server:

1. In Ignite, in the left panel, click **Connections** to display any available connections.
2. In the upper right, click **Create Connection** to display Ignite connectors.
3. To create a connection that uses File Transfer Protocol, click the **FTP** connector.
   Or, to create a connection that uses Secure File Transfer Protocol, click the **SFTP** connector.
4. Configure the connection.
   - For an FTP connection:
     - **Host** is the only parameter that you must specify. Enter the host name of the server that you want to connect to. For example, if the name of your FTP host is `FTP.WEST`, then you would enter exactly that, `FTP.WEST`. Do not specify the protocol, for example, you should not specify something like this: `ftp://FTP.WEST`.
     - **Port** is required and has a default value of `21`. This is the port that the FTP server is listening on.

- All other parameters are either not required or have default values. The defaults are suitable for most integrations. Descriptions of these parameters are after this procedure.

- For an SFTP connection, there must be values for these parameters:
  - **Host** is the host name of the SFTP server that you want to connect to. For example, if the name of your SFTP host is **`SFTP.EAST`**, then you would enter exactly that, **`SFTP.EAST`**. Do not specify the protocol, for example, you should not specify something like this: **`sftp://SFTP.EAST`**.
  - **Port** has a default of **`22`**. This is the port that the SFTP server is listening on.
  - **User name** of the account that you want to use to access the SFTP server.
  - **Password** that is associated with that user name.
  - All other parameters have default values. The defaults are suitable for most integrations. Descriptions of these parameters are after this procedure.

5. Click **Validate**. Ignite immediately tries to validate the connection and displays a message that indicates whether or not validation is successful. If validation fails, revise the configuration details as needed and try again.
6. When validation is successful, click **Next**.
7. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections. For example, enter **`XLight FTP Server`**.
8. In the **Description** field, optionally enter any information that is helpful to know about this connection.
9. In the upper right, click **Create** to see that the connection you created is now available. If you entered the example name, you would see that **XLight FTP Server** is now available.

Descriptions of other parameters

- **Connect timeout** defaults to **10000** milliseconds and indicates a maximum wait of 10 seconds to establish the connection. If 10 seconds elapse without a connection then Ignite waits for the number of milliseconds defined by **Reconnect delay** and then tries to reconnect.
- **Reconnect delay** defaults to **1000** milliseconds and indicates the wait time before trying to reconnect again.
- **Maximum reconnect attempts** defaults to **3**. Ignite tries as many as 3 times to establish a connection.
- **Binary file transfer mode** is used by default. Select **No** for ASCII transfer mode.
- **Passive connection mode** defaults to **Yes**, which is usually the preferred mode. In passive mode, the client opens communication channels with the server as a way to avoid firewall issues. If you select **No** then active mode is used.
- **Disconnect from the server after use** defaults to **No**. The connection remains established after it performs the action. Select **Yes** if you want to disconnect from the server after the connection performs the upload or download.

- **Data timeout** defaults to **30000** milliseconds and indicates the maximum length of time that Ignite waits for a reply.

3.7.2. Adding an FTP or SFTP connection to an integration

After you create an FTP connection or an SFTP connection, you can add it to any number of integrations. See the following topics:

- Section 3.7.2.1, “Obtaining files from an FTP or SFTP server”
- Section 3.7.2.2, “Uploading files to an FTP or SFTP server”

3.7.2.1. Obtaining files from an FTP or SFTP server

To trigger integration execution when an FTP or SFTP connection finds the file(s) you are interested in, you must add an FTP or SFTP connection as an integration’s start connection.

To add an FTP or SFTP connection as a start connection:

1. In the Ignite panel on the left, click **Integrations**.
2. Click **Create Integration**.
3. On the **Choose a Start Connection** page, click the FTP or SFTP connection that you want to use to poll an FTP or SFTP server.
4. On the **Choose an Action** page, click **Download**.
5. In the **File name expression** field, if you are interested in a single file, then enter an Apache Camel Simple language expression that resolves to a file name. You cannot specify a regular expression. The connection polls (periodically checks) the server for this file and downloads it when it is found. Leave this field blank if you want to download more than one file.
6. In the **FTP directory** field, enter the absolute or relative path of the server directory to poll. The connection watches this directory for any content and downloads all files when it finds any content.
7. In the **Milliseconds before polling starts** field, accept the default of **1000** milliseconds or change the number of milliseconds.
8. In the **Milliseconds before the next poll** field, accept the default of **500** milliseconds or change the number of milliseconds. This is the interval between polls.
9. In the **Delete after download** field, accept the default of **No** or select **Yes** to download the file(s) and then delete it(them) from the server.
10. Click **Next** to specify the action’s output type. See Section 3.2, “Specifying connection input or output types”.

3.7.2.2. Uploading files to an FTP or SFTP server

To finish an integration by uploading files to an FTP or SFTP server, you add an FTP or SFTP connection as the finish connection. You can also upload files to an FTP or SFTP server in the middle of an integration. To do this, you add an FTP or SFTP connection as a middle connection.

To add an FTP or SFTP connection that uploads files:

1. Start creating the integration.
2. Add and configure the start connection.
3. On the **Choose a Finish Connection** page, do one of the following:
   - To finish an integration by uploading files, click the FTP or SFTP connection that you want to use.
   - To upload files in the middle of an integration, click the connection you want to use to finish the integration. Configure that connection. When the finish connection is part of the integration, in the left panel, hover over the plus sign where you want to add an FTP or SFTP connection and click **Add a connection**. Click the FTP or SFTP connection that you want to use to upload files in the middle of an integration.
4. On the **Choose an Action** page, click **Upload**.
5. In the **File name expression** field, if you want to upload only one particular file, then enter an Apache Camel Simple language expression that resolves to a file name. This is the name of the file that the action uploads to the server. You cannot specify a regular expression. To upload more than one file, leave this field blank.
6. In the **FTP directory** field, enter the absolute or relative name of a server directory. If the **File name expression** field contains an expression, then the connection stores the specified file in this directory. If the **File name expression** field is blank, then the connection uploads to this directory all files that were received from the previous integration step.
7. In the **If file exists** field, indicate the behavior when you are uploading a file that has the same path and name as a file that is on the server. Accept the default, **Override**, to overwrite the file that is on the server with the file that you are uploading. Or, select one of the following:
   - **Append** adds the content in the file being uploaded to the file that is on the server.
   - **Fail** throws `GenericFileOperationException`. The integration does not enter an error state.
   - **Ignore** does not upload the file. The integration continues running under the assumption that everything is okay.
   - **Move** renames one of the files.
   - **TryRename** uploads the file with a temporary name and renames the file to the desired name. This operation does not check for the existence of a file with the desired name, which makes the operation faster on most servers than when existence checks are done.
8. In the **Temporary file prefix while copying** field, specify a string. The connection prepends this string to the name of a file while it is being uploaded. This enables the connection to write to a temporary file on the server and then rename that temporary file to have the correct name. This is useful for reducing locks when uploading very large files.
9. In the **Temporary file name while copying** field, specify a string. The connection renames a file being uploaded to have this name while it is being uploaded. This enables the connection to write to a temporary file on the server and then rename that temporary file to have the correct name. This is useful for reducing locks when uploading very large files.
10. Click **Next** to specify the action's input type. See Section 3.2, "Specifying connection input or output types".

3.8. CONNECTING TO HTTP AND HTTPS ENDPOINTS

In an integration, you can connect to HTTP and HTTPS endpoints to execute the **`GET`**, **`PUT`**, **`POST`**, **`DELETE`**, **`HEAD`**, **`OPTIONS`**, **`TRACE`**, or **`PATCH`** method. To do this, create an HTTP or HTTPS connection and then add it to an integration. The following topics provide details:

- Section 3.8.1, “Create a connection to an HTTP or HTTPS endpoint”
- Section 3.8.2, “Add an HTTP or HTTPS connection to an integration”

3.8.1. Create a connection to an HTTP or HTTPS endpoint

To create a connection to an HTTP or HTTPS endpoint:

1. In Ignite, in the left panel, click **Connections** to display any available connections.
2. In the upper right, click **Create Connection** to display Ignite connectors.
3. If you want to use Hyper Text Transfer Protocol to connect to the endpoint, then click the **HTTP** connector. If you want to use Secure Hyper Text Transfer protocol, then click the **HTTPS** connector.
4. In the **Base URL** field, enter the endpoint path. For example, **`www.mycompany.com/sales`**.
5. Click **Validate**. Ignite immediately tries to validate the connection and displays a message that indicates whether validation is successful. If validation fails, revise specification of the base URL and try again.
6. If validation is successful, click **Next**.
7. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections. For example, enter **`HTTPS My Company Sales`**.
8. In the **Description** field, optionally enter any information that is helpful to know about this connection.
9. In the upper right, click **Create** to see that the connection you created is now available. If you entered the example name, you would see that **HTTPS My Company Sales** is now available.

3.8.2. Add an HTTP or HTTPS connection to an integration

After you create an HTTP or HTTPS connection, you can use it in any number of integrations as follows:

- To start an integration by periodically invoking an HTTP or HTTPS endpoint, add an HTTP or HTTPS connection as the integration’s start connection.
- To finish an integration by invoking an HTTP or HTTPS endpoint once, add an HTTP or HTTPS connection as the integration’s finish connection.
- In the middle of an integration, to invoke an HTTP or HTTPS endpoint once, add an HTTP or HTTPS connection after the start connection and before the finish connection.

If you are creating an integration, Ignite prompts you to choose and configure the start connection, and then choose and configure the finish connection. To add a middle connection, hover over the plus sign in the left panel at the location where you want to add the connection, and select **Add a connection**.

In all of these situations, Ignite displays the available connections. To add an HTTP or HTTPS connection:

1. Click the HTTP or HTTPS connection that you want to add to the integration.
2. Select the action that you want the connection to perform:
   - If you are adding a start connection, then **Periodic invoke URL** is the only available action. This action invokes the endpoint at intervals that you specify and triggers the integration if the endpoint returns any data.
   - If you are adding a finish or middle connection, then **Invoke URL** is the only available action. This action invokes the endpoint once.
3. In the **URL Path** field, specify the location of the endpoint that you want to invoke.
4. In the **HTTP Method** field, select the method that you want the connection to perform. The default method is **`GET`**.
   - **`GET`** obtains the content at the URL path.
   - **`PUT`** replaces the content at the URL path with the integration data.
   - **`POST`** stores the integration data at the URL path to create new content.
   - **`DELETE`** removes content at the URL path.
   - **`HEAD`** obtains metadata about the content at the URL path.
   - **`OPTIONS`** obtains communication option settings at the URL path.
   - **`TRACE`** obtains information for testing and diagnostic purposes.
   - **`PATCH`** partially updates the content at the URL path according to the integration data.
5. If you are adding a start connection, which periodically invokes the URL, then in the **Period** field, accept the default interval of **1** second or specify a number and its unit (milliseconds, seconds, minutes, or hours) to indicate how long to wait between invocations.
6. Click **Done** to specify the action's input or output type. See Section 3.2, "Specifying connection input or output types".

3.9. CONNECTING TO MQTT

MQ Telemetry Transport (MQTT) is a lightweight, machine-to-machine, internet of things, connectivity protocol. In an integration, you can obtain messages from or publish messages to an MQTT broker. To do this, create a connection to the MQTT broker of interest and then add that connection to an integration. Details are in the following topics:

- Section 3.9.1, "Create a connection to an MQTT broker"
- Section 3.9.2, "Adding an MQTT connection to an integration"

3.9.1. Create a connection to an MQTT broker

To create an MQTT connection:

1. In Ignite, in the left panel, click **Connections** to display any available connections.
2. In the upper right, click **Create Connection** to display connectors.
3. Click the **MQTT** connector.
4. To configure the connection:
   a. In the **MQTT broker URL** field, enter the location of the MQTT broker that you want to send data to or obtain data from. This is the only required field.
   b. In the **User Name** field, optionally enter the user name for the MQTT account whose credentials you want to use to access the broker.
   c. In the **Password** field, if you specified a user name, then specify the password associated with that account.
   d. In the **Client ID** field, optionally enter the ID that allows connections to close and reopen without missing messages. The connection must subscribe to or publish to a topic.
5. Click **Validate**. Ignite immediately tries to validate the connection and displays a message that indicates whether validation is successful. If validation fails, revise the input parameters and try again.
6. If validation is successful, click **Next**.
7. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections. For example, you might enter **`MQTT West`**.
8. In the **Description** field, optionally enter any information that is helpful to know about this connection.
9. In the upper right, click **Create** to see that the connection you created is now available. If you entered the example name, you would see that **MQTT West** now available.

3.9.2. Adding an MQTT connection to an integration

You must create an MQTT connection before you can add it to an integration. If you did not already create an MQTT connection, see Section 3.9.1, “Create a connection to an MQTT broker”.

The procedure for adding an MQTT connection to an integration varies according to whether you want to use the MQTT connection to obtain a message or publish a message. See the following topics:

- Section 3.9.2.1, “Obtain a message from an MQTT broker”
- Section 3.9.2.2, “Publishing a message to an MQTT broker”

3.9.2.1. Obtain a message from an MQTT broker

To trigger execution of an integration based on receiving a message from an MQTT broker, add an MQTT connection as the start connection. When you publish the integration, the MQTT connection continuously watches for messages on the MQTT queue or topic that you specify. When the connection finds a message, it passes it to the next step in the integration. An MQTT connection handles one message at a time.

To start an integration when a message from an MQTT broker is found:

1. In the Ignite panel on the left, click **Integrations**.
2. Click **Create Integration**.
3. On the **Choose a Start Connection** page, click the MQTT connection that you want to use to start the integration.
4. On the **Choose an Action** page, click the **Subscribe** action to receive messages from the queue or topic you specify.
5. In the **MQTT queue/topic name** field, enter the name of the queue or topic to subscribe to in order to receive data.
6. Click **Done** to add the start connection.

3.9.2.2. Publishing a message to an MQTT broker

In an integration, you can publish a message to an MQTT broker to finish an integration. To do this, add an MQTT connection as the integration's finish connection. To publish a message to an MQTT broker in the middle of integration, add an MQTT connection to an integration after the start connection and before the finish connection.

To add an MQTT connection that publishes a message:

1. Start creating the integration.
2. Add and configure the start connection.
3. On the **Choose a Finish Connection** page, do one of the following:
   - To finish an integration by publishing a message, click the MQTT connection that you want to use.
   - To publish a message in the middle of an integration, click the connection you want to use to finish the integration. Configure that connection. When the finish connection is part of the integration, in the left panel, hover over the plus sign where you want to add an MQTT connection and click **Add a connection**. Click the MQTT connection that you want to use to publish a message in the middle of an integration.
4. On the **Choose an Action** page, click **Publish**.
5. In the **MQTT queue/topic name** field, specify the name of he queue or topic to publish the message to.
6. Click **Done** to add the connection to the integration.

3.10. CONNECTING TO REST APIS

In an integration, to connect to a REST API, you must have created a connector for that API by uploading a Swagger specification that describes the API. See Section 4.1, "Adding REST API client connectors".

When a connector for the REST API you want to connect to is available in Ignite, the steps for connecting to that REST API are:

- Section 3.10.1, "Register Ignite as an API client" if required

- Section 3.10.2, “Create an API client connection”
- Section 3.10.3, “Add an API client connection to an integration”

3.10.1. Register Ignite as an API client

Before Ignite creates an API client connector, it prompts you to indicate the API’s security requirements. For APIs that use OAuth, when Ignite creates the connector it also adds an entry for the API to the Ignite **Settings** page. This is where you provide the API client ID and the API client secret that authorize Ignite to access the API.

If the API you want to connect to does not use OAuth, skip this section and see Section 3.10.2, “Create an API client connection”.

To register Ignite as an authorized API client:

1. In Ignite:
   a. In the left panel, click **Settings**.
   b. Near the top of the **OAuth Application Management** page, where you see **`During registration, enter this callback URL:`**, copy that URL to the clipboard.
   c. Look for the name of the API you want to connect to and click its **Register** button to display its client ID and client secret fields.
2. In another browser window, you must register Ignite as an OAuth client of the API you want to connect to. The exact steps for doing this vary for each API service. Typically, the API service provides an OAuth custom application setting page. Go to that page.
3. On that page:
   a. Provide the Ignite callback URL, which you copied at the beginning of this procedure.
   b. Obtain the client ID and client secret that the API service assigns to your installation of Ignite.
4. In the Ignite **Settings** page, in the entry for the API service you are registering with, paste the assigned client ID and the client secret.
5. Click **Save**.

3.10.2. Create an API client connection

To create a connection to a REST API:

1. In Ignite, in the left panel, click **Connections** to display any available connections.
2. In the upper right, click **Create Connection** to display available connectors.
3. Click the connector for the API that you want to create a connection for.
4. Respond to prompts for additional information. The definition of the API determines what Ignite prompts for. For example, for an API that uses HTTP Basic Authorization, Ignite prompts for the user name and password to use to access the API. For an API that uses OAuth, Ignite displays a button for you to click so that Ignite can verify its registration credentials for connecting to the API.

5. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections.

6. In the **Description** field, optionally enter any information that is helpful to know about this connection.

7. In the upper right, click **Create** to see that the connection you created is now available.

3.10.3. Add an API client connection to an integration

Before you can add an API connection to an integration, you must create a connection to that API. See Section 3.10.2, “Create an API client connection”.

In this release, in an integration, a connection to an API can be the finish connection or a middle connection. It cannot be the start connection. The instructions below assume that Ignite is prompting you to select a finish connection or a connection that is not the start connection.

To add an API connection to an integraton:

1. On the page that displays available connections, click the API connection that you want to add to the integration.

2. Click the action that you want the connection to perform. The actions that are available are based on the resource operations specified in the Swagger file that was uploaded to Ignite and that describes the API you are connecting to.

3. Depending on the action you select, enter any parameters that Ignite prompts for.

4. Click **Done**.

3.11. CONNECTING TO SALESFORCE

To connect to Salesforce in an integration, see the following topics:

- Section 3.11.1, “Register Ignite as a Salesforce client”
- Section 3.11.2, “Create a Salesforce connection”
- Section 3.11.3, “Adding a Salesforce connection to an integration”

3.11.1. Register Ignite as a Salesforce client

You must register your installation of Ignite as an application that can access Salesforce. This lets you create any number of integrations that connect to Salesforce. In other words, you need to register a particular installation of Ignite with Salesforce only once.

In each Ignite environment, there can be only one registration of Ignite as a Salesforce client. However, while each Salesforce connection uses the same registration, it can use different user credentials.

Perform these steps to register Ignite as a Salesforce client:

1. In Ignite:

   a. In the left panel, click **Settings**.

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

c. To the right of the **Salesforce** entry, click **Register** to display the **Client ID** and **Client Secret** fields.

2. In another browser tab, log in to your Salesforce account and follow the steps below to create a connected app. These instructions assume that you are using the Salesforce Classic user interface. To switch from the Salesforce Lightning Experience interface, click your profile icon and select **Switch to Salesforce Classic**. For additional information, see the Salesforce documentation for Create a Connected App.

a. In Salesforce, in the upper right, click **Setup**.

b. In the left panel, select **Build** > **Create** > **Apps**.

c. Scroll down to **Connected Apps** and click **New**.

d. Enter the required information and then select **Enable OAuth Settings**.

e. In the **Callback URL** field, paste your Ignite URL, which you copied at the beginning of this procedure. For example: **`https://app-proj9128.7b63.fuse-ignite.openshiftapps.com/api/v1/credentials/callback`**.

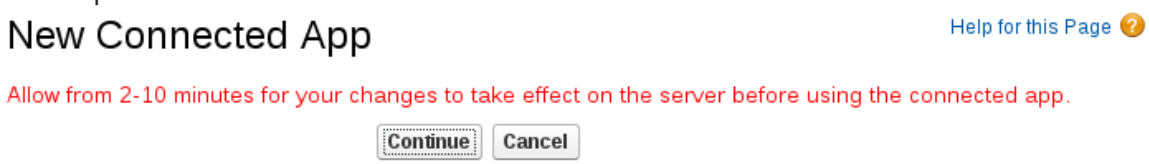
f. For **OAuth Scopes**, add:

- **Access and manage your data**
- **Allow access to your unique identifier**
- **Perform requests on your behalf at any time**

g. Select **Include ID token** and then **Include Standard Claims**.

h. Scroll down and click **Save**.

i. Scroll up to see that Salesforce indicates a short wait:



j. Click **Continue**.

k. Copy the consumer key that Salesforce provides.

3. Return to your Ignite installation **Settings** page and paste the Salesforce-provided consumer key into the Salesforce **Client ID** field.

4. Back in Salesforce, copy the consumer secret that Salesforce provides.

5. Return to your Ignite installation **Settings** page and paste the Salesforce-provided consumer secret into the Salesforce **Client Secret** field.

6. Click **Save** and then click **Ok**.

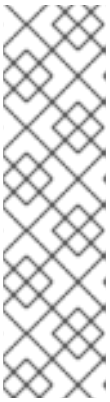
3.11.2. Create a Salesforce connection

A connection to Salesforce requires registration of Ignite as an application that can access Salesforce.

Be sure to wait 2 - 10 minutes after registering your Ignite installation as a Salesforce client before you try to create a Salesforce connection. After you create a Salesforce connection, you can use it in multiple integrations.

To create a Salesforce connection:

1. In the left panel, click **Connections** to display available connections.
2. In the upper right, click **Create Connection** to display the available connectors. A connector is a template for creating one or more connections.
3. Click the **Salesforce** connector.
4. Click **Connect Salesforce** to display a Salesforce authorization page. You might need to log in to Salesforce before you see the authorization page.



NOTE

The following error indicates that Salesforce does not have the correct Ignite callback URL:

```
error=redirect_uri_mismatch&error_description=redirect_uri%20must%20match%20configuration
```

If you get this error message, then in Salesforce, ensure that the Ignite callback URL is specified according to the instructions in Section 3.11.1, “Register Ignite as a Salesforce client”.

5. Click **Allow** to return to Ignite.
6. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections. For example, enter **`SF Connect 1`**.
7. In the **Description** field, optionally enter any information that is helpful to know about this connection. For example, enter **`Sample Salesforce connection that uses my Salesforce login credentials.`**
8. In the upper right, click **Create** to see that the connection you created is now available. If you entered the example name, you would see that **SF Connect 1** is now available.

3.11.3. Adding a Salesforce connection to an integration

You must create a Salesforce connection before you can add a Salesforce connection to an integration. If you did not already create a Salesforce connection, see Section 3.11.2, “Create a Salesforce connection”.

You must be creating an integration or updating an integration to add a connection to that integration. If you need to, see Section 5.2, “Procedure for creating an integration” or Section 7.10, “Updating integrations”.

The instructions below assume that Ignite is prompting you to select a start connection, a finish connection or a middle connection.

To add a Salesforce connection to an integraton:

1. On the page that displays available connections, click the Salesforce connection that you want to add to the integration. When the integration uses the connection you select to connect to Salesforce, Ignite uses the credentials defined in that connection.

2. Click the action that you want the selected connection to perform. Each Salesforce connection that you add to an integration performs only the action you choose.

3. Specify the Salesforce object that the action operates on, for example, it might be a contact, lead or price book entry. Click in the **Object** field to select from a list of Salesforce objects or enter the name of the object.

4. Click **Done** to add the connection to the integration.

3.12. CONNECTING TO SLACK

In an integration, you can connect to an instance of Slack and deliver a message to a particular user or to a channel. For example, this is useful when an integration downloads a file from an FTP server and processes it in some way. You can finish an integration by notifying a Slack channel or user that the process was successful.

To connect to Slack in an integration, create a Slack connection. You can then add that same connection to any number of integrations. Details are in the following topics:

- Section 3.12.1, “Create a Slack connection”
- Section 3.12.2, “Adding a Slack connection to an integration”

3.12.1. Create a Slack connection

To create a connection to an instance of Slack:

1. In Ignite, in the left panel, click **Connections** to display any available connections.

2. In the upper right, click **Create Connection** to display Ignite connectors.

3. Click the **Slack** connector.

4. In the **Slack webhook URL** field, enter the webhook URL of the Slack instance to send messages to. This is the only required parameter.

5. Optionally, enter values for additional parameters:

   a. In the **Sending user name for messages** field, enter the user name that the bot has when it sends messages to Slack.

   b. In the **Message avatar emoji** field, specify the emoji that the bot uses as the message avatar when it sends a message.

   c. In the **Message avatar icon URL** field, specify the URL of the avatar that the bot uses when it sends messages to Slack.

   If you specify an emoji and an icon URL, then the integration uses the icon URL.

6. Click **Validate**. Ignite immediately tries to validate the connection and displays a message that indicates whether validation is successful. If validation fails, revise the connection configuration values and try again.

7. If validation is successful, click **Next**.

8. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections. For example, enter **`Slack for Company Sales`**.

9. In the **Description** field, optionally enter any information that is helpful to know about this connection.

10. In the upper right, click **Create** to see that the connection you created is now available. If you entered the example name, you would see that **Slack for Company Sales** is now available.

3.12.2. Adding a Slack connection to an integration

In an integration, to send a message to a Slack channel or user, first create a Slack connection. You can then add that connection to any number of integrations as a finish connection or as a middle connection. It does not make sense to add a Slack connection that starts an integration because you must map the message content from a previous step to a Slack connection field. In other words, a data mapping step must be in the integration just before the Slack connection.

To add a Slack connection to an integration, you must be creating or editing an integration. If you are creating an integration, then Ignite might be prompting you to choose a finish connection. To add a middle connection, hover over the plus sign in the left panel in the location where you want to add the connection and select **Add a connection**.

To add a Slack connection:

1. Click the Slack connection that you want to add to the integration.

2. Select the action that you want the connection to perform.

   - Click **User name** to send a message to one user. To configure this action, in the **User name** field, specify the name of the user to send the message to.

   - Click **Channel** to publish a message on a channel. To configure this action, in the **Channel** field, specify the channel to publish the message to. The integration sends the integration data in a message to that channel.

3. Click **Next** to add the connection to the integration.

4. After you add all connections to the integration, add a data mapping step just before the Slack connection. In the mapping step, map a string from a previous step to the Slack **message** field. This string should contain the message that you want to send to the Slack user or channel. See Section 5.3.1, “Add a data mapping step”.

3.13. CONNECTING TO SQL DATABASES

In an integration, you can connect to any of the following types of SQL databases:

- Apache Derby

- MySQL

- PostgreSQL

You create a connection to the database that you want to access in an integration and then you add that connection to an integration.

To connect to other types of databases, you must upload a JDBC driver for that database.

See the following topics for details:

- Section 3.13.1, “Create a database connection”
- Section 3.13.2, “Add a database connection to an integration”
- Section 3.13.3, “Connecting to proprietary databases”

3.13.1. Create a database connection

You create a separate connection for each database that you want to connect to in an integration. You can use the same connection in multiple integrations.

A database connection operates on a database table that you specify or invokes a stored procedure that you specify. The database table or the stored procedure must exist when an integration connects to the database.

To create a database connection:

1. Ensure that the JDBC driver for the database that you want to connect to is on your classpath. If you uploaded a JDBC driver library extension to connect to a proprietary database, then the upload process puts the driver on your classpath. See Section 4.2.4, “Creating JDBC driver library extensions”.
2. In Ignite, in the left panel, click **Connections** to display any available connections.
3. In the upper right, click **Create Connection** to display Ignite connectors.
4. Click the **Database** connector.
5. Configure the connection by entering:
    a. In the **Password** field, enter the password associated with the user account you want to use to access the database.
    b. In the **Schema** field, enter the name of the schema for the database. If the connection URL specifies the schema, ensure that this field indicates the same schema as the connection URL. For example, enter **`sampledb`**.
    c. In the **Connection URL** field, enter the JDBC URL for the database that you want to connect to. For example, enter **`jdbc:postgresql:/ignite-db1234/sampledb`**.
    d. In the **Username** field, enter the name of the account that you want to use to access the database. Ensure that the specified password and user name are for the same account.
6. Click **Validate**. Ignite tries to validate the connection and displays a message that indicates whether validation is successful. If validation fails, revise the configuration details as needed and try again.
7. If validation is successful, click **Next**.

8. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections. For example, enter **`PostgreSQL DB 1`**.

9. In the **Description** field, optionally enter any information that is helpful to know about this connection. For example, enter **`Sample PostgreSQL connection that uses my login credentials.`**

10. In the upper right, click **Create** to see that the connection you created is now available. If you entered the example name, you would see that **PostgreSQL DB 1** is available.

3.13.2. Add a database connection to an integration

You must create a database connection before you can add a database connection to an integration. If you did not already create the database connection, see Section 3.13.1, “Create a database connection”.

See the instructions for where in the integration you want to access the database:

- Section 3.13.2.1, “Starting an integration by accessing a database”
- Section 3.13.2.2, “Accessing a database in the middle or to complete an integration”

3.13.2.1. Starting an integration by accessing a database

To start an integration by accessing a database:

1. In the Ignite panel on the left, click **Integrations**.

2. Click **Create Integration**.

3. On the **Choose a Start Connection** page, click the database connection that you want to use to start an integration.

4. On the **Choose an Action** page, click one of the following:

   - **Periodic SQL Invocation** obtains data by periodically invoking the SQL query you specify.

   - **Periodic Stored Procedure** obtains data by periodically invoking the stored procedure you specify or select.

5. If you selected **Periodic SQL Invocation**, in the **Query** field, enter a SQL **`SELECT`** statement to obtain one or more records. For example: **`SELECT * from my_db_table`**. The database table that contains the data you want must already exist.
   If you selected **Periodic Stored Procedure**, in the **Stored Procedure** field, select or enter the stored procedure to invoke to obtain the data of interest. The stored procedure you specify must already exist. The database administrator should have created any stored procedures you need to use in an integration.

6. In the **Period** field, enter an integer and indicate whether the unit is minutes, hours, or days. For example, if you specify **`5 minutes`** then the connection invokes the specified query or stored procedure every five minutes.

7. Click **Done**.

Ignite tries to validate the connection, which includes checking that a specified SQL query is syntactically correct and confirming that the query or stored procedure target data exists. If verfication is successful then Ignite adds the start connection to the integration. If verification fails then Ignite displays a message

about the problem. Update your input as needed and try again.

3.13.2.2. Accessing a database in the middle or to complete an integration

To finish an integration by accessing a database, add a database connection as the finish connection. To access a database in the middle of an integration, add a database connection between the start and finish connections. The instructions below assume that you are on the Ignite **Choose a Finish Connection** page or the **Choose a Connection** page.

To add a database connection:

1. Click the database connection for the database you want to access.
2. On the **Choose an Action** page, click one of the following:
   - **Invoke SQL** operates on data by executing the SQL statement you specify.
   - **Invoke Stored Procedure** operates on data by invoking the stored procedure you specify or select.
3. If you selected **Invoke SQL**, in the **SQL Statement** field:
   - For a middle connection, enter a SQL **`SELECT`** statement that obtains one record or enter a SQL **`INSERT`**, **`UPDATE`**, or **`DELETE`** statement that operates on one or more records. The database table that contains the data must already exist.
     In this release, when a database connection is a middle connection, a **`SELECT`** statement can obtain only one record. You should define the **`SELECT`** statement so that it obtains one record.
   - For a finish connection, enter a SQL **`INSERT`**, **`UPDATE`** or **`DELETE`** statement to operate on one or more records.

   If you selected **Invoke Stored Procedure**, in the **Stored Procedure** field, select or enter the name of the stored procedure to invoke to operate on the data of interest. The stored procedure you specify must already exist. The database administrator should have created any stored procedures you need to use in an integration.
   - See the information below about specifying placeholder parameters in queries.
4. Click **Done**.

Ignite tries to validate the connection, which includes checking that a specified SQL query is syntactically correct and confirming that the query or stored procedure target data exists. If verfication is successful then Ignite adds the connection to the integration. If verification fails then Ignite displays a message about the problem. Update your input as needed and try again.

Specifying parameters in queries

When you access a database in the middle of an integration or to complete an integration, you can specify placeholder parameters in the SQL query or there can be placeholders in the stored procedure. For example:

```
INSERT INTO TODO(task, completed) VALUES(:#param_1, :#param_2)
DELETE FROM TODO WHERE task LIKE :#param_3
```

To specify the values of these placeholders, add a data mapping step to your integration before the database connection. In the data mapping step, map the appropriate source data fields to the target data

fields, for example, map source data to the **`:#param_1`**, **`:#param_2`**, and **`:#param_3`** target fields. See Section 5.3.1, “Add a data mapping step”.

3.13.3. Connecting to proprietary databases

To connect to a proprietary SQL database, the main tasks that must be accomplished are as follows:

1. A developer creates a library extension that contains the JDBC driver for the database that you want to access in an integration. See Section 4.2.4, “Creating JDBC driver library extensions”.
2. The developer provides a **`.jar`** file that contains the library extension.
3. You upload that **`.jar`** file to Ignite. See Section 4.2.2, “Making custom features available”.
4. You create a connection to your database by selecting the Ignite **Database** connector and specifying the connection URL for your database. See Section 3.13.1, “Create a database connection”.
5. In an integration, you add the connection to your database. See Section 3.13.2, “Add a database connection to an integration”.

3.14. CONNECTING TO TWITTER

To connect to Twitter in an integration, see the following topics:

- Section 3.14.1, “Register Ignite as a Twitter client”
- Section 3.14.2, “Create a Twitter connection”
- Section 3.14.3, “Adding a Twitter connection to an integration”

3.14.1. Register Ignite as a Twitter client

You must register your installation of Ignite as an application that can access Twitter. This lets you create any number of integrations that connect to Twitter. In other words, you need to register a particular installation of Ignite with Twitter only once.

In each Ignite environment, there can be only one registration of Ignite as a Twitter client. However, while each Twitter connection uses the same registration, it can use different user credentials.

Perform these steps:

1. In Ignite:
   a. In the left panel, click **Settings**.
   b. Near the top of the **OAuth Application Management** page, where you see **`During registration, enter this callback URL:`**, copy the URL at the end of the sentence to the clipboard.
   c. To the right of the **Twitter** entry, click **Register** to display the **Client ID** and **Client Secret** fields.
2. In another browser tab, go to the Twitter **Application Management** web site at https://apps.twitter.com and do the following:

a. Confirm that the URL is **`apps.twitter.com`** and not just **`twitter.com`**.

b. If you are not already logged in to the Twitter **Application Management** site, log in.

c. Click **Create New App**.

d. In the **Name** field, enter a name for your new app. This name must be unique among all names of apps registered with Twitter.

e. In the **Description** field, enter helpful information. Twitter requires some input in this field.

f. In the **Website** field, paste the URL that you copied at the beginning of this procedure and remove **`api/v1/credentials/callback`** from the end of the URL.

g. In the **Callback URL** field, paste the URL again. It should be something like this: **`https://app-proj9128.7b63.fuse-ignite.openshiftapps.com/api/v1/credentials/callback`**.

h. Click **Yes** to agree to the Twitter developer agreement.

i. Click **Create your Twitter application**.

j. Click the **Keys and Access Tokens** tab.

k. Copy the **Consumer Key**.

3. On your Ignite installation **Settings** page, paste the Twitter consumer key into the Twitter **Client ID** field.

4. On the Twitter **Keys and Access Tokens** tab, copy the **Consumer Secret** and paste it into the Ignite Twitter **Client Secret** field.

5. Click **Save** and then click **Ok**.

3.14.2. Create a Twitter connection

A connection to Twitter requires registration of Ignite as an application that can access Twitter. If you did not already register Ignite, see Section 3.14.1, “Register Ignite as a Twitter client”.

After you create a Twitter connection, you can use it in multiple integrations.

To create a Twitter connection:

1. In Ignite, in the left panel, click **Connections** to display any available connections.

2. In the upper right, click **Create Connection** to display the available connectors. A connector is a template that you use to create one or more connections.

3. Click the **Twitter** connector.

4. Click **Connect Twitter** to display a Twitter authorization page. You might need to log in to Twitter before you see the authorization page.

5. Click **Authorize app** to return to Ignite.

6. In the **Connection Name** field, enter your choice of a name that helps you distinguish this connection from any other connections. For example, enter **`Twitter Connect 1`**.

7. In the **Description** field, optionally enter any information that is helpful to know about this connection. For example, enter **`Sample Twitter connection that uses my Twitter login credentials.`**

8. In the upper right, click **Create** to see that the connection you created is now available. If you entered the example name, you would see that **Twitter Connect 1** is now available.

3.14.3. Adding a Twitter connection to an integration

You must create a Twitter connection before you can add a Twitter connection to an integration. If you did not already create a Twitter connection, see Section 3.14.2, “Create a Twitter connection”.

You must be creating an integration or updating an integration to add a connection to that integration. If you need to, see Section 5.2, “Procedure for creating an integration” or Section 7.10, “Updating integrations”.

The instructions below assume that Ignite is prompting you to select a start connection, a finish connection or a middle connection.

To add a Twitter connection to an integraton:

1. On the page that displays available connections, click the Twitter connection that you want to add to the integration. When the integration uses the selected connection to connect to Twitter, Ignite uses the credentials defined in that connection.

2. Click the action that you want the selected connection to perform. Each Twitter connection that you add to an integration performs only the action you choose.

3. Optionally, enter the configuration information that Ignite prompts for. For example, the **Search** action prompts you to specify how often to search and keywords to search for.

4. Click **Done** to add the connection to the integration.

CHAPTER 4. ADDING AND MANAGING CUSTOMIZATIONS

Ignite lets you add API client connectors and custom steps to integrations. A custom step operates on integration data between connections. See the following topics for details:

- Section 4.1, “Adding REST API client connectors”
- Section 4.2, “Adding extensions”

4.1. ADDING REST API CLIENT CONNECTORS

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

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

- Section 4.1.1, “Requirements for API client connectors”
- Section 4.1.2, “About Swagger specification content”
- Section 4.1.3, “Creating API client connectors”
- Section 4.1.4, “Updating API client connectors”
- Section 4.1.5, “Deleting API client connectors”

After you create a REST API client connector, for details about using that connector in an integration, see Section 3.10, “Connecting to REST APIs”.

4.1.1. Requirements for API client connectors

After you upload a Swagger specification to Ignite, a connector to the API becomes available. An integrator can select the connector, create an API client connection, and then add that connection to an integration.

When Ignite creates an API client connector, it maps each resource operation in the Swagger specification to an action. The action name and action description comes from documentation in the Swagger specification.

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

An API that uses OAuth must have an authorization URL that takes a client callback URL as input. For Ignite to obtain authorization to access an API that uses OAuth, you provide the Ignite callback URL at the client API authorization URL. The details for doing this are described in Section 3.10.1, “Register Ignite as an API client”.

Ignite cannot create connectors for APIs that support the HTTP 2.0 protocol.

4.1.2. About Swagger specification content

The following topics provide information about the content of the Swagger specification that Ignite uses to create a connector to a REST API:

- Section 4.1.2.1, “Guidelines for Swagger specifications”
- Section 4.1.2.2, “Providing client credentials in parameters”
- Section 4.1.2.3, “Automatically refreshing access tokens”

4.1.2.1. Guidelines for Swagger specifications

The more detail that the Swagger specification provides, the more support Ignite 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, Ignite 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 Swagger specification before you upload it to Ignite. Identify the Swagger resource operations that will map to the actions you want the API connection to perform. In the Swagger specification, ensure that there is a JSON schema that specifies each operation’s request and response types.

If the Swagger specification 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 Swagger specification specifies **`consumes`** or **`produces`** parameters that define both **`application/json`** and **`application/xml`**, then the connector uses the JSON format.

4.1.2.2. Providing client credentials in parameters

When Ignite 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 Ignite 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 Ignite should pass client credentials as parameters, in the **`securityDefinitions`** section of the Swagger specification, 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 Ignite uses HTTP basic authentication to provide client credentials.

4.1.2.3. Automatically refreshing access tokens

If an access token has an expiration date, then Ignite integrations that use that token to connect to an application stop running successfully when the taken 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 Ignite automatically requests a new access token in the following situations:

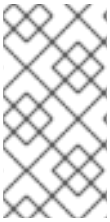
- When the expiration date has been reached.
- When HTTP response status codes that you specify are received.

IMPORTANT

This is a Technology Preview feature.

To specify that Ignite should automatically try to obtain a new access token in the situations described, in the **`securityDefinitions`** section of the Swagger specification, 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 Ignite receives a message from an OAuth2 provider and the message has one of these response status codes, then Ignite 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, Ignite does not retry the failed API operation after it receives the new access token.

4.1.3. Creating API client connectors

To create an API client connector:

1. In the Ignite 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 Swagger file that you want to upload.

   - Select **Use a URL** and paste the URL for your Swagger specification in the input field.

4. Click **Next**. If there is invalid or missing content, Ignite displays information about what needs to be corrected. Select a different Swagger file to upload or click **Cancel**, revise the Swagger file, and upload the updated file.
   If the specification is valid and complete, Ignite displays a summary of the actions that the connector will provide. This might include errors and warnings about the action specifications.

5. To revise the Swagger file, click **Cancel**, revise the Swagger file, and upload the updated file.
   If you are satisfied with the action summary, click **Next**.

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** — Ignite prompts you to enter:

      i. **Authorization URL** is the location for registering Ignite as a client of the API. Registration authorizes Ignite to access the API. See Section 3.10.1, "Register Ignite as an API client". The Swagger specification 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 Swagger specification or other documentation for the API should provide this URL. If it does not then you must contact the service provider.

7. Click **Next**. Ignite displays the connector's name, description, host, and base URL as indicated by the Swagger file. For connections that you create from this connector,

   - Ignite 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`.

   - Ignite applies the schema specified in the Swagger file to data mapping steps. If the Swagger file supports more than one schema then Ignite 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, Ignite generates one. You can upload an icon at a later time. When Ignite 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 Swagger file, edit the field value that you want to change. After Ignite creates a connector, you cannot change it. To effect a change, you need to upload an updated Swagger file so that Ignite can create a new connector.

9. When you are satisfied with the connector details, click **Create API Connector**.

For details about using your new API connector, see Section 3.10, “Connecting to REST APIs”.

4.1.4. Updating API client connectors

You cannot update an API client connector. If there is an update to the API’s Swagger specification, then you must upload an updated Swagger file and create a new API client connector.

To update integrations to use connections based on the updated Swagger specification:

1. Create a new API client connector based on the updated Swagger specification.
2. Create a new connection from the new connector.
3. Edit the integration to remove the old connection and add the new connection.
4. Publish the updated integration.

4.1.5. 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.

To delete an API client connector:

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**.

4.2. ADDING EXTENSIONS

Extensions let you add customizations to Ignite so you can integrate applications the way you want to.

The following topics provide details:

- Section 4.2.1, “About extensions”
- Section 4.2.2, “Making custom features available”
- Section 4.2.3, “Managing extensions”
- Section 4.2.4, “Creating JDBC driver library extensions”

4.2.1. About extensions

If Ignite does not provide a feature that you need, then a developer can code an extension that lets you integrate data the way you need to. An extension defines one of the following:

- A single custom connector for creating connections to a particular application or service that you want to integrate.
- One or more custom steps that operate on integration data between connections.

- A JDBC driver for connecting to a proprietary SQL database, such as Oracle.

Share your requirements with a developer who codes the extension. The developer gives you a `.jar` file that contains the extension. You then upload the `.jar` file in Ignite to make the custom connector, custom step(s), or JDBC driver available for use within Ignite.

An extension `.jar` file that you upload to Ignite 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.

For custom connectors and custom steps, information about coding the extension and creating the `.jar` file is in the Tooling User Guide.

For information about creating extensions that provides JDBC drivers, see Section 4.2.4, “Creating JDBC driver library extensions”.

4.2.2. Making custom features available

To make a custom feature available for use in an integration, you upload the extension to Ignite as follows:

1. In the left Ignite 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.
   A developer needs to make this file available to you. Ignite immediately tries to validate that the file contains an extension. If there is a problem, Ignite 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 Ignite 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, Ignite 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 Ignite 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 Ignite generates an icon.

   For a step extension, Ignite 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 publish an integration that has a connection that was created with the older driver, Ignite automatically uses the new driver instead.

6. Click **Import Extension**. Ignite makes the custom connector or custom step(s) available and displays the extension's details page.

See also:

- Section 3.1.5, "Creating connections from custom connectors"
- Section 5.3.5, "Add a custom step"
- Section 3.13.3, "Connecting to proprietary databases"

4.2.3. Managing extensions

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.

For details, see:

- Section 4.2.3.1, "Identifying integrations that use extensions"
- Section 4.2.3.2, "Updating extensions"
- Section 4.2.3.3, "Deleting extensions"

4.2.3.1. Identifying integrations that use extensions

Before you update or delete an extension, you should identify the integrations that use customizations provided by that extension. To do this:

1. In the left Ignite 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.

Ignite displays details about the extension including a list of any integrations that use a customization provided by the extension.

4.2.3.2. Updating extensions

To update an extension:

1. Obtain an updated `.jar` file for the extension from the developer.
2. In Ignite, in the left panel, click **Customizations**.
3. Click the **Extensions** tab.
4. At the right of the entry for the extension that you want to update, click **Update**.
5. Click **Browse**, select the updated `.jar` file, and click **Open**.
6. Confirm that the extension details are correct and click **Update**.

7. 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 stop (click **Unpublish**) and restart (click **Publish**) each integration that uses a customization defined in the updated extension. See Section 7.3, “Unpublishing integrations”.

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.

4.2.3.3. 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 publish an integration that uses a customization that was provided by that extension.

To delete an extension:

1. In the left Ignite 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.

There might be unpublished or draft integrations that use a customization provided by an extension that you delete. To publish one of these integrations, you will need to edit the integration to remove the customization. See Section 4.2.3.1, “Identifying integrations that use extensions” and Section 7.10, “Updating integrations”.

4.2.4. Creating JDBC driver library extensions

To connect to a SQL database other than Apache Derby, MySQL, and PostgreSQL, a developer creates an extension that wraps a JDBC driver for the database you want to connect to. After uploading this extension to Ignite, the Ignite-provided **Database** connector can access the driver to validate and create connections to the proprietary database.

The Syndesis open source community provides a project for creating an extension that wraps a JDBC driver. This kind of extension is referred to as a library extension because the result of uploading the extension is that you can use the built-in **Database** connector to create connections to your proprietary database. You do not create a new connector for your particular database.

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.

To use the Syndesis project, you must have a GitHub account.

To create a JDBC driver library extension:

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 Ignite.

CHAPTER 5. CREATING INTEGRATIONS

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

It is assumed that you are familiar with the information in these topics:

- Section 1.5, “Planning integrations”
- Section 1.6, “Workflow for creating an integration”

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

- Section 5.1, “Preparing to create an integration”
- Section 5.2, “Procedure for creating an integration”
- Section 5.3, “Adding steps between connections”

5.1. PREPARING TO CREATE AN INTEGRATION

Preparation for creating an integration starts with answers to the questions listed in Section 1.5, “Planning 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 Ignite as a client that is authorized to access that application. See Section 3.1.2, “Obtaining authorization to access applications”.
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 integration.
3. For each application that you want to integrate, create a connection. See Section 3.1.1, “About creating connections”.

5.2. PROCEDURE FOR CREATING AN INTEGRATION

To create an integration:

1. In the left panel in Ignite, 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 you want to use to start the integration. When this integration is running, Ignite 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. Click **Done** to add the start connection.

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, Ignite 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. Click **Done** to add the finish connection.
5. Optionally, add one or more connections between the start connection and the finish connections. 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 Section 5.3, “Adding steps between connections”.
7. When the integration contains all needed steps, click **Save as Draft** or **Publish** according to whether you want to start running the integration.
8. In the **Integration Name** field, enter a name that distinguishes this integration from any other integrations.
9. In the **Description** field, enter a description, for example, you can indicate what this integration does.
10. If you are ready to start running the integration, in the upper right, click **Publish**. Otherwise, click **Save as Draft**.

In the Ignite **Integrations** page, you can see your new integration in the list of integrations. If you published the integration, then you can see that Ignite is in the process of publishing it. It may take a few moments for the status of the integration to become **Published**, which means that it is running. If you saved the integration as a draft, then **Draft** appears on the integration’s entry.

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

5.3. ADDING STEPS BETWEEN CONNECTIONS

After you add connections to an integration, you can optionally add steps between connections. Each step operates on data obtained from the previous connection(s) and any other previous steps and makes the data available to the next step in the integration.

Often, you must map data fields that are received from a connection to data fields that the next connection in the integration can operate on. After you add all connections to your integration, check the integration visualization panel on the left. For each connection that requires data mapping before it can operate on the input data, Ignite displays  . Click this icon to see **Data Type Mismatch: Add a data mapping step before this connection to resolve the difference.** Click the link in the message to display the **Configure Mapper** page in which you specify the data mapping step.

For details about adding additional steps, see the following topics:

- Section 5.3.1, “Add a data mapping step”

- Section 5.3.2, “Add a basic filter step”
- Section 5.3.3, “Add an advanced filter step”
- Section 5.3.4, “Add a log step”
- Section 5.3.5, “Add a custom step”

5.3.1. Add a data mapping step

You can add a step to an integration to map data from the previous connection(s) and any other steps to the next connection. For example, suppose the integration data contains a **`Name`** field and the next connection in the integration has a **`CustomerName`** field. You need to map the source **`Name`** field to the target **`CustomerName`** field.

When you add a data mapper step you might be creating a new integration or editing an integration. The flow of the integration appears in the left panel.

To add a data mapper step:

1. In the left panel, where you want to add a data mapper step, hover over the  .
2. In the popup that appears, click **Add a Step**.
3. On the **Choose a Step** page, click **Data Mapper**.

For details, see Chapter 6, *Mapping data to fields for the next connection*.

5.3.2. Add a basic filter step

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

Add all connections to your integration before you add additional steps. When you add a step, Ignite operates on the data it receives from the previous connection(s) and any other previous step(s).

If you cannot define the filter you need in a basic filter step, see Section 5.3.3, “Add an advanced filter step”.

You can add a step when you are creating an integration or editing an integration. The flow of the integration appears in the left panel. To add a filter step:

1. In the left panel, where you want to add a filter step to the integration, hover over the  and in the popup that appears, click **Add a Step**.
2. On the **Add a Step** page, 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. **For this field**: In the field on the left, 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 the integration 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 **text** field. A dropdown box appears with a list of available fields. Select the field of interest from the list.

   b. **This condition must be satisfied**: In the middle field, select a condition from the dropdown box. The setting defaults to **Contains**. For the integration 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**: In the third field, 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 integration.

5.3.3. Add an advanced filter step

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

Add all connections to your integration before you add additional steps. When you add a step, Ignite operates on the data it receives from the previous connections and any additional step(s).

When you add a step you might be creating a new integration or editing an integration. The flow of the integration appears in the left panel.

To add an advanced filter step:

1. In the left panel, where you want to add an advanced filter step to the integration, hover over the  and in the popup that appears, click **Add a Step**.

2. Select **Advanced Filter**.

3. In the edit box, use the Camel Simple Expression language to specify a filter expression.

4. Click **Done**.

5.3.4. Add a log step

Ignite provides log information for each integration version that it executes. To learn what information is automatically logged, see Section 7.5, “Viewing integration log information”.

To log additional information between any two steps, add a log step to the integration. For each message that it receives, a log step can provide one or more of the following:

- The message’s header, which provides metadata about the message.
- 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.

To add a log step, you must be creating a new integration or editing an integration. The integration must already have its start and finish connections.

To add a log step:

1. In the integration visualization panel on the left, hover over the  at the location where you want to add a log step.
2. In the popup, click **Add a Step**.
3. On the **Choose a Step** page, click **Log**.
4. 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, Ignite resolves the expression and logs the resulting text.
5. When log step configuration is complete, click **Done**.
6. When the integration is complete, publish it to start seeing output from the new log step.

After an integration that has a log step has been executed, output from its log step appears in the integration’s **Activity** tab. See Section 7.5, “Viewing integration log information”.

5.3.5. Add a custom step

If Ignite does not provide a step that you need in an integration, a developer can define one or more custom steps in an extension. A custom step operates on integration data between connections. See Section 4.2.2, “Making custom features available”.

You add a custom step to an integration in the same way that you add a built-in step. Create an integration, choose the start and finish connections, add other connections as needed and then add additional steps. When you add a step, Ignite operates on the data it receives from the previous step(s).

When you click **Add a Step**, the list of available steps includes any custom steps that are defined in extensions that were uploaded to your installation of Ignite.

At the top of the list of steps, in the **Name** field, you can select **Custom Steps** to display only custom steps.

Click the custom step that you want to add to the integration. Ignite prompts you for any information required to perform the step. This information varies for each custom step.

CHAPTER 6. MAPPING DATA TO FIELDS FOR THE NEXT CONNECTION

In most integrations, you need to map data fields that have been obtained or processed to data fields that the next connection can process. Ignite provides a data mapper to simplify doing this. In an integration, 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:

6.1. ABOUT MAPPING DATA

The data mapper displays:

- **Sources** - a list of the data fields that are obtained or processed in each previous step
- **Target** - a list of the data fields that are processed in the next connection

According to the needs of your integration, you can:

- Map one source field to one target field. This is the default mapping behavior. For example, map the **`Name`** field to the **`CustomerName`** field.
- Combine multiple source fields into one target field. For example, map the **`FirstName`** and **`LastName`** fields to the **`CustomerName`** field.
- Separate a source field into multiple target fields. For example, map the **`Name`** field to the **`FirstName`** and **`LastName`** fields.
- Transform the target field to specify how Ignite stores it. For example, pad or trim the target field value, capitalize the first letter in a string, or create a time stamp.

6.2. FINDING THE DATA FIELD YOU WANT TO MAP

In an integration that has a few steps or that operates on a small set of data, it is probably easy to find the data field you want to map. But in an integration that has many steps, or that has a step that accesses a large set of data, the list of data fields might be very long. To quickly find the data field you want to map, you can:

- 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 an integration, Ignite numbers and renumbers them to indicate the order in which the integration processes the steps. When you are creating or editing an integration, these numbers are visible in the integration visualization panel on the left. When you add 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 an integration 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.

6.3. IDENTIFYING WHERE DATA MAPPING IS NEEDED

To identify where data mapping is needed:

1. When you are creating or editing an integration, add all connections to the integration.
2. In the integration visualization panel on the left, look for any  icons.
3. 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.

To see the input type or output type for a particular connection:

1. In the Ignite left panel, click **Integrations**.
2. In the list of integrations, identify the integration that has the connection whose input type or output type you want to know.
3. At the right of that integration's entry click  .
4. Select **Edit**.
5. In the integration's visualization panel, to the right of a connection, click  to display information about that connection, including its input and/or output type.

6.4. MAPPING ONE SOURCE FIELD TO ONE TARGET FIELD

The default mapping behavior maps one source field to one target field. To do this:

1. In the **Sources** panel, click the data field you want to map from.
   You might need to expand an integration 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 you want to map to.

The data mapper displays a line that connects the two fields you just selected.

To confirm that the mapping is defined, in the upper right, click  to display the defined mappings.

Click  again to display the data field panels.

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**.

6.5. COMBINING MULTIPLE SOURCE FIELDS INTO ONE TARGET FIELD

To combine the content from multiple source fields into one target field:

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

2. In the **Mapping Details** panel, under **Action**, select **Combine**.

3. In the **Mapping Details** panel, under **Source**, start to overtype `[None]` with the name of the first source field that you want to map into the target field.

4. When the name of the first source field appears, click it. The data mapper displays a line from the first source field to the target field.

5. For each additional source field:

   a. In the **Mapping Details** panel, click **Add Source**.

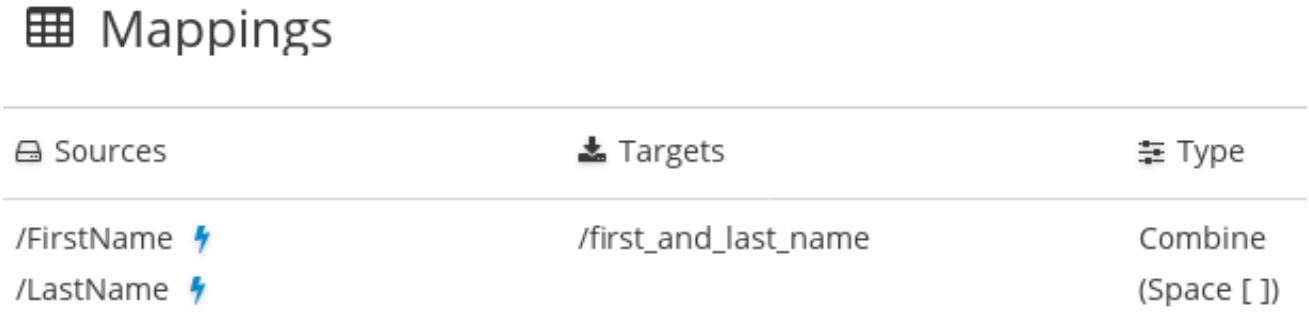
   b. Start to overtype `Search` with the name of the next source field.

   c. When the name of the next source field appears, click it. The data mapper displays another line to the target field but this line is from this source field.

6. 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 source fields. The default is a space.

In the data mapper, blue lines indicates the current focus.

To confirm that the mapping is correctly defined, 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:



.

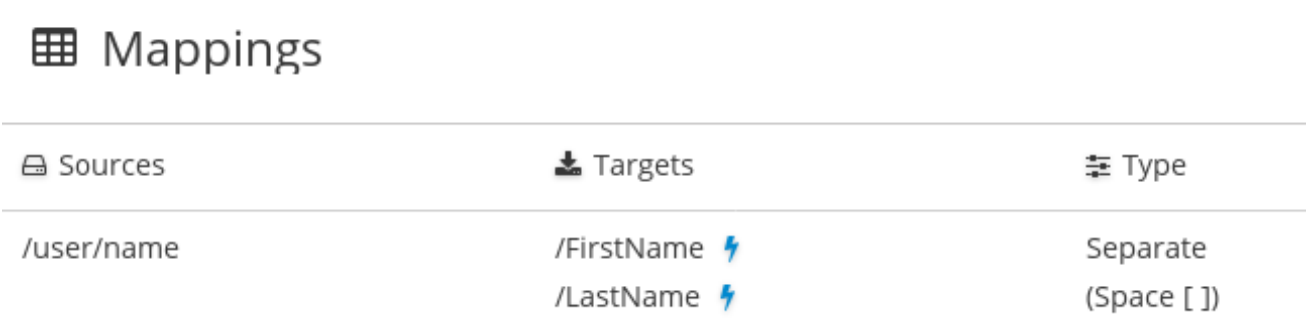
6.6. SEPARATING ONE SOURCE FIELD INTO MULTIPLE TARGET FIELDS

To separate and map the content in one source field to multiple target fields:

1. In the **Sources** panel, click the field you want to map.
2. If the **Mapping Details** panel is not already visible to the right of the **Target** panel, in the upper right, click the plust sign to display it.
3. In the **Mapping Details** panel, under **Action**, click the down caret and select **Separate**.
4. Below that, under **Targets**, click in **`Search`** and start to type the name of the first target data field.
5. When the name of the first target data field appears, click it. The data mapper displays a line from the source field to the first target field.
6. For each additional target field:
    a. In the lower right, click **Add Target**.
    b. In the new target, click in **`Search`** and start to type the name of the next target field.
    c. When the name of the next target data field appears, click it. The data mapper displays another line from the same source field, but this time it goes to this target data field.
7. In the **Mapping Details** panel, in the **Separator** field, accept or select the character that indicates where to separate the source data field value. The default is a space.

In the data mapper, blue lines indicates the current focus.

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:



.

6.7. TRANSFORMING TARGET DATA

After you define a mapping, you can transform the data in the target field to define how you want to store the date. For example, you could specify the **Capitalize** transformation to ensure that the first letter of a data value is uppercase.

To transform a target field:

1. Create the mapping whose target you want to transform.
2. In that mapping, click the target field you want to transform.
3. In the **Mapping Details** panel, click **Add Transformation**.
4. Click the down caret to display the list of transformations.
5. Click the transformation you want to perform.
6. If the transformation requires any input parameters, specify them in the appropriate input fields.

See Section 6.8, “Descriptions of available transformations”.

6.8. 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.

Transformation	Input Type	Output Type	Parameter (* = required)	Description
`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.
`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 each of the **`fromUnit`** and **`toUnit`** parameters, specify a string whose value is **`Square Foot`**, **`Square Meter`**, or **`Square Mile`**.

Transformation	Input Type	Output Type	Parameter (* = required)	Description
ConvertDistanceUnit	number	number	**fromUnit** * **toUnit** *	Convert a number that represents a distance to another unit. For each of the **fromUnit** and **toUnit** parameters, specify a string whose value is **Foot**, **Inch**, **Meter**, **Mile**, or **Yard**.
ConvertMassUnit	number	number	**fromUnit** * **toUnit** *	Convert a number that represents mass to another unit. For each of the **fromUnit** and **toUnit** parameters, specify a string whose value is **Kilo Gram** or **Pound**.
ConvertVolumeUnit	number	number	**fromUnit** * **toUnit** *	Convert a number that represents volume to another unit. For each of the **fromUnit** and **toUnit** parameters, specify a string whose value is **Cubic Foot**, **Cubic Meter**, **Gallon US Fluid**, or **Liter**.
CurrentDate	None	date	Note	Return the current date.
CurrentDateTime	None	date	None	Return the current date and time.

Transformation	Input Type	Output Type	Parameter (* = required)	Description
`CurrentTime`	None	date	None	Return the current time.
`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.
`Floor`	number	number	None	Return the whole number floor of a number.

Transformation	Input Type	Output Type	Parameter (* = required)	Description
`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.
`GenerateUUID`	None	string	None	Create a string that represents a random UUID.
`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.

6.9. 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.

To view mappings when you are already in the data mapper, in the upper right, click  .

To dismiss the list of mappings and redisplay the source and target fields, click  again.

To view mappings when you are editing an integration but you are not adding or editing a data mapper step:

1. In the integration visualization panel on the left, click the data mapper step for which you want view the defined mappings.
2. In the data mapper, in the upper right, click  .

To view mappings when you are not editing a data mapper step:

1. In the left panel, click **Integrations**.
2. In the entry for the integration whose data mappings you want to view, on the right, click the  .
3. In the popup menu, click **Edit**.
4. In the integration visualization panel on the left, click the data mapper step for which you want view the defined mappings.
5. In the data mapper, in the upper right, click  .

CHAPTER 7. MANAGING INTEGRATIONS

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

7.1. 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 initial running version with the updated version. To facilitate this, for each integration, Ignite maintains multiple versions as well as each version’s state. The following topics provide an understanding of the behavior to help you manage your integrations.

7.1.1. Understanding integration versions

In each Ignite environment, each integration can have multiple versions. The benefit of multiple integration versions is that if you publish a version that does not work correctly, then you can return to running a correct integration. To do that, you unpublish (stop) the incorrect version and republish a version that runs the way you want it to.

Ignite assigns a new version number each time it publishes 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. Ignite unpublishes the running integration, which stops it, and publishes the updated 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.

See also: Section 7.2, “Publishing integrations” and Section 7.4, “Republishing older integration versions”.

7.1.2. Understanding integration states

For a given integration:

- There can be exactly one draft version.
- There can be exactly one published version. A published version is running.

Each version of an integration is always in one of the following states:

State	Description
Draft	**Draft** is always the initial state of a new version of an integration. Only one version of an integration can be in the **Draft** state. That is, you can update one version of an integration at a time.
Published	A **Published** version is running. When you publish an integration, Ignite builds the integration and starts running it. Only one version of an integration can be running. That is, only one version at a time can be in the **Published** state.
Unpublished	An **Unpublished** version is not running. It might have been published and then unpublished, but it also might never have been published. If no version of this integration is in the **Published** state, then you can publish an unpublished version to run it. If you publish a previously published version, then the newly published version has the same version number as when the integration was previously published. If you publish a previously unpublished version, then the published integration gets the next available version number.
Error	An integration version that is in the **Error** state encountered an error while being published or while running. The error suspended publication or execution. If this happens, try publishing an earlier integration version that ran correctly. Alternatively, you can contact technical support for help. To do that, in any Ignite page, in the upper right, click the  icon and select **Support**.

7.1.3. Viewing integration history

To view a list of the versions of an integration:

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.

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 published version. For each version, you can also see the date on which it was last published.

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

7.2. PUBLISHING INTEGRATIONS

Publishing an integration builds and deploys the integration runtime. The integration starts running. Exactly one version of an integration can be running at one time.

To publish an integration, do 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 or an undeployed version of an integration:
  1. In the left Ignite panel, click **Integrations**.
  2. In the list of integrations, click the entry for an integration whose status is **Draft** or **Unpublished**.
  3. On the integration's summary page, in the **Details** tab, identify the integration version that you want to publish.
  4. On the right of that entry, click  and select **Publish**.

7.3. UNPUBLISHING INTEGRATIONS

Each integration can have exactly one version that is running. A running version is in the **Published** state. To stop running an integration, you unpublish it.

To unpublish an integration:

1. In the left Ignite 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 **Published**.
3. At the far right of this integration's entry, click  and select **Unpublish**.

Ignite stops running the integration. **Unpublished** appears in the integration's entry in the list of integrations.

7.4. REPUBLISHING 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.

To republish an older integration version:

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. Ignite 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 **Unpublish**.
4. Click **OK** to confirm that you want to stop running this version.
5. Wait for **Unpublished** to appear to the right of the integration name near the top of the page.
6. Optionally, 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. For details, see Section 7.10, “Updating integrations”.
    c. When updates are complete, in the upper right, click **Publish**, and then click **OK** 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 **Publish**.
8. Click **OK** to confirm that you want to publish this version of the integration.

Ignite publishes the integration, which takes a few minutes. When the integration is running, then **Published version *n*** appears to the right of the integration’s name.

7.5. VIEWING INTEGRATION LOG INFORMATION

Ignite provides log information for each integration version that it executes. To see this information:

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

For each integration step, Ignite 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

To log 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 version that you want to view log information for. You view Ignite information for a log step in the same way that you view Ignite information for any other step.

To add a log step, see Section 5.3.4, "Add a log step".

7.6. VIEWING INTEGRATION METRICS

To view integration metrics:

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**.

Ignite provides the following metrics:

- **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.

7.7. VIEWING SYSTEM METRICS

System metrics appear on the Ignite home page. To see them, click **Home** in the left panel. Ignite updates the following metrics every 5 seconds:

- The number of integrations that are defined in this environment regardless of the integration state. 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.

7.8. TESTING INTEGRATIONS

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

To test an integration in a different Ignite environment:

1. See Section 7.11.1, “About copying integrations”.
2. Export the integration from the development environment. See Section 7.11.2, “Exporting integrations”.
3. Import the integration into the test environment. See Section 7.11.3, “Importing integrations”.

7.9. TROUBLESHOOTING INTEGRATION EXECUTION

If an integration stops working, check its logs and activity details. See Section 7.5, “Viewing integration log information” and Section 7.1.3, “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 visual integration flow, click the icon for the application that you want to reconnect to.
4. In the connection’s details page, click **Reconnect**.
5. Respond to that application’s OAuth workflow prompts.
   Ignite 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, republish the integration.

If reconnection is not successful, try this:

1. Re-register Ignite as a client of the application. See Section 3.1.2, “Obtaining authorization to access applications”.
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.

7.10. UPDATING INTEGRATIONS

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

1. In the left Ignite 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**.

In the left panel, you can see that each step in the integration is represented by an icon that indicates whether it is a connection or a data operation between connections. Update the integration as needed:

- To add a step, in the left panel, hover over the plus sign that is in the location where you want to add it. Click **Add a Connection** or **Add a Step**.
- To delete a step, in the left panel, click  to the right of the step that you want to delete.
- 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.

See also: Section 7.2, "Publishing integrations".

7.11. COPYING INTEGRATIONS TO OTHER ENVIRONMENTS

To publish 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 7.11.1, "About copying integrations"
- Section 7.11.2, "Exporting integrations"
- Section 7.11.3, "Importing integrations"

7.11.1. About copying integrations

Each Ignite 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 Ignite 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

Ignite 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 Ignite 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.	Ignite 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.	Ignite 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 installation of Ignite as a client of the application that this connection accesses.

7.11.2. Exporting integrations

To export an integration:

1. In the left panel of Ignite, 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**.

Ignite downloads a zip file to your local **`Downloads`** folder. To import the integration into another Ignite environment, open that environment and import this zip file.

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

7.11.3. Importing integrations

To import an integration:

1. Open the Ignite 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 Ignite imports the file(s), click **Done**. Ignite 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 this installation of Ignite with the application. The steps vary for each application. See the appropriate topic:

   - Section 3.6.1, "Register Ignite as a Dropbox client"
   - Section 3.10.1, "Register Ignite as an API client"
   - Section 3.11.1, "Register Ignite as a Salesforce client"
   - Section 3.14.1, "Register Ignite as a Twitter client"

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**.

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 as Draft** 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, Ignite updates the integration to use the updated connections.

7.12. DELETING INTEGRATIONS

To delete an integration:

1. In the left Ignite 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.

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

CHAPTER 8. INSTALLING IGNITE ON OPENSHIFT CONTAINER PLATFORM

This topic provides information and instructions for installing Ignite on OpenShift Container Platform (OCP) on premise. The main steps are:

1. Ensure that you meet the prerequisites.
2. Decide how you want to install Ignite with regard to the OpenShift project to install into, the OpenShift route for Ignite, and the level of accessibility of OpenShift logs.
3. Download the installation script.
4. Invoke the installation script with the command that implements your decisions.
5. Confirm that Ignite is running.

Prerequisites

You must be running OCP on premise.

You must be connected to the OCP cluster in which you want to install Ignite.

Decisions

To correctly specify the installation command, you need to decide on the answers to these questions:

- Into which OpenShift project do you want to install Ignite?
  The default is that the installation script installs Ignite into the current project.

  If the project into which you want to install Ignite

  - Exists
    The installation script will prompt you to confirm that it is okay to delete the project's content before installing Ignite. You must confirm deletion of the project's content for the installation script to continue.
  - Does not exist
    You can create it now so that it is the current project. Alternatively, when you run the installation script, you can specify a project name and the script will create the project.
- Do you want to install Ignite into the current OpenShift project or do you want to specify the project into which you want to install Ignite in the command that you invoke to do the installation?
- Do you want the installation script to calculate the OpenShift route by which Ignite 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 Ignite integrations by clicking a link in the Ignite user interface?
  While you can always access OpenShift logs manually, you might want to enable direct access by means of a link in the Ignite user interface.

  The default is that the installation script does not enable this link.

Download

Download the Ignite installation script to the current local directory by invoking the following command:

```
$ wget https://raw.githubusercontent.com/syndesisio/fuse-ignite-install/1.3/install_ocp.sh
```

Installation

To install Ignite into the current project, under the OpenShift route chosen by the installation script, without enabling the link from Ignite to OpenShift logs, invoke the following commands:

1. Ensure that the current project is the project into which you want to install Ignite:
   `$ oc project`

2. In the directory in which you downloaded the installation script, invoke the installation script as follows:
   `$ bash install_ocp.sh`

To install Ignite into a project that you specify, under an OpenShift route that you specify, and also enable the link from Ignite to OpenShift logs, invoke a command such as the following:

```
$ bash install_ocp.sh \
    --project ignite \
    --route ignite.6a63.fuse-ignite.openshiftapps.com \
    --console https://console.fuse-ignite.openshift.com/console
```

According to how you want to install Ignite, you can specify

- All three options
- Any two options
- Any one of the options
- No options

The installation script uses the default for an option that you do not specify.

Confirm installation

To confirm that installation was successful:

1. 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 the script displays the calculated route near the end of its execution. Replace **`openshift-route`** with the value that the script provided.

2. 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 Ignite home page appears either immediately or after you log in to the OpenShift console.

Deleting your Ignite project

If you want to delete your Ignite project, invoke the **`delete project`** command. For example, to delete a project whose name is **`ignite`**, enter the following command:

```
$ oc delete project ignite
```