

## Resources

Date published: 2021-04-06

Date modified: 2025-09-30

# CLOUDERA

# Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

# Contents

<b>The Resources view.....</b>	<b>5</b>
--------------------------------	----------

<b>Managing workspace resources.....</b>	<b>5</b>
--	----------

Deployments.....	6
Viewing data flow in NiFi.....	6
Starting a flow.....	6
Stopping a flow.....	7
Changing flow version.....	7
Downloading NiFi application log.....	9
Suspending a deployment.....	10
Resuming a deployment.....	10
Export deployment configuration.....	11
Terminating a deployment.....	11
Flow drafts.....	12
Opening a draft for editing.....	12
Starting, stopping, and restarting a test session.....	12
Publishing a flow draft.....	12
Reassigning a flow draft to a different project.....	13
Deleting a flow draft.....	13
Parameter groups.....	13
Modifying parameters in a parameter group.....	14
Deleting a parameter group.....	14
Importing a parameter group to a flow draft.....	15
Duplicating a parameter group.....	15
Reassigning a parameter group to a different project.....	15
Inbound connections.....	16
Renewing the certificate for an inbound connection endpoint.....	16
Reassigning an inbound connection endpoint to a different project.....	16
Create an Inbound Connection Endpoint during flow deployment.....	17
Connecting applications to an endpoint.....	20
TLS keys and certificates.....	21
Inbound connections.....	21
Tutorial: MiNiFi to Cloudera Data Flow flow deployment.....	22
Tutorial: Invoking an HTTP endpoint with curl.....	32
Using Inbound Connections with an external load balancer.....	37
Custom NAR configs.....	39
Validating a custom NAR config.....	39
Reassigning a custom NAR config to a different project.....	40
Custom Python configs.....	40
Validating a custom Python config.....	40
Reassigning a custom Python config to a different project.....	40

<b>Parameter groups.....</b>	<b>41</b>
------------------------------	-----------

<b>Inbound connections.....</b>	<b>44</b>
---------------------------------	-----------

Create an Inbound Connection Endpoint during flow deployment.....	45
---	----

Connecting applications to an endpoint.....	48
TLS keys and certificates.....	49
Tutorial: MiNiFi to Cloudera Data Flow flow deployment.....	49
Tutorial: Invoking an HTTP endpoint with curl.....	59
Using Inbound Connections with an external load balancer.....	64
Configure an Application Gateway in Azure.....	65
<b>Best practices for building custom components.....</b>	<b>66</b>
Best practices for packaging custom Python processors [Technical Preview].....	67
Preparing cloud storage to deploy custom processors.....	69
<b>Python scripts.....</b>	<b>70</b>
Upload and run Python scripts in flow deployments.....	70
Install custom Python libraries in flow deployments.....	71

## The Resources view

The Resources page is the central place for accessing Cloudera Data Flow resources across environments.

### About this task

On the **Resources** you see all workspaces where you have DFFlowUser role. The number of resources is listed by type on each workspace card. To view or manage a resource type in a given workspace, click on the number displayed under the selected resource type.

The following resource types are listed:

- Deployments
- Flow drafts
- Parameter groups
- Inbound connections
- Custom NAR configurations
- Custom Python configurations

You can filter this view by starting to type the name of a workspace in the search box.



**Note:** If you do not see the workspaces you expect, check that you have been granted the proper roles for environments.

To manage workspace resources, select a workspace from the list. By clicking the blue number under the resource type, you will be redirected directly to the tab of that resource type on the **Workspace Resources** page. By default, the page opens on the **Deployments** tab.

## Managing workspace resources

You can access all workspace resources accessible to you through the Resources view. Select a workspace to view, manage, and reassign resources.

### Before you begin

You must have the DFAdmin role for the environments where you want to manage resources.

### About this task

You can manage the following types of resources:

- Flow deployments

- Flow drafts
- Parameter groups
- Inbound connections
- Custom NAR configurations
- Custom Python configurations

The actions available to you depend on the resource type you select.

### Procedure

1. Open Cloudera Data Flow by clicking the Data Flow tile in the Cloudera sidebar.
2. Select Resources.
3. Select the workspace where you want to manage resources.



**Tip:** Start typing the resource name in the search bar to filter for a workspace.

4. On the **Workspace Resources** page select the tab with the type of resource you want to manage.

## Deployments

On the Deployments tab you can view and manage deployments within a workspace.

### Viewing data flow in NiFi

You can go to the NiFi cluster where your flow is deployed and view or edit the data flow.


#### About this task

When you access the NiFi cluster, the ability to view or edit the flow is based on your Cloudera Data Flow authorizations. The DFFlowUser role has read-only privileges. The DFFlowAdmin role has full privileges.

#### Before you begin

You must have deployed a data flow in Cloudera Data Flow.

### Procedure

1. Select the **Deployment** that you want to manage.
2. Click  [Options] View in NiFi .

The UI for the NiFi cluster where your flow is deployed opens.

3. View your data flow or edit it based on your NiFi privileges.

If you edit the flow in NiFi and want the changes to exist in a new deployment, perform the following steps:

- a) Download the flow as a flow definition.  
For more information, see *Downloading a flow definition from NiFi*.
- b) Import the flow definition (as a new flow definition or as a new version of an existing flow definition).  
For more information, see *Importing a flow definition to Cloudera Data Flow*.
- c) Deploy the flow definition.  
For more information, see *Deploy a flow*.

### Starting a flow

You can start a stopped flow a Cloudera Data Flow deployment.


### About this task

Starting a flow deployment starts all processors of a Cloudera Data Flow deployment.

### Before you begin

- You must have a stopped flow deployment in Cloudera Data Flow.
- You must have DFFlowAdmin permission.

### Procedure

1. Select the **Deployment** that you want to manage.
2. Click  [Options] Start flow .  
The Start [Deployment Name] pop-up appears.
3. Confirm your choice by clicking Start Flow.

## Stopping a flow

Stopping the flow of a Cloudera Data Flow deployment temporarily pauses the NiFi flow.

### About this task


Stopping a flow results in the following:

- All processors are stopped and no data processing happens within the NiFi flow.
- KPI alerts are stopped. Your KPI alerts are activated again when the flow is restarted.
- Any active KPI alerts are resolved.
- All underlying cloud resources remain allocated for the Cloudera Data Flow deployment.
- You can modify deployment configuration while the flow is stopped.
- Stopped flows are still billable however if auto-scaling is enabled for the flow, a certain amount of cost reduction may occur.

### Before you begin

You must have deployed a flow definition in Cloudera Data Flow.

### Procedure

1. Select the **Deployment** that you want to manage.
2. Click  [Options] Stop flow .  
The Stop [Deployment Name] pop-up appears.
3. Click Stop Flow to stop the flow deployment.

## Changing flow version

Learn how to change the flow definition version of a running flow deployment. Using the 'change flow version' capability eliminates the need to terminate and re-create deployments when you want to deploy a new version of your flow definition.

### Before you begin

- You must have DFFlowAdmin permission.
- There is at least one more version for the same flow definition present in the catalog.
- The state of the flow deployment is either Good Health or Stopped.
- You have read the applicable restrictions and version change strategies.

## About this task



**Important:** As every flow is unique, you need to test version change before performing it in a production environment

## Restrictions

The following restrictions apply to flow version changes:

- Changing inbound connections is not supported.
- Changing custom resource (custom NARs and custom Python resources) configurations is not supported.
- While you can add, change or remove assets when moving to a new version, you cannot introduce assets (text files, binaries, JARs, or similar) if the currently deployed version does not have any.
- Components where state or provenance and other repositories must be kept between flow versions must keep their flow JSON ids. The id changes if you move the component to a different process group or if you delete and then re-add the component to the same process group. NiFi identifies components by these ids. If you move a component to a different process group between versions, its id changes and NiFi perceives it as a new component. This results in the original component being deleted during flow version change together with its state and a new, identical processor being created in a different process group. In an extreme case, you could change to an identical flow version with just the component ids changed and it would result in the deletion of the entire NiFi flow and the recreation of an identical one, with all history and data lost.
- Remapping Parameter Group and Parameter Context assignments is not supported as the original assignment is not removed. For example, you have Process Group 1 (PG1) with Parameter Context 1 (PC1) and Process Group 2 (PG2) with Parameter Context 2 (PC2) assigned. If you initiate a flow version change where parameter contexts are flipped, resulting in a PG1-PC2 and PG2-PC1 assignment, NiFi will not re-map the PG to PC assignments.

## Version change strategies

Depending on the type of your flow, you may select the flow version change strategy most appropriate to you.

### Stop & Process Data

This strategy prioritizes data consistency by stopping source processors and waiting until data is processed before stopping all other components. Once all components have stopped, the flow version is changed and components are started.

Use this strategy when your sources are durable and can handle your source processor being stopped. This generally works well when your source processors are pulling data from sources like Kafka or other messaging queues, databases or file systems.

Should the queued data not be processed within the set time, version change will fail and you can retry the operation with a bigger timeout or you can cancel

### Only Restart Affected Components

This strategy prioritizes uptime by identifying and stopping only components that have changed while keeping all others running, replacing and then starting affected components.

Use this strategy when you want to prioritize uptime of unchanged components or you have made only minor processor configuration changes.

This works well for deployments with inbound connections and will keep your source processors running if they have not changed compared to the previous version.

### Stop & Empty Queues



This strategy forces a version change by stopping all components, emptying all queues, changing flow version, and then starting all components.

Use this strategy only when you want to force a flow version change without keeping any processors running or attempting to process queued data.

All processors will be stopped and all queues will be emptied as part of this strategy.

### Procedure


1. Select the **Deployment** that you want to manage.

2. Click  [Options]  Change Flow Version .

The **Change Flow Version** modal window opens. It shows the list of available flow versions. The current version is grayed out.



**Tip:** You can filter versions by name or by tags.

3. Select the flow version you want to change to and click Continue .
4. Review a summary of the configuration changes caused by the version change and make any necessary edits from the left pane.
5. Select a flow version change strategy.

The available options are:

- Stop & Process Data - If you select this strategy, you can set the maximum wait time for data to be processed and queues to be emptied before the request timed out. The default value is 15 minutes.



**Note:** If you initiate flow version change with this strategy on a stopped flow and there is no data in the queues, flow version change instantly happens regardless of the wait time you configured. However if there is data left in the queues of the stopped flow, the change operation instantly fails.

- Only Restart Affected Components



**Note:** If you initiate flow version change with this strategy and there are queued flow files on any connection that is going to be removed in the new flow version, the version change will fail.

- Stop & Empty Queues - If you select this strategy, you must accept potential data loss by selecting I understand and choose to proceed with the configuration as is.

6. Click  Change Flow Version.

### Results

After you click Deploy, you are redirected to the **Alerts** tab in the **Flow Details** view where you can track how the version change progresses.

## Downloading NiFi application log

You can download the NiFi application log from the CDF Deployment Manager to use it for troubleshooting.


### About this task

This feature allows you to download the NiFi application log that is currently being written. As the log file is rotated and the old file is archived once the file size reaches 10 MB, this is the theoretical maximum you can download using this method. For information on downloading archived log files, see *Diagnostic bundle collection*.

### Before you begin

You need DFFlowAdmin permission to perform this action.

### Procedure

1. Select the **Deployment** that you want to manage.
2. Click  [Options] Download NiFi Log .

The current NiFi application log is downloaded to your computer in tar.gz format.

## Suspending a deployment

Suspending a Cloudera Data Flow deployment terminates cloud resources belonging to a NiFi flow, while maintaining flow persistence.

### About this task


Suspending a Cloudera Data Flow deployment results in the following:

- The NiFi flow stops processing data and all underlying cloud resources are terminated. Any unprocessed data in the flow is stored in memory and its processing resumes when you resume the deployment.
- Flow persistence is maintained while a deployment is suspended.
- You cannot modify deployment configuration while the deployment is suspended.
- Suspended deployments are not billable, resulting in reduced costs.

### Before you begin

You must have deployed a flow definition in Cloudera Data Flow.

### Procedure

1. Select the **Deployment** that you want to manage.
2. Click  [Options] Suspend Deployment .  
The Suspend [Deployment Name] modal opens.
3. Select the Finish data processing option and set a maximum wait time in minutes for data to be processed and queues to be emptied before the request times out.  
This option stops source processors first and waits for queued data to be processed before the flow is suspended. Set a wait time between 5 and 60 minutes using the slider.
4. Click Suspend to suspend the Cloudera Data Flow deployment.

## Resuming a deployment

You can resume a suspended Cloudera Data Flow deployment.


### About this task

Resuming a Cloudera Data Flow deployment reallocates the underlying cloud resources and returns a deployment to the state it was in before being suspended.

### Before you begin

You must have a suspended flow deployment in Cloudera Data Flow.

### Procedure

1. Select the **Deployment** that you want to manage.
2. Click  [Options] Resume Deployment .  
The Resume [Deployment Name] modal opens.
3. Click Resume Deployment to resume the flow deployment, reallocating cloud resources.



## Export deployment configuration

You can export a deployment configuration to create additional deployments with a similar configuration in the same or a different environment.

### About this task

- Exported configurations may be edited, and you can also modify them after the importing step during flow deployment.
- One deployment can have only one exported configuration. Performing a new export overwrites the existing one.
- Exported deployment configurations are available for every user who can start a new deployment in a given environment, even if the exported deployment was originally created under a specific Project.

### Procedure

1. Select the **Deployment** that you want to manage.
2. Click  [Options]  Export Configuration .  
The Export Deployment Configuration modal opens.
3. You can optionally add comments to the exported configuration.
4. Confirm your choice by clicking Export in the modal.

### Results

The configuration is exported to the {LOG location}/cdf-deployment-backup directory. {LOG location} is configured during the creation of the associated Cloudera Environment. If you want to reuse the exported configuration in a different environment, you can either configure that to use the same {LOG location}, or you can copy the exported .tar.gz and JSON files to the {LOG location}/cdf-deployment-backup directory of the target environment.

### What to do next

You can reuse the exported configuration during deployment of the same flow definition to recreate a flow with similar configuration.

## Terminating a deployment

You can terminate a deployment to remove it from Cloudera Data Flow.


### About this task

If you terminate a deployment, you delete the associated NiFi resources and your flow no longer remains active. The associated flow definition remains in the catalog and is available to be deployed again in a new deployment.

### Before you begin


You must have deployed a flow definition in Cloudera Data Flow.

### Procedure

1. Select the **Deployment** that you want to manage.
2. Click  [Options] Terminate Deployment .  
The Terminate [Deployment Name] modal opens.
3. If you select the Delete assigned endpoint hostname option. If you do not select this option, you can reassign existing, unassigned endpoints during flow deployment.

4. Enter the name of the deployment to confirm and click Terminate.



**Tip:** You can also click the  copy to clipboard icon to copy the deployment name, paste the name of the deployment, and then click Terminate.

## Flow drafts

On the Flow drafts tab you can manage the life cycle of flow drafts and associated test sessions.


### Opening a draft for editing

You can open a selected flow draft for editing in the Flow Designer.

#### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera Data Flow Authorization.

#### Procedure

1. Select the **Flow draft** that you want to manage.
2. Click  [Options] Open .  
The flow draft opens in the Flow Designer.




### Starting, stopping, and restarting a test session

Learn how you can start, stop, or restart a test session for a flow draft.

#### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera Data Flow Authorization.

#### Procedure

1. Select the **Flow draft** that you want to manage.
2. Select one of the following options
  - To start a test session, click  [Options] Start Test Session . For more information on configuring a test session, see [Test sessions](#).
  -  **Note:** You cannot start a test session if there is already an **Active** or **Failed** test session.
  - To stop a test session, click Stop Test Session, then click End to confirm your choice.
  - To restart an **Inactive** test session, click  [Options] Restart Test Session


### Publishing a flow draft

Learn about publishing a draft to the Catalog as a flow definition.

#### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera Data Flow Authorization.

### Procedure

1. Select the **Flow draft** that you want to manage.
2. Click  [Options] Publish .
3. Provide a Flow Name and optional Flow Description, Custom Tag, and Version Description.
4. Click Publish.


## Reassigning a flow draft to a different project

Learn how to reassign a flow draft to another project.

### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera Data Flow Authorization.

### Procedure

1. Select the **Flow draft** that you want to reassign.
2. Click  Reassign.  
The **Reassign Resource** modal opens.
3. Select a Project and click Reassign.



## Deleting a flow draft

Learn about deleting flow drafts.

### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera Data Flow Authorization.

### Procedure

1. Select the **Flow draft** that you want to delete.
2. Click  [Options]  Delete .
3. Click Delete to confirm your choice.

## Parameter groups

You can manage shared parameter groups in the Resources view of Cloudera Data Flow.

## Procedure

Parameter Group

Resources / Workspace Resources / Parameter Group

data\_transform

WORKSPACE: cdf-priv-azure PROJECT: cdev

Parameters Used By Details

Parameters

Name ↑	Value
replace with value	2
value to replace	1

REFRESHED: 19 seconds ago

⊕ Add Parameter

- To view the list of flow deployments and flow drafts using this parameter group, select the **Used By** tab. It is important to be aware of the flow deployments and flow drafts that are going to be affected by any change you make to a parameter group.
- To view or modify the Parameter Group Name and Description, select the **Details** tab.

## Modifying parameters in a parameter group

### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera Data Flow Authorization.

### Procedure

1. Select the **Parameter Group** where you want to modify parameters.
2. Click [Options] Open .
3. Click on the parameter card of the parameter you want to modify.  
In the parameter details pane modify the parameter Value and Description as necessary.
4. Click Apply Changes.

## Deleting a parameter group

### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera Data Flow Authorization.

### Procedure

1. Select the **Parameter Group** that you want to delete.
2. Click [Options] Open .

3. Click on the parameter card that you want to delete.

In the parameter details pane click  Delete to delete the parameter from the group.



If the parameter group is not used by any flow draft, the **Delete [parameter group name]** modal opens. If the parameter group is in use by a flow draft, it cannot be deleted.

4. Click Delete to confirm your choice.

5. Click  Apply Changes.

## Importing a parameter group to a flow draft

### Procedure

1. Select the **Parameter Group** that you want to import to a flow draft.
2. Click  [Options]  Import to Flow Draft .
3. In the **Import to Flow Draft** modal select the Flow Draft where you want to import the parameter group.
4. Click Import.

## Duplicating a parameter group



Learn how to duplicate parameter groups. Duplicating a parameter group allows you to reassign the newly created group to another project, or to customize it according to your needs without touching the original group.

### About this task



**Note:** Duplicating a parameter group does not duplicate any assets (file-type parameters) associated with it. You need to reupload those to the newly created group.

### Procedure

1. Select the **Parameter Group** that you want to import to duplicate.
2. Click  [Options]  Duplicate .
3. In the **Duplicate Parameter Group** modal provide a Parameter Group Name and an optional Description.
4. Click Duplicate.

## Reassigning a parameter group to a different project

Learn how to reassign a parameter group to another project.

### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera Data Flow Authorization.

### About this task

You cannot reassign a parameter group that is used by flow drafts. You have to remove the group from all flow drafts before you can reassign it to a different project.



**Note:** When you reassign a parameter group to another project, assets associated with it are not moved with it. You need to reupload those to the new project.

### Procedure

1. Select the **Parameter Group** that you want to reassign.



**Note:** You can reassign multiple items simultaneously by selecting the checkboxes in front of them.

2. Click Reassign.

If the parameter group is not used by any flow draft, the **Reassign Resource** modal opens.

3. Select a Project and click Reassign.

4. Click  Apply Changes.

## Inbound connections

On the Inbound Connections tab you can reassign and renew the certificates of inbound connections.

### Renewing the certificate for an inbound connection endpoint

If you need to replace an X.509 certificate for an inbound connection endpoint before it expires, you can do so manually.

#### Before you begin

You need DFFlowAdmin privilege to perform this action.

### Procedure

1. Select the **Inbound Connection** that you want to manage.
2. Click Renew.

- To renew the server certificate, select NiFi Inbound SSL Context Service.



**Note:** Each server certificate is limited to five renewals in a 7 day sliding window.

- To renew the client certificate, select Client SSL Context.
- If you leave Revoke previously issued client certificates unchecked, existing client certificates remain valid and existing clients can continue to connect to your deployment using it. By selecting the Revoke previously issued client certificates option, you invalidate all existing certificates and you will need to add the new certificate to existing clients so that they can keep connecting to your Cloudera Data Flow deployment.

3. Click Renew & Restart.

The UI switches to the **KPIs and Alerts** pane where you can monitor as your deployment restarts and the new certificate or certificates become available.

#### What to do next

##### If you have renewed the NiFi Inbound SSL Context Service:

You have to take no further action.

##### If you have renewed the Client SSL Context:

After your Cloudera Data Flow deployment has restarted, you switch to the NiFi Configuration pane to download the Client Certificate and the Client Private Key. You can then add these to your client.

## Reassigning an inbound connection endpoint to a different project

Learn how to reassign an inbound connection to another project.



### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera Data Flow Authorization.

### About this task

You cannot reassign an inbound connection that is currently used by a deployment. You have to terminate the deployment using it making sure that the Delete assigned endpoint hostname option is not selected before you can reassign it to a different project.

### Procedure

1. Select the **Inbound Connection** that you want to reassign.



**Note:** You can reassign multiple items simultaneously by selecting the checkboxes in front of them.

2. Click Reassign.

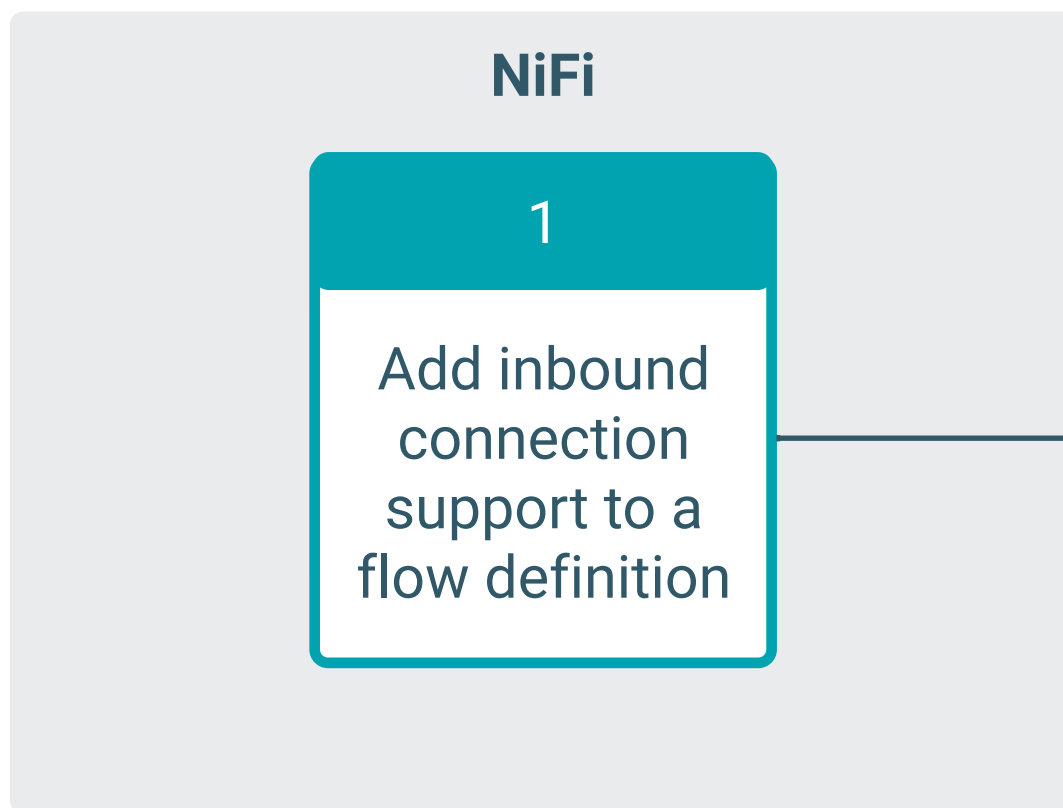
If the inbound connection is not used by any deployment, the **Reassign Resource** modal opens.

3. Select a Project and click Reassign.

4. Click  Apply Changes.

## Create an Inbound Connection Endpoint during flow deployment

During flow deployment, you can enable an endpoint that allows NiFi to listen to external data



sources.

### About this task

You can create flow deployments with an Inbound Connection Endpoint during flow deployment, using the Cloudera Data Flow Deployment wizard.



**Note:** Currently you cannot create flow deployments with Inbound Connection Endpoints using the CLI.

### Before you begin

- A flow definition supporting inbound connections has been created and added to the Catalog.
- You have DFCatalogViewer role to view flow definitions in the Catalog.
- You have DFFlowAdmin right for the environment where you want to create the endpoint.

### Procedure

1. During flow deployment, at the NiFi Configuration step, select **Inbound Connections Allow NiFi to Receive Data**.

An Endpoint Hostname, generated from the Deployment Name you provided at the Overview step is offered and port configuration options become available.

dataflow-azure / New Flow Deployment

NiFi Configuration

NiFi Runtime Version [Change Version](#)

CURRENT VERSION  
Latest Version (1.16.0.2.3.4.0-33)

Autostart Behavior

☒ Automatically start flow upon successful deployment

Inbound Connections

☒ Allow NiFi to receive data [?](#)

Endpoint Hostname [Select Preexisting Endpoint](#)

listenhttpflow .inbound.dfx.dtefgqis.xcu2-8y8x.dev.cldr.work

[?](#) This endpoint is valid

Listening Ports [?](#) [Edit Ports](#)

Protocol <a href="#">?</a>	Port
TCP <a href="#">?</a>	7001

Custom NAR Configuration

☐ This flow deployment uses custom NARs [?](#)

Overview

FLOW DEFINITION  
ListenHTTP filter to Kafka v.1

ENVIRONMENT DEPLOYING TO  
dataflow-azure

DEPLOYMENT NAME  
ListenHTTP Flow

[Cancel](#) [← Previous](#) [Next →](#)

2. You can accept the generated hostname as is, or you can change the prefix. You can also click **Select Preexisting Endpoint** to select an available endpoint within your environment.

Endpoints are unavailable if they are currently used by another flow deployment.

### 3. Select Listening Ports.

Provide the protocol and port or ports where your flow will listen to incoming data. Cloudera Data Flow automatically suggests ports based on listen processors detected in the flow definition. Custom Processors, if applicable, are not detected and must be added manually.

Using ports with mixed protocols (TCP and UDP) is not allowed. As long as the suggested ports use mixed protocols, you are not allowed to add new ports or to proceed to the next step in the Wizard.



**Note:** If you want your dataflow to use HTTP, select TCP protocol.

✕

**Protocol**  

TCP
▼

**Port** ?  

Specify a port
⊕ Add Port

Protocol	Port
TCP <span style="color: #0070C0; font-size: 0.8em;">?</span>	7001 <span style="float: right; color: #0070C0; font-size: 0.8em;">⊖</span>

Cancel

OK

### 4. Specify Trusted IP Addresses.

Specify a comma-separated list of trusted CIDRs or IP ranges. To allow all traffic, select Allow all traffic.



**Important:** Selecting Allow all traffic keeps your inbound connection open to all incoming traffic. This can expose your connection to potential security risks.

### Results

As part of the flow deployment, a Layer-4 Load Balancer (LB) is created to forward traffic to the flow deployment's NiFi cluster on the configured ports and protocols. DNS records are created for the Inbound Connection Endpoint hostname to resolve to the Layer-4 LB.



**Note:** There is an initial delay for the cloud provider (AWS or Azure) to fully provision and start the LB resources, during which the flow deployment appears in the Cloudera Data Flow monitoring dashboard as healthy, but it is not yet able to receive data from clients. The load balancer provisioning normally completes within 10 minutes of the flow deployment being finished, but can take up to three hours.

### What to do next

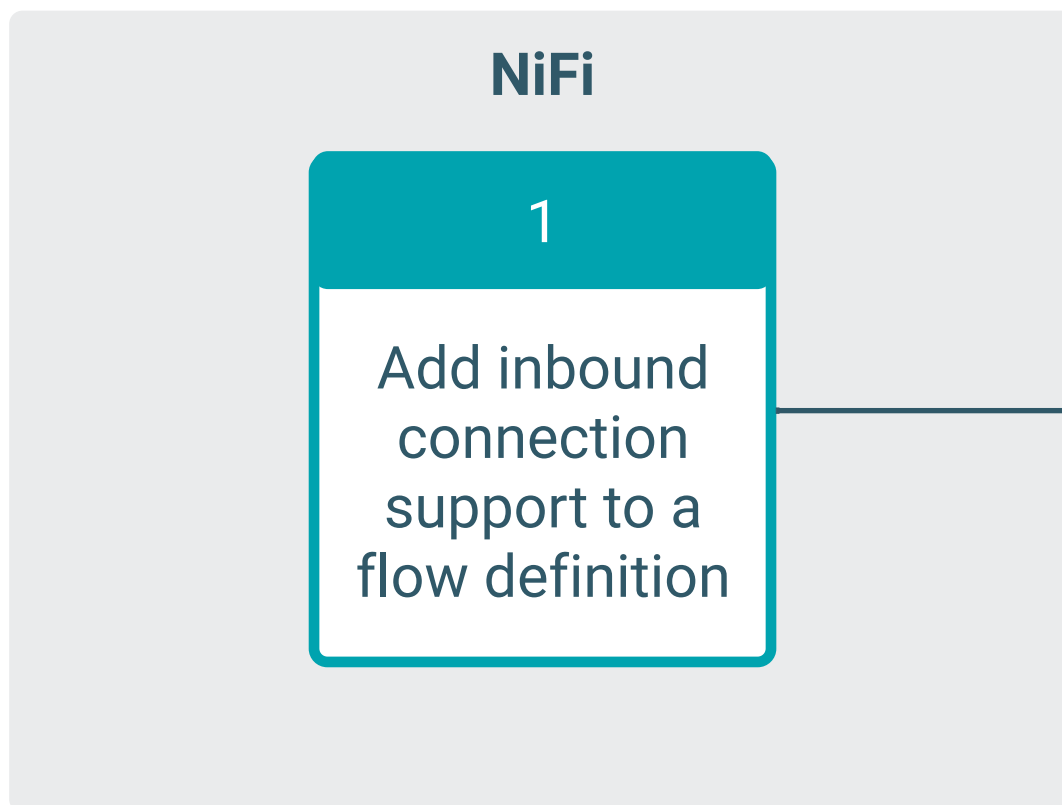
If the listening port in your flow definition is parameterized instead of having a pre-defined value, you have to fill in the same listening port numbers you provided in the NiFi Configuration step. The listening ports you specified in the NiFi Configuration step are visible in the Overview pane.

### Related Information

[Deploying a flow definition using the wizard](#)

## Connecting applications to an endpoint

Once a Cloudera Data Flow flow deployment with inbound connection is available, you can go on and connect an external application to start sending data.



### Before you begin

- A flow deployment with inbound connection is available.
- A network connection through which the client can reach the flow deployment endpoint is available.
- You have been assigned at least the DFFlowUser role for the environment to which you want to configure the inbound connection.

### Procedure

1. Select the flow deployment where you want to send data and go to **Deployment Manager Deployment Settings**.
2. Select the **NiFi Configuration** tab.
3. Make a note of the **Endpoint Hostname** and **port**.
4. Click **Download Client Certificate**.

The X.509 client certificate downloads to your computer in PEM format.

5. Click **Download Client Private Key** to obtain the RSA Private Key.

The unencrypted RSA Private Key encoded with PKCS8 downloads to your computer in PEM format.

6. Depending on your client, you may have to convert the certificate and the private key to a different format.

For example, to convert PEM to PKCS12 format, use the following command:

```
openssl pkcs12 -export -in [***DOWNLOADED PEM CERT FILE***] -in  
key [***DOWNLOADED PEM PRIVATE KEY***] -out certificate.p12
```

To further convert the PKCS12 file to JKS format for a Java client, run the following command:

```
keytool -importkeystore -srckeystore [***CERTIFICATE NAME***].p12 -srcsto  
retype pkcs12 -destkeystore [***DESTINATION KEYSTORE***].jks
```

7. Add the certificate file and the private key files to the keystore of your application.
8. Configure your application to stream data to the Endpoint Hostname, port, and protocol of the flow deployment.

## TLS keys and certificates

When using Inbound Connection Endpoints, sensitive information is sent over the network between Cloudera Data Flow (CDF) and external data sources including configuration files that contain passwords. To secure this transfer, Cloudera strongly recommends that you configure mutual Transport Layer Security (TLS) encryption.

TLS is an industry standard set of cryptographic protocols for securing communications over a network.

Configuring TLS involves creating a private key and a public key for use by server and client processes to negotiate an encrypted connection. In addition, TLS can use certificates to verify the trustworthiness of keys presented during the negotiation to prevent spoofing and mitigate other potential security issues.

## Inbound connections

An Inbound Connection Endpoint allows you to stream data from an external source application to a flow deployment.

An Inbound Connection Endpoint provides a stable hostname that can be used to send data to a Cloudera DataFlow flow deployment located in the same environment. You can create an Inbound Connection Endpoint during flow deployment, provided that the flow definition supports creating such endpoints.

An endpoint has the following attributes:

- The environment in which it exists
- A stable hostname
- An SSLContextService configured that can work with the TLS context auto created during flow deployment that corresponds to the Inbound Connection Endpoint hostname
- A server certificate that corresponds to the hostname, signed by a public root CA
- A single Intermediate CA (issued by the Environment CA) for signing client certificates for this endpoint, with the public certificate part available to download for client truststores

Endpoints exist within the environment where they were created. They cannot be moved between environments. If the environment is deleted, the endpoint gets deleted as well, and cannot be reused.

One endpoint can be assigned to one flow deployment at a time. To reassign an existing inbound connection endpoint, you need to first terminate the flow deployment to which it is currently assigned, then assign the existing endpoint to a new flow deployment in the NiFi configuration step during flow deployment.

## Workflow

Setting up an Inbound Connection Endpoint is a complex task, affecting how you develop a flow definition in NiFi and how you deploy it in Cloudera DataFlow. Once your flow has been deployed, you need to configure your client, be it directly an external application or through an external load balancer, to communicate with the Inbound Connection Endpoint of your flow deployment.

## Tutorial: MiNiFi to Cloudera Data Flow flow deployment

This tutorial walks you through creating an inbound connection endpoint in Cloudera Data Flow used by a flow deployment to receive data from one or more MiNiFi agents managed by Edge Flow Manager.

### Before you begin

- You have an enabled and healthy Cloudera Data Flow environment.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Cloudera Data Flow Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.
- You have an enabled and healthy Cloudera Edge Management (CEM) environment.
- You have sufficient rights to configure the MiNiFi Agent in EFM.

### Procedure

1. In a development NiFi environment, create a Controller Service of type StandardRestrictedSSLContextService at the root canvas level and name it Inbound SSL Context Service.
  - a. In the Operate palette click Configuration Controller Services Create a new controller service
  - b. Filter for ssl, select StandardRestrictedSSLContextService then click Add.
  - c. Click Configure.
  - d. On the **Settings** tab change the Name to Inbound SSL Context Service, then click Apply.

You do not need to make further configuration on this Controller Service; it acts as a placeholder and will be created with a managed SSL Context when deployed by Cloudera Data Flow.



#### Tip:

For testing the dataflow during development, the SSL Context may be configured with test keys and certificates for your development NiFi environment.

2. Create a Process Group on the root canvas to hold your flow definition and give it a name.  
This tutorial uses the name ListenHTTP Flow.
3. Enter the process group.
4. Inside the Process Group, add a listen processor.  
This tutorial uses ListenHTTP.

## 5. Configure the listen processor:

### Base Path

This tutorial uses the default contentListener.

### Listening Port

Define a value that is valid for your use case. This tutorial uses port 9000.

### SSL Context Service

Select Inbound SSL Context Service.

### Client Authentication

Select REQUIRED.

Click Apply.

**Configure Processor** | ListenHTTP 1.15.3

Invalid

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

Required field

✓

+

Property	Value
Base Path	contentListener
Listening Port	9000
Listening Port for Health Check Requests	No value set
Max Data to Receive per Second	No value set
SSL Context Service	Inbound SSL Context Service →
Client Authentication	REQUIRED
Authorized Subject DN Pattern	.*
Authorized Issuer DN Pattern	.*
Max Unconfirmed Flowfile Time	60 secs
HTTP Headers to receive as Attributes (Regex)	No value set
Return Code	200
Multipart Request Max Size	1 MB

CANCEL

APPLY

## 6. Connect the ListenHTTP processor to a downstream processor of your choice.

This tutorial uses LogAttribute, where all relationships terminate.

## 7. From the root canvas, right click on the Process Group and select Download flow definition Without external controller services .

## 8. Upload the flow definition JSON to the Flow Catalog of your Cloudera Data Flow deployment.

**9. Deploy the flow.**

- a) At the NiFi Configuration step of the Deployment wizard, select **Inbound Connections Allow NiFi to Receive Data** to enable inbound connections.

Accept the automatically created endpoint hostname and automatically discovered port by clicking **Next**.

- b) At **Parameters**, click **Next**.

- c) At **Sizing & Scaling** select the **Extra Small NiFi Node Size** then click **Next**.

- d) Add a KPI on the **ListenHTTP** processor to monitor how many bytes it is receiving, by clicking **Add new KPI**.

Make the following settings:

**KPI Scope**

Processor

**Processor Name**

ListenHTTP

**Metric to Track**

Bytes Received



Add new KPI
✕

Details

KPI Scope ?
Processor Name ?

Processor ▼

ListenHTTP ▼

Metric to Track ?

Bytes Received ▼

METRIC DESCRIPTION:  
Number of bytes received from a source that is external from the flows

Alerts

☐ Trigger alert when metric is greater than

Value

MBytes ▼

☐ Trigger alert when metric is less than

Value

MBytes ▼

Alert will be triggered when metric is outside the boundary(s) for

Value

Minutes ▼

Cancel
Add

e) Review the information provided and click Deploy.

Soon after the flow deployment has started, the client certificate and private key required for sending data to the NiFi flow become available for the flow deployment that is being created.

10. Collect the information required to configure your load balancer.

- a. Once the deployment has been created successfully, select it in the Deployments view and click Manage Deployment.
- b. In the Deployment Settings section, navigate to the **NiFi Configuration** tab to find information about the associated inbound connection endpoint.
- c. Copy the endpoint hostname and port and download the certificate and private key.

**11. Start designing your MiNiFi flow in EFM.****For C++**

To design a flow for your MiNiFi C++ agent class:

- a. Copy the downloaded client-private-key-encoded key and client-certificate-encoded.cer certificate files to the host with the running MiNiFi C++ agent, so they are accessible by filepath from the agent.
- b. Create a Service of type SSL Context Service with the following configuration:

**Service Name**

Specify a name for this service. This tutorial uses Client SSL Context Service.

**CA Certificate**

Leave it empty. As Cloudera Data Flow uses Let's Encrypt as a Certificate Authority, the certificate will be accepted automatically, without additional configuration.

**Client Certificate**

[\*\*\*/PATH/TO/\*\*/]client-certificate-encoded.cer

For example, /opt/minifi/minifi-test/client-certs/client-certificate-encoded.cer.

**Passphrase**

Set no value.

**Private Key**

[\*\*\*/PATH/TO/\*\*/]client-private-key-encoded

For example, /opt/minifi/minifi-test/client-certs/client-private-key-encoded

**Use System Cert Store**

Keep the default False value.

»

SSLContextService (Service)

## Configuration

### Settings

Service Name\*

Client SSL Context Service

### Properties

Property	Value
CA Certificate	<a href="#">?</a> No value set <a href="#">↑</a>
Client Certificate	<a href="#">?</a> /opt/minifi/minifi-test/client-certs/client-certificate-encoded... <a href="#">↑</a>
Passphrase	<a href="#">?</a> No value set <a href="#">↑</a>
Private Key	<a href="#">?</a> /opt/minifi/minifi-test/client-certs/client-private-key-encoded <a href="#">↑</a>
Use System Cert Store	<a href="#">?</a> false <a href="#">↑</a>

### About

SERVICE ID  
8d7cb8fd-af49-480c-8399-d0845fbb95de

SERVICE TYPE  
SSLContextService 1.22.04

BUNDLE  
org.apache.nifi.minifi - minifi-system

### Comments

Describe the changes made in this update

Apply

c. Click Apply.

d. Create an InvokeHTTP processor named Send to CDF with the following configuration:

#### Automatically Terminated Relationships

Select all relationships.

#### Content-type

Depends on your flow file data type. This tutorial uses text/plain.

#### HTTP Method

POST

#### Remote URL

https://[\*\*\*ENDPOINT HOSTNAME COPIED FROM CLOUDERADATAFLOW FLOW DEPLOYMENT MANAGER\*\*\*]:9000/contentListener

For example, https://my-flow.inbound.my-dfx.c94x5i9m.xcu2-8y8z.mycompany.test:9000/contentListener

## SSL Context Service

### Client SSL Context Service

Leave all other settings with their default values.

The screenshot displays the Cloudera Data Flow interface. On the left, the 'Design / Agent Class' pane shows a flow for the 'minifi-cpp-latest' agent class. The flow starts with a 'GenerateFlowFile' processor, followed by a 'success' label, and then a 'Send to CDF (Processor)' service, which is highlighted with a blue circle. On the right, the 'Configuration' pane for the 'Send to CDF (Processor)' service is shown. It includes sections for 'Automatically Terminated Relationships' (with checkboxes for failure, no retry, response, retry, and success), 'Scheduling' (with a 'Timer Driven' strategy and 'Concurrent Tasks' set to 1), 'Run Schedule\*' (set to 1000 ms), and 'Run Duration\*' (a slider from 0ms to 2s). Below these is a 'Properties' table with various settings.

Property	Value
Always Output Response	<input type="radio"/> false
Attributes to Send	<input type="radio"/> No value set
Connection Timeout	<input type="radio"/> 5 s
Content-type	<input type="radio"/> application/plain-text
Disable Peer Verification	<input type="radio"/> false
Follow Redirects	<input type="radio"/> true
HTTP Method	<input type="radio"/> POST

An 'Apply' button is located at the bottom right of the configuration pane.

### For Java

To design a flow for your MiNiFi Java agent class:

- a. Convert the downloaded client-private-key-encoded key and client-certificate-encoded.cer certificate files to a JKS Keystore:
  1. Create a PKCS12 keystore:
 

```
openssl pkcs12 -export -in client-certificate-encoded -inkey client-private-key-encoded -out client-keystore.p12
```
  2. Convert the PKCS12 keystore to a JKS keystore:
 

```
keytool -importkeystore -srckeystore client-keystore.p12 -srcstoretype pkcs12 -destkeystore client-keystore.jks
```
- b. Copy the resulting client-keystore.jks file to the host with the running MiNiFi Java agent, so they are accessible by filepath from the agent.
- c. Obtain the CA root cert and add it to truststore client-truststore.jks, by running the following commands:

```
wget https://letsencrypt.org/certs/isrgrootx1.pem
```

```
keytool -import -file isrgrootx1.pem -alias isrgrootx1 -keystore client-truststore.jks
```

MiNiFi Java requires you to specify an explicit truststore for inbound connections. Remember the password you used for creating client-truststore.jks, as you will need it .

- d. Create a Service of type Restricted SSL Context Service with the following configuration:

#### Service Name

Specify a name for this service. This tutorial uses Client SSL Context Service.

#### Keystore Filename

[\*\*\*/PATH/TO/\*\*/]client-truststore.jks

#### Keystore Password

[\*\*\*THE PASSWORD YOU PROVIDED WHEN CREATING THE JKS STORE\*\*\*]

#### Key Password

[\*\*\*THE PASSWORD YOU PROVIDED WHEN CREATING THE JKS STORE\*\*\*]

#### Keystore Type

JKS

#### Truststore Filename

client-truststore.jks

#### Truststore Type

JKS

#### Truststore Password

[\*\*\*THE PASSWORD YOU PROVIDED WHEN CREATING THE CLIENT TRUSTSTORE\*\*\*]

The screenshot displays the Cloudera Edge Management interface. On the left is a dark sidebar with navigation options: Monitor, Edge Events, Design (highlighted), and Agent Manager. The main area is titled 'Design / Agent Class' and contains a 'Services' section with a table listing 'Client SSL Context Service'. To the right, the 'Configuration' page for this service is shown, featuring a 'Settings' section with a text input for 'Service Name\*' containing 'Client SSL Context Service'. Below this is a 'Properties' table with various configuration parameters and their values.

Property	Value
Keystore Filename	/opt/minifi/minifi-current/client-cert...
Keystore Password	#{Keystore Password}
Key Password	#{Key Password}
Keystore Type	JKS
Truststore Filename	Empty string set
Truststore Password	No value set
Truststore Type	No value set
TLS Protocol	TLS

Below the properties table is an 'About' section with the following details:

- SERVICE ID:** f19bd51a-db0e-47ad-916e-542d4482590b
- SERVICE TYPE:** StandardRestrictedSSLContextService 1.3.1.0
- BUNDLE:** org.apache.nifi.minifi - minifi-ssl-context-service-nar

An 'Apply' button is located at the bottom right of the configuration panel.

- e. Click Apply.
- f. Create an InvokeHTTP processor named Send to CDF with the following configuration:  
**Automatically Terminated Relationships**  
 Select all relationships.

### Content-type

Depends on your flow file data type. This tutorial uses text/plain.

### HTTP Method

POST

### Remote URL

`https://[***ENDPOINT HOSTNAME COPIED FROM CLOUDERA DATAFLOW FLOW DEPLOYMENT MANAGER***]:9000/contentListener`

For example, `https://my-flow.inbound.my-dfx.c94x5i9m.xcu2-8y8z.mycompany.test:9000/contentListener`

### SSL Context Service

Client SSL Context Service

Leave all other settings with their default values.

The screenshot shows the Cloudera Edge Management (CEM) Design / Agent Class interface. On the left, a sidebar contains navigation options: Monitor, Edge Events, Design (selected), and Agent Manager. The main workspace displays a flow diagram for the 'cdf-client-java' agent class. A 'GenerateFlowFile' processor is connected to a 'Send to CDF' processor via a relationship named 'GenerateFlowFile/success/InvokeHTTP'. The 'Send to CDF' processor is highlighted with a blue circle. On the right, the configuration panel for the 'Send to CDF (Processor)' is shown. The configuration includes sections for Configuration, Automatically Terminated Relationships, Scheduling, and Properties.

**Configuration**

Processor Name\*  
Send to CDF

Penalty Duration\* 0 ms Yield Duration\* 0 ms

Automatically Terminated Relationships  
☒ Original ☒ Failure ☒ Retry ☒ No Retry ☒ Response

**Scheduling**

Scheduling Strategy\* Timer Driven Concurrent Tasks\* 1

Run Schedule\* 0 ms

Run Duration\* 0ms 25ms 50ms 100ms 250ms 500ms 1s 2s

**Properties**

Property	Value
HTTP Method	POST
Remote URL	https://kevindemo.inbound.dfx.p8jdxchd.xcu2-8y...
SSL Context Service	Client SSL Context Service
Connection Timeout	5 secs
Read Timeout	15 secs

Apply

12. Build the rest of your data flow to read data and send to your Cloudera Data Flow flow deployment using InvokeHTTP. As a simple example, this tutorial uses the GenerateFlowFile processor, with the following settings:

Run Schedule

Set to 10000 ms (10 seconds).

Custom Text

The message you type here will be sent to the ListenHTTP Flow you have created, with the frequency specified by Run Schedule. For example, Hello DFX! This is MiNiFi.

Data Format

Set to Text.

Unique FlowFiles

Set to false.

Design / Agent Class

PROCESSOR

REMOTE PROCESS GROUP

FUNNEL

minifi-cpp-latest

GenerateFlowFile  
GenerateFlowFile

NAME: success

Send to CDF  
InvokeHTTP

SERVICESPARAMETERS

»

GenerateFlowFile (Processor)

Configuration

Penalty Duration\*

30000 ms

Yield Duration\*

1000 ms

Automatically Terminated Relationships

☐ SUCCESS

Scheduling

Scheduling Strategy\*

Timer Driven

Concurrent Tasks\*

1

Run Schedule\*

10000 ms

Run Duration\*

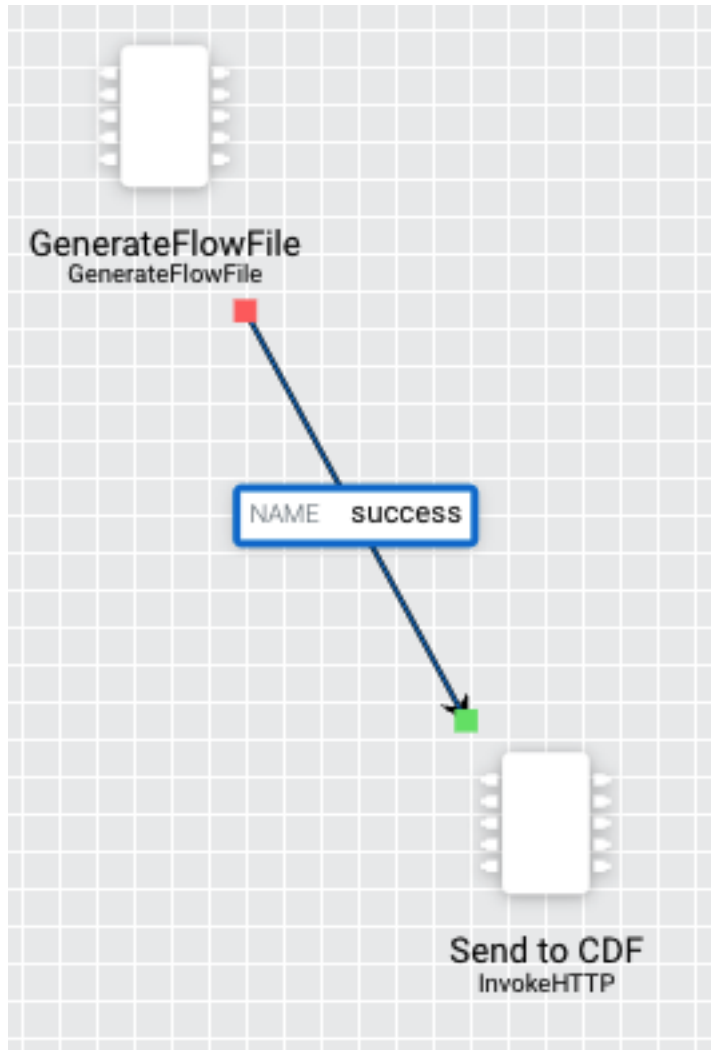
0ms25ms50ms100ms250ms500ms1s2s

Properties

Property	Value
Batch Size	<div>1</div>
Custom Text	<div>Hello DFX! This is MINIFI.</div>
Data Format	<div>Text</div>
File Size	<div>1kB</div>
Unique FlowFiles	<div>false</div>

Apply

13. Connect the GenerateFlowFile processor to the InvokeHTTP processor.



14. Click Actions Publish... to publish the flow and start it on your MiNiFi agent.

15. Select your flow deployment in the Cloudera Data Flow Dashboard and click KPIs.

Observe that your Cloudera Data Flow flow deployment is now receiving data from MiNiFi.

### Related Tasks

[Connecting applications to an endpoint](#)

### Related Information

[Creating a flow definition in NiFi](#)

[Deploying a flow definition](#)

[Building a dataflow in CEM](#)

[Publishing a dataflow in CEM](#)

## Tutorial: Invoking an HTTP endpoint with curl

This tutorial walks you through invoking an HTTP Inbound Connection Endpoint with curl using the ListenHTTP filter to Kafka ReadyFlow from the ReadyFlow Gallery.

### Before you begin

- You have an enabled and healthy Cloudera Data Flow environment.



- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Cloudera Data Flow Catalog.
- You have added the ListenHTTP filter to Kafka ReadyFlow to the Catalog. Adding a ReadyFlow to the Catalog requires the DFCatalogAdmin role.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.
- You have network connectivity established from your curl client to the virtual network or VPC where your Flow Deployment will run.
- You have downloaded and installed curl.

## Procedure

1. Deploy the ListenHTTP filter to Kafka Ready Flow.
  - a. Navigate to the ReadyFlow Gallery, locate the ListenHTTP filter to Kafka ReadyFlow and click View Added Flow Definition.
  - b. Click Deploy and select your target environment to start the Deployment Wizard for the latest version of this ReadyFlow.
  - c. Specify a deployment name, for example, Inbound Connections curl and click Next.
  - d. Select Allow NiFi to receive data checkbox to configure an endpoint host.
  - e. Accept the automatically created endpoint hostname and automatically discovered port by clicking Next.
  - f. Optional: This ReadyFlow performs schema validation for incoming events using Cloudera's Schema Registry before sending the events to a Kafka topic. If you have a Streams Messaging cluster available, fill in the Kafka and Schema Registry connection properties.

If you only want to validate inbound connection endpoint connectivity, enter dummy values for the empty parameters, set the Input and Output format to JSON while keeping the Listening Port set to 7001.

- g. At Sizing & Scaling select the Extra Small NiFi Node Size and click Next.
- h. Add a KPI on the ListenHTTP processor to monitor how many bytes it is receiving, by clicking Add new KPI.

Make the following settings:

**KPI Scope**

Processor

**Processor Name**

ListenHTTP

**Metric to Track**

Bytes Received

Add new KPI
✕

Details

KPI Scope ?
Processor Name ?

Processor ▼

ListenHTTP ▼

Metric to Track ?

Bytes Received ▼

METRIC DESCRIPTION:  
Number of bytes received from a source that is external from the flows

Alerts

☐ Trigger alert when metric is greater than

Value

MBytes ▼

☐ Trigger alert when metric is less than

Value

MBytes ▼

Alert will be triggered when metric is outside the boundary(s) for

Value

Minutes ▼

Cancel
Add

i. Click Next.

j. Review the information provided and click Deploy.

Soon after the flow deployment has started, the client certificate and private key required for sending data to the NiFi flow become available for the flow deployment that is being created.

2. Collect the information required to configure your load balancer.

- a. Once the deployment has been created successfully, select it in the Deployments view and click Manage Deployment.
- b. In the Deployment Settings section, navigate to the **NiFi Configuration** tab to find information about the associated inbound connection endpoint.
- c. Copy the endpoint hostname and port and download the certificate and private key.

### 3. Create the curl request to validate connectivity to the HTTP inbound connection endpoint.

Using the endpoint hostname, port, client certificate and private key you can now construct a curl command to call the endpoint and validate connectivity:

```
curl -v -X POST https://[***ENDPOINT_HOSTNAME***]:7001/contentListener --key [***PATH/TO/
***]client-private-key-encoded --cert [***PATH/TO/***]client-certificate-encoded
```

You receive an HTTP 200 response code in a similar message, indicating that your client was able to securely connect to the inbound connection endpoint:

```
* Trying 10.36.84.149:7001...
* Connected to [***ENDPOINT_HOSTNAME***] (10.36.84.149) port 7001 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
* CAfile: /Users/mkohs/letsencrypt-stg-root-x1.pem
* CAPath: none
* (304) (OUT), TLS handshake, Client hello (1):
* (304) (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Request CERT (13):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS handshake, CERT verify (15):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
* subject: CN=dfx.dtefgqis.xcu2-8y8x.dev.cldr.work
* start date: May 11 21:02:20 2022 GMT
* expire date: Aug 9 21:02:19 2022 GMT
* subjectAltName: host "[***ENDPOINT_HOSTNAME***]" matched cert's
  "[***ENDPOINT_HOSTNAME***]"
* issuer: C=US; O=(STAGING) Let's Encrypt; CN=(STAGING) Artificial Apricot R3
* SSL certificate verify ok.
> POST /contentListener HTTP/1.1
> Host: [***ENDPOINT_HOSTNAME***]:7001
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 12 May 2022 00:45:57 GMT
< Content-Type: text/plain
< Content-Length: 0
< Server: Jetty(9.4.46.v20220331)
<
* Connection #0 to host [***ENDPOINT_HOSTNAME***] left intact
```

### 4. Use curl to post data to the HTTP inbound connection endpoint.

If you want to post events to the NiFi deployment you can add header and content definition to the curl request.

This example sends JSON data following a simple schema to the endpoint:

```
curl -v -X POST [***ENDPOINT_HOSTNAME***]:7001/contentListener \
--key [***PATH/TO/***]client-private-key-encoded \
--cert [***PATH/TO/***]client-certificate-encoded \
```

```
-H 'Content-Type: application/json' \  
-d '{"created_at":6453,"id":6453,"text":"This is my test event","timestamp_ms":34534,"id_store":12}'
```

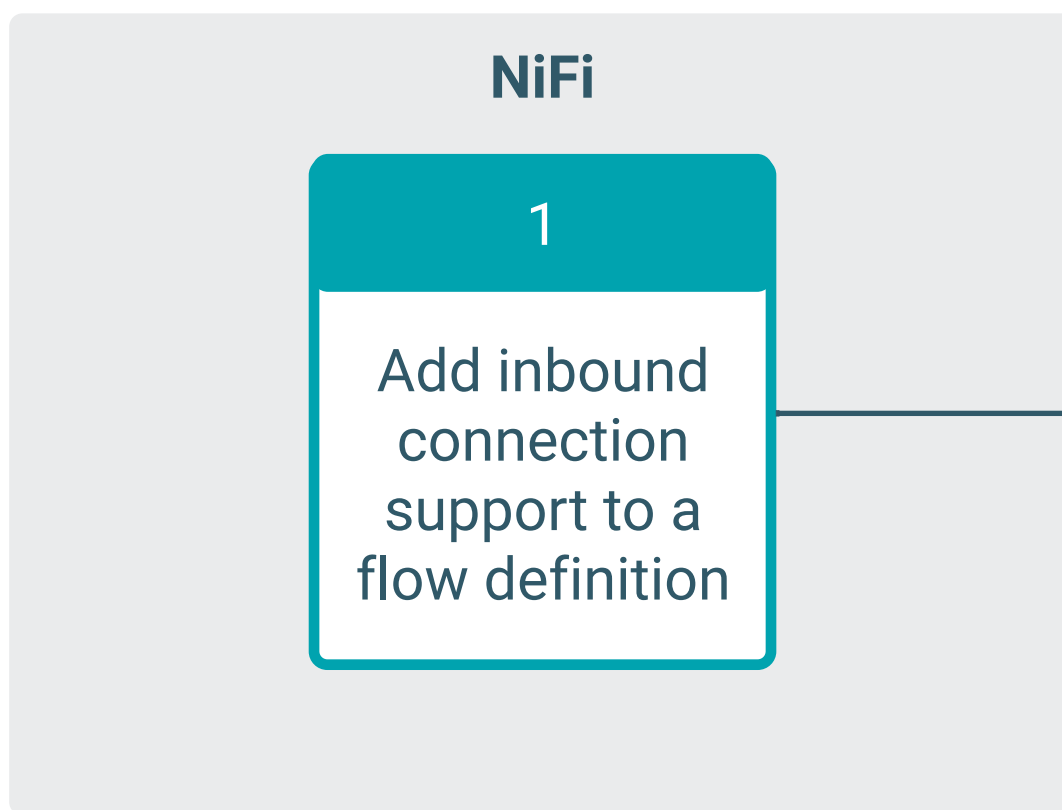
The NiFi deployment tries to validate the schema against the schema name and Schema Registry that you provided when deploying the ReadyFlow. If you provided dummy values, you receive a response indicating that the flow was unable to look up the schema.

### Related Information

[ReadyFlow overview: ListenHTTP to CDP Kafka](#)

## Using Inbound Connections with an external load balancer

Once a Cloudera Data Flow deployment with an Inbound Connection Endpoint is available, you can go on and connect an external load balancer to start sending data.



Inbound Connection Endpoints are created in Cloudera Data Flow with an internal Layer 4 (L4) load balancer (LB). Nevertheless, it is also possible to use your own native Layer 7 (L7) LB (Application Gateway on Azure, Application Load Balancer on AWS, respectively) in front of the Cloudera managed L4 LB.

Cloudera recommends achieving this by configuring your L7 LB to use the Cloudera Data Flow deployment LB as a backend. Enabling TLS between your LB and the Cloudera Data Flow LB is recommended, but mTLS is not possible for the backend connection. This means that your Listen Processor (e.g., ListenHTTP) in your NiFi flow cannot be configured with Client Auth = Required when using an external LB as a gateway.

You may configure the listening side of your LB and routing rules according to the requirements of your organization.

Alternatively, you may be required to use a L4 LB provided by your organization in front of the Cloudera managed LB. This is also possible, although Cloudera recommends directly using the Cloudera managed L4 LB when possible.

Typically, when using an external load balancer to act as a gateway, the internal managed load balancer should stay private. This can be accomplished by deselecting the “Use Public Endpoint” option when enabling Cloudera Data Flow for your environment, which limits Cloudera Data Flow to only use private subnets for all resources. If public access is needed, that would be done by exposing private resources via the external gateway load balancer.

### Configuration workflow

Currently, an Inbound Connection Endpoint can only be created during flow deployment, and cannot be reassigned without terminating the flow deployment for which it was created.

To configure an external load balancer, you need to go through the following steps:

- Design a flow to accept inbound connections.
  - If the flow will be used with a Layer 7 LB, a compatible Layer 7 processor (such as ListenHTTP or HandleHttpRequest) must be used.
  - Decide if TLS will be used between your LB and your flow deployment
    - Cloudera recommends enabling TLS, which is done by configuring your listen processor to use an SSL Context Service named Inbound SSL Context Service. Cloudera Data Flow will manage the certificates for this context service when deployed. mTLS between cloud native load balancers is generally not supported, so unless you know that your Gateway LB supports client certificates and mTLS for the backend connection, it is recommended to configure the Client Auth property for your Listen processor to something other than REQUIRED (that is, use NONE, AUTO, or WANT)
    - If you will terminate TLS at your Gateway LB, you can optionally choose to use an unencrypted connection on the backend, which you can do by configuring your listen processor with no SSL Context Service.
- Add the flow to the Cloudera Data Flow Catalog.
- Create a flow deployment, with an Inbound Connection Endpoint.



**Note:** If you have an existing endpoint left over after terminating a previous flow deployment, you can reuse it within the same environment.

- Download the client certificate and client key, note down the Endpoint Host Name, port number and protocol from the flow deployment where you want to send data using the external LB.
  1. Once the deployment has been created successfully, select it in the Deployments view and click Manage Deployment.
  2. In the Deployment Settings section, navigate to the **NiFi Configuration** tab to find information about the associated inbound connection endpoint.
  3. Copy the endpoint hostname and port and download the certificate and private key.



**Note:** Ports are flow-specific, all the other attributes are specific to the individual endpoint host.

- Create a Layer 7 Load Balancer in your environment and configure its backend using the endpoint information obtained from the flow deployment.

### Configure an Application Gateway in Azure

Learn about the settings required to set up an Azure Application Gateway to communicate with an Inbound Connection Endpoint.

### About this task

Create an Azure Application Gateway service (you find it in the Networking services category) using the following settings:

## Procedure

1. Make the following Backend Pool settings:

### Backend Pool without targets

Set to No.

### Backend Targets

#### IP address or FQDN

Set to the `[*** INBOUND CONNECTION ENDPOINT HOSTNAME ***]` acquired from the NiFi settings of the flow deployment where you want to connect with your gateway.

For example, `my-endpoint.inbound.dfx.p8jdxchd.xcu2-8y8x.cloudera.site`.

For all other settings you can keep the default values.



#### Tip:

You can use a single gateway for multiple flow deployments, with rules for routing traffic to the correct flow deployment. For this scenario, setup multiple backend pools, one for each flow deployment.

2. Make the following Backend Settings:

- If your flow listen processor uses TLS (recommended):

#### Backend protocol:

HTTPS

#### Trusted root certificate

Yes

#### Backend port

Match the port of your HTTP Listen Processor.

For all other settings you can keep the default values.

- If your flow listen processor does not use TLS:

#### Backend protocol

HTTP

#### Backend port

Match the port of your HTTP Listen Processor.

For all other settings you can keep the default values.



**Tip:** You can use a single Application Gateway for multiple HTTP listen processors, with rules to route traffic to the correct listen processor port. For this scenario, setup multiple backend settings, one for each listen processor.

## Related Information

[Create an application gateway](#)

## Custom NAR configs

On the Custom NAR Configs tab you can reassign and validate Custom NAR configs.



## Validating a custom NAR config

You can validate custom NAR files and their locations in the Resources view.

### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see [Cloudera Data Flow Authorization](#).

### Procedure

1. Select the **Custom NAR** that you want to validate.
2. Click  [Options]  **Validate** .  
Cloudera Data Flow displays a message about the validity of the NAR config and its storage location.

## Reassigning a custom NAR config to a different project

Learn how to reassign a custom NAR config to another project.



### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera Data Flow Authorization.

### About this task

You cannot reassign an inbound connection that is currently used by a deployment. You have to terminate the deployment using it making sure that the Delete assigned endpoint hostname option is not selected before you can reassign it to a different project.

### Procedure

1. Select the **Custom NAR** that you want to reassign.  
 **Note:** You can reassign multiple items simultaneously by selecting the checkboxes in front of them.
2. Click Reassign.  
If the custom NAR config is not used by any deployment, the **Reassign Resource** modal opens.
3. Select a Project and click Reassign.
4. Click  **Apply Changes**.

## Custom Python configs

On the Custom Python Configs tab you can reassign and validate Custom Python configs.



### Validating a custom Python config

You can validate custom Python configs and their storage locations in the Resources view.

### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera Data Flow Authorization.

### Procedure

1. Select the **Custom Python Config** that you want to validate.
2. Click  [Options]  **Validate** .  
Cloudera Data Flow displays a message about the validity of the Python config and its storage location.

## Reassigning a custom Python config to a different project

Learn how to reassign a custom Python config to another project.



### Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see [Cloudera Data Flow Authorization](#).

### Procedure

1. Select the **Custom Python Config** that you want to reassign.



**Note:** You can reassign multiple items simultaneously by selecting the checkboxes in front of them.

2. Click Reassign.

If the custom Python config is not used by any deployment, the **Reassign Resource** modal opens.

3. Select a Project and click Reassign.

4. Click  Apply Changes.

## Parameter groups

You can create groups of parameters that can be shared within a project. This allows you to disseminate common parameter values between several flow drafts and deployments, without the need of individually checking and manually updating them.

When creating a flow deployment in the deployment Wizard, creating a flow draft in Flow Design, or managing deployments in the Deployment Manager, you may add shared parameters by editing a single parameter or importing parameters from a shared parameter group.

### Editing parameters

When you edit parameters in a parameter context, either individually or as part of a parameter group, you have the option to reference parameter values from shared parameter groups available to you.

Cloudera Data Flow matches parameters by name, it does not check if actual parameter values are valid in the given context. It does, however, check for parameter type mismatches. For more information, see [Parameter type mismatch](#) on page 43.

- You are offered a list of shared parameter groups, where a parameter with the same name is available and, depending on your selection, that parameter value is substituted.
- When you click Apply, the parameter assumes the value from the selected group and the **Shared Parameter Group** column for the parameter is updated with the name of the selected group.
- If a referenced parameter group is changed, you may need to Update Parameters. For more information, see ['Update Parameters' alerts](#) on page 44.

Edit Parameter Value

CDP Workload User ?

Value \*

Enter parameter values.

☐ Set empty string

Custom


Custom


CDP creds

Apply

Cancel

### Importing parameters

You may import parameters to a parameter context from an available parameter group. To do so, select the  Import Shared Parameters option.

When you select  Import Shared Parameters for a parameter context, the **Import Parameters** dialog offers you all parameter groups available to you that have at least one matching parameter. The dialog also displays the number of matching parameters in each relevant group.

Cloudera Data Flow matches parameters by name, it does not check if actual parameter values are valid in the given context. It does, however check for parameter type mismatches. For more information, see [Parameter type mismatch](#) on page 43.

## Import Parameters



You are importing a Shared Parameter Group. To modify parameters in this group, go to the [Resources/Workspace Resources](#) page.

## Shared Parameter Group

test\_env\_parameters 2

test\_env\_parameters 2

test\_parameters 1

<input checked="" type="checkbox"/>	Parameter ↑	Current Value	Value to Import
<input type="checkbox"/>	empty param		Empty string set
<input checked="" type="checkbox"/>	file param	/nifi-shared-assets/test/file para...	→ /nifi-shared-assets/test/file para...
<input type="checkbox"/>	files param		/nifi-shared-assets/test/files par...
<input checked="" type="checkbox"/>	local file	No value set	→ assss
<input type="checkbox"/>	sens param		Sensitive value set
<input type="checkbox"/>	text param		test

Import

Cancel

- You cannot select non-matching parameters.
- Parameters that were already imported from the selected group are grayed out and cannot be selected again.

<input checked="" type="checkbox"/>	file param	/nifi-shared-assets/test/file para...	Imported from selected group
-------------------------------------	------------	---------------------------------------	------------------------------

- If a referenced parameter group is changed, you may need to Update Parameters. For more information, see ['Update Parameters' alerts](#) on page 44.

**Parameter type mismatch**

When matching parameter contexts to shared parameters, besides name matching Cloudera Data Flow also matches parameter types. Parameter type mismatch results in the following messages:

Message	Description
Sensitive value expected	The parameter in the flow definition expects a sensitive value, but the parameter in the shared group has a non-sensitive value.
Non-sensitive value expected	The parameter in the flow definition expects a non-sensitive value, but the parameter in the shared group has a sensitive value.
Text value expected	The parameter in the flow definition expects a text value, but the parameter in the shared group has a file as value.
One file expected	The parameter in the flow definition expects exactly one file as value, but the parameter in the shared group has more than one file as a value.



**Tip:** In case of parameter type mismatch, it is an option to create a new parameter group with the correct parameter types if you do not want to use custom parameter values.

## 'Update Parameters' alerts

You may run into these alerts while you edit or import parameters. They occur when a parameter group, referenced in one or more parameter contexts, has been modified. The actual change to the parameter group may or may not affect your flow, as Cloudera Data Flow has no information on the nature of the change, it simply registers the fact that a newer version of the parameter group became available. Depending on the alert type, you are required to take different courses of action.




### Update mandatory

 Referenced shared parameter group `test_env_parameters` has been modified. [Update Parameters](#)

You are in the process of editing parameters or importing parameter groups and you have not applied your changes yet. A referenced parameter group was changed in the meantime. As Cloudera Data Flow has no information on the nature of the changes, it cannot verify whether or not they collide with the unsaved changes you have made. Because of that, you are required to Update Parameters at least for the affected parameter contexts before you can apply any changes.



### Update required

 Referenced shared parameter group `test_parameters` has been modified. [Update Parameters](#)

You have referenced a parameter group in one or more parameter contexts earlier, and have already applied that change. The parameter group was modified sometime after.

You are only required to Update Parameters if you want to reference a new parameter from the updated parameter group, in a parameter context already referencing parameters from that group. This is because Cloudera Data Flow does not reference multiple versions of the same parameter group in one parameter context. Unless you opt for the Update All Parameter Contexts option, any other parameter context referencing an earlier version of the same parameter group stays on that earlier version.

You do not need to Update Parameters and you are still able to apply your changes if:

- You want to reference the updated parameter group in a parameter context not already referencing it.
- You want to reference a different parameter group in the parameter context referencing the updated parameter group.
- You want to provide a custom value to any parameter in any of the parameter contexts.

## Related Information

[Matching parameter contexts to shared parameter groups during draft creation](#)

[Matching parameter contexts to shared parameter groups during NiFi 1.x to NiFi 2.x migration](#)

[Editing parameters in a flow deployment](#)

[Flow version change](#)

# Inbound connections

An Inbound Connection Endpoint allows you to stream data from an external source application to a flow deployment.

An Inbound Connection Endpoint provides a stable hostname that can be used to send data to a Cloudera DataFlow flow deployment located in the same environment. You can create an Inbound Connection Endpoint during flow deployment, provided that the flow definition supports creating such endpoints.

An endpoint has the following attributes:

- The environment in which it exists
- A stable hostname
- An SSLContextService configured that can work with the TLS context auto created during flow deployment that corresponds to the Inbound Connection Endpoint hostname
- A server certificate that corresponds to the hostname, signed by a public root CA
- A single Intermediate CA (issued by the Environment CA) for signing client certificates for this endpoint, with the public certificate part available to download for client truststores

Endpoints exist within the environment where they were created. They cannot be moved between environments. If the environment is deleted, the endpoint gets deleted as well, and cannot be reused.

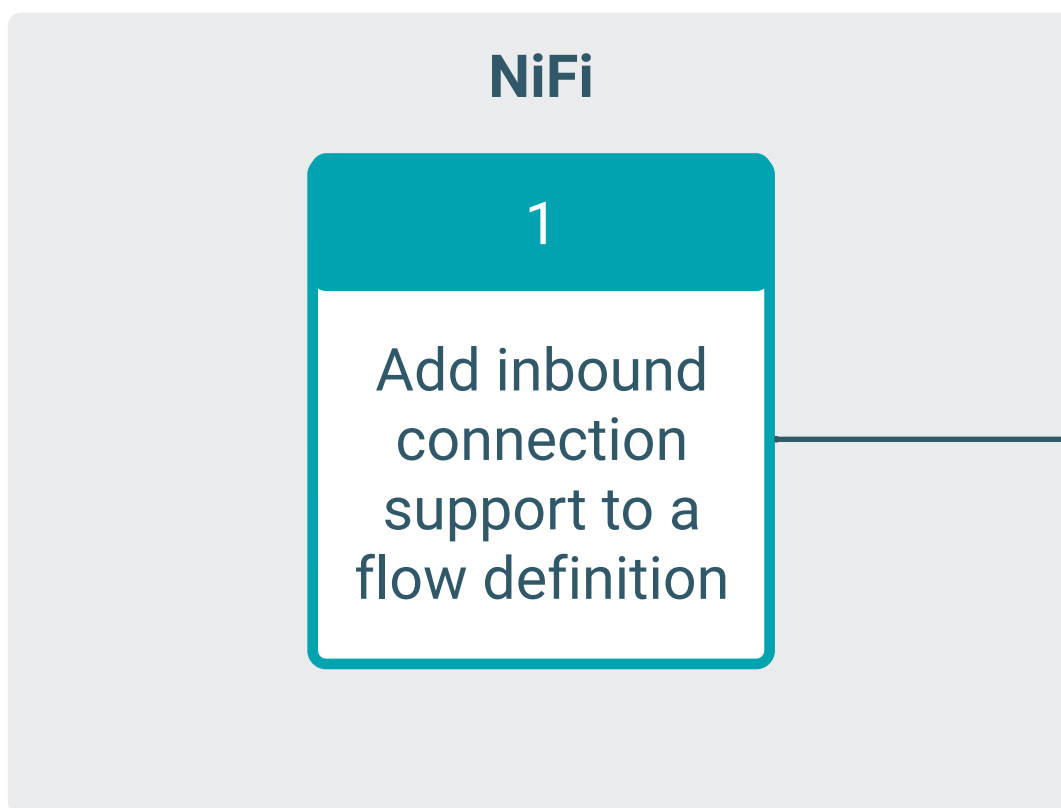
One endpoint can be assigned to one flow deployment at a time. To reassign an existing inbound connection endpoint, you need to first terminate the flow deployment to which it is currently assigned, then assign the existing endpoint to a new flow deployment in the NiFi configuration step during flow deployment.

### Workflow

Setting up an Inbound Connection Endpoint is a complex task, affecting how you develop a flow definition in NiFi and how you deploy it in Cloudera DataFlow. Once your flow has been deployed, you need to configure your client, be it directly an external application or through an external load balancer, to communicate with the Inbound Connection Endpoint of your flow deployment.

## Create an Inbound Connection Endpoint during flow deployment

During flow deployment, you can enable an endpoint that allows NiFi to listen to external data



sources.

### About this task

You can create flow deployments with an Inbound Connection Endpoint during flow deployment, using the Cloudera Data Flow Deployment wizard.



**Note:** Currently you cannot create flow deployments with Inbound Connection Endpoints using the CLI.

### Before you begin

- A flow definition supporting inbound connections has been created and added to the Catalog.
- You have DFCatalogViewer role to view flow definitions in the Catalog.
- You have DFFlowAdmin right for the environment where you want to create the endpoint.

### Procedure

1. During flow deployment, at the NiFi Configuration step, select **Inbound Connections Allow NiFi to Receive Data**.

An Endpoint Hostname, generated from the Deployment Name you provided at the Overview step is offered and port configuration options become available.

dataflow-azure / New Flow Deployment

2. You can accept the generated hostname as is, or you can change the prefix. You can also click **Select Preexisting Endpoint** to select an available endpoint within your environment.

Endpoints are unavailable if they are currently used by another flow deployment.

### 3. Select Listening Ports.

Provide the protocol and port or ports where your flow will listen to incoming data. Cloudera Data Flow automatically suggests ports based on listen processors detected in the flow definition. Custom Processors, if applicable, are not detected and must be added manually.

Using ports with mixed protocols (TCP and UDP) is not allowed. As long as the suggested ports use mixed protocols, you are not allowed to add new ports or to proceed to the next step in the Wizard.



**Note:** If you want your dataflow to use HTTP, select TCP protocol.

×

Protocol

Port ?

TCP ▼

Specify a port

+ Add Port

Protocol	Port	
TCP <span style="font-size: 18px; color: #0070C0;">?</span>	7001	- <span style="font-size: 18px; color: #0070C0;">-</span>

Cancel

OK

### 4. Specify Trusted IP Addresses.

Specify a comma-separated list of trusted CIDRs or IP ranges. To allow all traffic, select Allow all traffic.



**Important:** Selecting Allow all traffic keeps your inbound connection open to all incoming traffic. This can expose your connection to potential security risks.

### Results

As part of the flow deployment, a Layer-4 Load Balancer (LB) is created to forward traffic to the flow deployment's NiFi cluster on the configured ports and protocols. DNS records are created for the Inbound Connection Endpoint hostname to resolve to the Layer-4 LB.



**Note:** There is an initial delay for the cloud provider (AWS or Azure) to fully provision and start the LB resources, during which the flow deployment appears in the Cloudera Data Flow monitoring dashboard as healthy, but it is not yet able to receive data from clients. The load balancer provisioning normally completes within 10 minutes of the flow deployment being finished, but can take up to three hours.

### What to do next

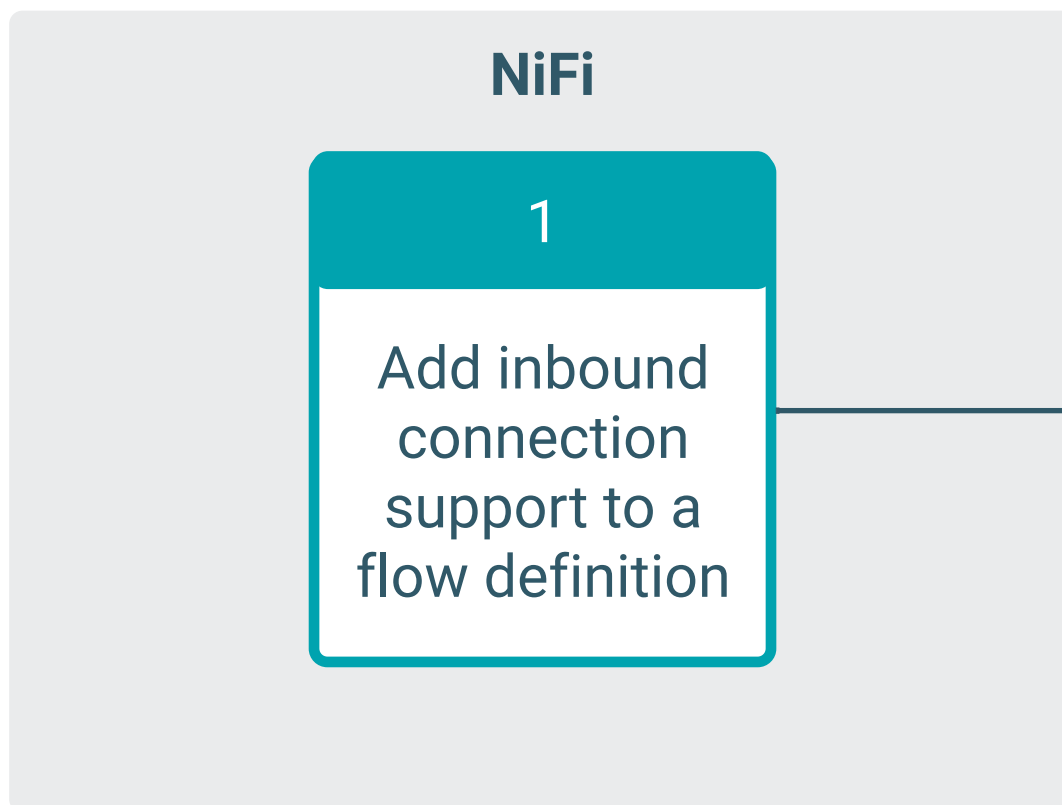
If the listening port in your flow definition is parameterized instead of having a pre-defined value, you have to fill in the same listening port numbers you provided in the NiFi Configuration step. The listening ports you specified in the NiFi Configuration step are visible in the Overview pane.

### Related Information

[Deploying a flow definition using the wizard](#)

## Connecting applications to an endpoint

Once a Cloudera Data Flow flow deployment with inbound connection is available, you can go on and connect an external application to start sending data.



### Before you begin

- A flow deployment with inbound connection is available.
- A network connection through which the client can reach the flow deployment endpoint is available.
- You have been assigned at least the DFFlowUser role for the environment to which you want to configure the inbound connection.

### Procedure

1. Select the flow deployment where you want to send data and go to **Deployment Manager Deployment Settings**.
2. Select the **NiFi Configuration** tab.
3. Make a note of the **Endpoint Hostname** and **port**.
4. Click **Download Client Certificate**.

The X.509 client certificate downloads to your computer in PEM format.

5. Click **Download Client Private Key** to obtain the RSA Private Key.

The unencrypted RSA Private Key encoded with PKCS8 downloads to your computer in PEM format.



- Depending on your client, you may have to convert the certificate and the private key to a different format.

For example, to convert PEM to PKCS12 format, use the following command:

```
openssl pkcs12 -export -in [***DOWNLOADED PEM CERT FILE***] -in
key [***DOWNLOADED PEM PRIVATE KEY***] -out certificate.p12
```

To further convert the PKCS12 file to JKS format for a Java client, run the following command:

```
keytool -importkeystore -srckeystore [***CERTIFICATE NAME***].p12 -srcsto
retype pkcs12 -destkeystore [***DESTINATION KEYSTORE***].jks
```

- Add the certificate file and the private key files to the keystore of your application.
- Configure your application to stream data to the Endpoint Hostname, port, and protocol of the flow deployment.

## TLS keys and certificates

When using Inbound Connection Endpoints, sensitive information is sent over the network between Cloudera Data Flow (CDF) and external data sources including configuration files that contain passwords. To secure this transfer, Cloudera strongly recommends that you configure mutual Transport Layer Security (TLS) encryption.

TLS is an industry standard set of cryptographic protocols for securing communications over a network.

Configuring TLS involves creating a private key and a public key for use by server and client processes to negotiate an encrypted connection. In addition, TLS can use certificates to verify the trustworthiness of keys presented during the negotiation to prevent spoofing and mitigate other potential security issues.

## Tutorial: MiNiFi to Cloudera Data Flow flow deployment

This tutorial walks you through creating an inbound connection endpoint in Cloudera Data Flow used by a flow deployment to receive data from one or more MiNiFi agents managed by Edge Flow Manager.

### Before you begin

- You have an enabled and healthy Cloudera Data Flow environment.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Cloudera Data Flow Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.
- You have an enabled and healthy Cloudera Edge Management (CEM) environment.
- You have sufficient rights to configure the MiNiFi Agent in EFM.

### Procedure

- In a development NiFi environment, create a Controller Service of type StandardRestrictedSSLContextService at the root canvas level and name it Inbound SSL Context Service.
  - In the Operate palette click Configuration Controller Services Create a new controller service
  - Filter for ssl, select StandardRestrictedSSLContextService then click Add.
  - Click Configure.
  - On the **Settings** tab change the Name to Inbound SSL Context Service, then click Apply.

You do not need to make further configuration on this Controller Service; it acts as a placeholder and will be created with a managed SSL Context when deployed by Cloudera Data Flow.



#### Tip:

For testing the dataflow during development, the SSL Context may be configured with test keys and certificates for your development NiFi environment.

2. Create a Process Group on the root canvas to hold your flow definition and give it a name.  
This tutorial uses the name ListenHTTP Flow.
3. Enter the process group.
4. Inside the Process Group, add a listen processor.  
This tutorial uses ListenHTTP.
5. Configure the listen processor:

**Base Path**

This tutorial uses the default contentListener.

**Listening Port**

Define a value that is valid for your use case. This tutorial uses port 9000.

**SSL Context Service**

Select Inbound SSL Context Service.

**Client Authentication**

Select REQUIRED.

Click Apply.

**Configure Processor** | ListenHTTP 1.15.3

Invalid

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

Required field

Property	Value
<b>Base Path</b>	contentListener
<b>Listening Port</b>	9000
Listening Port for Health Check Requests	No value set
Max Data to Receive per Second	No value set
SSL Context Service	Inbound SSL Context Service →
Client Authentication	REQUIRED
<b>Authorized Subject DN Pattern</b>	.*
Authorized Issuer DN Pattern	.*
<b>Max Unconfirmed Flowfile Time</b>	60 secs
HTTP Headers to receive as Attributes (Regex)	No value set
Return Code	200
<b>Multipart Request Max Size</b>	1 MB

CANCEL

APPLY

6. Connect the ListenHTTP processor to a downstream processor of your choice.  
This tutorial uses LogAttribute, where all relationships terminate.
7. From the root canvas, right click on the Process Group and select Download flow definition Without external controller services .
8. Upload the flow definition JSON to the Flow Catalog of your Cloudera Data Flow deployment.

**9. Deploy the flow.**

- a) At the NiFi Configuration step of the Deployment wizard, select **Inbound Connections Allow NiFi to Receive Data** to enable inbound connections.

Accept the automatically created endpoint hostname and automatically discovered port by clicking **Next**.

- b) At **Parameters**, click **Next**.
- c) At **Sizing & Scaling** select the **Extra Small NiFi Node Size** then click **Next**.
- d) Add a KPI on the **ListenHTTP** processor to monitor how many bytes it is receiving, by clicking **Add new KPI**.

Make the following settings:

**KPI Scope**

Processor

**Processor Name**

ListenHTTP

**Metric to Track**

Bytes Received

Add new KPI
✕

Details

KPI Scope ?
Processor Name ?

Processor
ListenHTTP

Metric to Track ?

Bytes Received

METRIC DESCRIPTION:  
Number of bytes received from a source that is external from the flows

Alerts

☐ Trigger alert when metric is greater than
Value
MBytes

☐ Trigger alert when metric is less than
Value
MBytes

Alert will be triggered when metric is outside the boundary(s) for
Value
Minutes

Cancel
Add

e) Review the information provided and click Deploy.

Soon after the flow deployment has started, the client certificate and private key required for sending data to the NiFi flow become available for the flow deployment that is being created.

10. Collect the information required to configure your load balancer.

- a. Once the deployment has been created successfully, select it in the Deployments view and click Manage Deployment.
- b. In the Deployment Settings section, navigate to the **NiFi Configuration** tab to find information about the associated inbound connection endpoint.
- c. Copy the endpoint hostname and port and download the certificate and private key.

**11. Start designing your MiNiFi flow in EFM.****For C++**

To design a flow for your MiNiFi C++ agent class:

- a. Copy the downloaded client-private-key-encoded key and client-certificate-encoded.cer certificate files to the host with the running MiNiFi C++ agent, so they are accessible by filepath from the agent.
- b. Create a Service of type SSL Context Service with the following configuration:

**Service Name**

Specify a name for this service. This tutorial uses Client SSL Context Service.

**CA Certificate**

Leave it empty. As Cloudera Data Flow uses Let's Encrypt as a Certificate Authority, the certificate will be accepted automatically, without additional configuration.

**Client Certificate**

[\*\*\*/PATH/TO/\*\*/]client-certificate-encoded.cer

For example, /opt/minifi/minifi-test/client-certs/client-certificate-encoded.cer.

**Passphrase**

Set no value.

**Private Key**

[\*\*\*/PATH/TO/\*\*/]client-private-key-encoded

For example, /opt/minifi/minifi-test/client-certs/client-private-key-encoded

**Use System Cert Store**

Keep the default False value.

»

SSLContextService (Service)

## Configuration

### Settings

Service Name\*

Client SSL Context Service

### Properties

Property	Value
CA Certificate	<a href="#">?</a> No value set <a href="#">↑</a>
Client Certificate	<a href="#">?</a> /opt/minifi/minifi-test/client-certs/client-certificate-encoded... <a href="#">↑</a>
Passphrase	<a href="#">?</a> No value set <a href="#">↑</a>
Private Key	<a href="#">?</a> /opt/minifi/minifi-test/client-certs/client-private-key-encoded <a href="#">↑</a>
Use System Cert Store	<a href="#">?</a> false <a href="#">↑</a>

### About

SERVICE ID  
8d7cb8fd-af49-480c-8399-d0845fbb95de

SERVICE TYPE  
SSLContextService 1.22.04

BUNDLE  
org.apache.nifi.minifi - minifi-system

### Comments

Describe the changes made in this update

Apply

c. Click Apply.

d. Create an InvokeHTTP processor named Send to CDF with the following configuration:

#### Automatically Terminated Relationships

Select all relationships.

#### Content-type

Depends on your flow file data type. This tutorial uses text/plain.

#### HTTP Method

POST

#### Remote URL

https://[\*\*\*ENDPOINT HOSTNAME COPIED FROM CLOUDERADATAFLOW FLOW DEPLOYMENT MANAGER\*\*\*]:9000/contentListener

For example, https://my-flow.inbound.my-dfx.c94x5i9m.xcu2-8y8z.mycompany.test:9000/contentListener

## SSL Context Service

### Client SSL Context Service

Leave all other settings with their default values.

The screenshot displays the Cloudera Data Flow interface for configuring a flow. On the left, the 'Design / Agent Class' pane shows a flow diagram for the 'minifi-cpp-latest' agent class. The flow starts with a 'GenerateFlowFile' processor, followed by a 'success' label, and then a 'Send to CDF (IngestHTTP)' processor, which is highlighted with a blue circle. On the right, the 'Configuration' pane is visible, showing settings for the 'Send to CDF (Processor)'.

**Configuration**

Automatically Terminated Relationships

☒ failure ☒ no retry ☒ response ☒ retry ☒ success

**Scheduling**

Scheduling Strategy\* Timer Driven

Concurrent Tasks\* 1

Run Schedule\* 1000 ms

Run Duration\*

0ms 25ms 50ms 100ms 250ms 500ms 1s 2s

**Properties**

Property	Value
Always Output Response	<input type="radio"/> false
Attributes to Send	<input type="radio"/> No value set
Connection Timeout	<input type="radio"/> 5 s
Content-type	<input type="radio"/> application/plain-text
Disable Peer Verification	<input type="radio"/> false
Follow Redirects	<input type="radio"/> true
HTTP Method	<input type="radio"/> POST

Apply

### For Java

To design a flow for your MiNiFi Java agent class:

- a. Convert the downloaded client-private-key-encoded key and client-certificate-encoded.cer certificate files to a JKS Keystore:
  1. Create a PKCS12 keystore:
 

```
openssl pkcs12 -export -in client-certificate-encoded -inkey client-private-key-encoded -out client-keystore.p12
```
  2. Convert the PKCS12 keystore to a JKS keystore:
 

```
keytool -importkeystore -srckeystore client-keystore.p12 -srcstoretype pkcs12 -destkeystore client-keystore.jks
```
- b. Copy the resulting client-keystore.jks file to the host with the running MiNiFi Java agent, so they are accessible by filepath from the agent.
- c. Obtain the CA root cert and add it to truststore client-truststore.jks, by running the following commands:

```
wget https://letsencrypt.org/certs/isrgrootx1.pem
```

```
keytool -import -file isrgrootx1.pem -alias isrgrootx1 -keystore client-truststore.jks
```

MiNiFi Java requires you to specify an explicit truststore for inbound connections. Remember the password you used for creating client-truststore.jks, as you will need it .

- d. Create a Service of type Restricted SSL Context Service with the following configuration:

#### Service Name

Specify a name for this service. This tutorial uses Client SSL Context Service.

#### Keystore Filename

[\*\*\*/PATH/TO/\*\*/]client-truststore.jks

#### Keystore Password

[\*\*\*THE PASSWORD YOU PROVIDED WHEN CREATING THE JKS STORE\*\*\*]

#### Key Password

[\*\*\*THE PASSWORD YOU PROVIDED WHEN CREATING THE JKS STORE\*\*\*]

#### Keystore Type

JKS

#### Truststore Filename

client-truststore.jks

#### Truststore Type

JKS

#### Truststore Password

[\*\*\*THE PASSWORD YOU PROVIDED WHEN CREATING THE CLIENT TRUSTSTORE\*\*\*]

The screenshot displays the Cloudera Edge Management (CEM) interface. On the left is a dark sidebar with navigation icons for Monitor, Edge Events, Design (highlighted), and Agent Manager. The main area is titled 'Design / Agent Class' and contains a 'Services' section with a list of services, including 'Client SSL Context Service'. To the right of the services list is a 'Configuration' panel for the selected service. This panel includes a 'Settings' section with a 'Service Name\*' field containing 'Client SSL Context Service'. Below this is a 'Properties' table with various configuration parameters and their values. At the bottom of the configuration panel is an 'About' section with service details. An 'Apply' button is located at the bottom right of the configuration panel.

Property	Value
Keystore Filename	/opt/minifi/minifi-current/client-cert...
Keystore Password	#{Keystore Password}
Key Password	#{Key Password}
Keystore Type	JKS
Truststore Filename	Empty string set
Truststore Password	No value set
Truststore Type	No value set
TLS Protocol	TLS

**About**

SERVICE ID  
f19bd51a-db0e-47ad-916e-542d4482590b

SERVICE TYPE  
StandardRestrictedSSLContextService 1.3.1.0

BUNDLE  
org.apache.nifi.minifi - minifi-ssl-context-service-nar



- e. Click Apply.
- f. Create an InvokeHTTP processor named Send to CDF with the following configuration:  
**Automatically Terminated Relationships**  
 Select all relationships.

### Content-type

Depends on your flow file data type. This tutorial uses text/plain.

### HTTP Method

POST

### Remote URL

`https://[***ENDPOINT HOSTNAME COPIED FROM CLOUDERA DATAFLOW FLOW DEPLOYMENT MANAGER***]:9000/contentListener`

For example, `https://my-flow.inbound.my-dfx.c94x5i9m.xcu2-8y8z.mycompany.test:9000/contentListener`

### SSL Context Service

Client SSL Context Service

Leave all other settings with their default values.

**Configuration**

Processor Name\*  
Send to CDF

Penalty Duration\* 0 ms Yield Duration\* 0 ms

Automatically Terminated Relationships  
☒ Original ☒ Failure ☒ Retry ☒ No Retry ☒ Response

**Scheduling**

Scheduling Strategy\* Timer Driven Concurrent Tasks\* 1

Run Schedule\* 0 ms

Run Duration\* 0ms 25ms 50ms 100ms 250ms 500ms 1s 2s

**Properties**

Property	Value
HTTP Method	POST
Remote URL	https://kevindemo.inbound.dfx.p8jdcchd.xcu2-8y...
SSL Context Service	Client SSL Context Service
Connection Timeout	5 secs
Read Timeout	15 secs

Apply

12. Build the rest of your data flow to read data and send to your Cloudera Data Flow flow deployment using InvokeHTTP. As a simple example, this tutorial uses the GenerateFlowFile processor, with the following settings:

Run Schedule

Set to 10000 ms (10 seconds).

Custom Text

The message you type here will be sent to the ListenHTTP Flow you have created, with the frequency specified by Run Schedule. For example, Hello DFX! This is MiNiFi.

Data Format

Set to Text.

Unique FlowFiles

Set to false.

Design / Agent Class

PROCESSOR

REMOTE PROCESS GROUP

FUNNEL

minifi-cpp-latest

GenerateFlowFile  
GenerateFlowFile

NAME success

Send to CDF  
InvokeHTTP

SERVICESPARAMETERS

>>

GenerateFlowFile (Processor)

Configuration

Penalty Duration\*

30000 ms

Yield Duration\*

1000 ms

Automatically Terminated Relationships

☐ SUCCESS

Scheduling

Scheduling Strategy\*

Timer Driven

Concurrent Tasks\*

1

Run Schedule\*

10000 ms

Run Duration\*

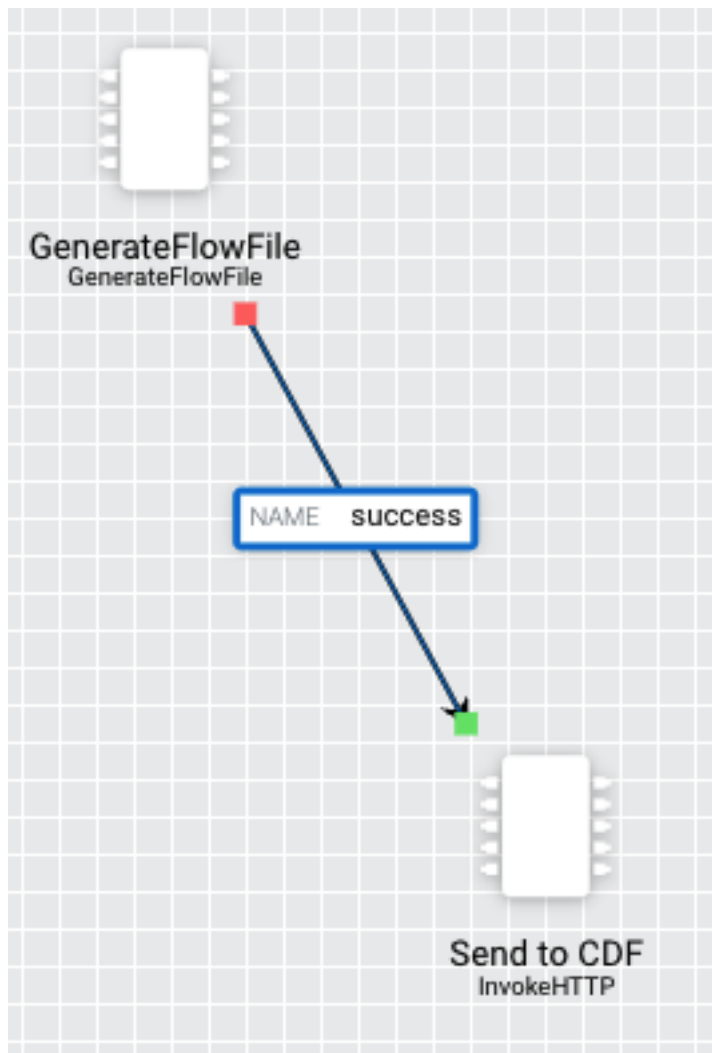
0ms25ms50ms100ms250ms500ms1s2s

Properties

Property	Value	
Batch Size	1	↑
Custom Text	Hello DFX! This is MINIFI.	↑
Data Format	Text	↑
File Size	1kB	↑
Unique FlowFiles	false	↑

Apply

13. Connect the GenerateFlowFile processor to the InvokeHTTP processor.



14. Click Actions Publish... to publish the flow and start it on your MiNiFi agent.

15. Select your flow deployment in the Cloudera Data Flow Dashboard and click KPIs.

Observe that your Cloudera Data Flow flow deployment is now receiving data from MiNiFi.

#### Related Tasks

[Connecting applications to an endpoint](#)

#### Related Information

[Creating a flow definition in NiFi](#)

[Deploying a flow definition](#)

[Building a dataflow in CEM](#)

[Publishing a dataflow in CEM](#)

## Tutorial: Invoking an HTTP endpoint with curl

This tutorial walks you through invoking an HTTP Inbound Connection Endpoint with curl using the ListenHTTP filter to Kafka ReadyFlow from the ReadyFlow Gallery.

**Before you begin**

- You have an enabled and healthy Cloudera Data Flow environment.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Cloudera Data Flow Catalog.
- You have added the ListenHTTP filter to Kafka ReadyFlow to the Catalog. Adding a ReadyFlow to the Catalog requires the DFCatalogAdmin role.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.
- You have network connectivity established from your curl client to the virtual network or VPC where your Flow Deployment will run.
- You have downloaded and installed curl.

## Procedure

1. Deploy the ListenHTTP filter to Kafka Ready Flow.
  - a. Navigate to the ReadyFlow Gallery, locate the ListenHTTP filter to Kafka ReadyFlow and click View Added Flow Definition.
  - b. Click Deploy and select your target environment to start the Deployment Wizard for the latest version of this ReadyFlow.
  - c. Specify a deployment name, for example, Inbound Connections curl and click Next.
  - d. Select Allow NiFi to receive data checkbox to configure an endpoint host.
  - e. Accept the automatically created endpoint hostname and automatically discovered port by clicking Next.
  - f. Optional: This ReadyFlow performs schema validation for incoming events using Cloudera's Schema Registry before sending the events to a Kafka topic. If you have a Streams Messaging cluster available, fill in the Kafka and Schema Registry connection properties.

If you only want to validate inbound connection endpoint connectivity, enter dummy values for the empty parameters, set the Input and Output format to JSON while keeping the Listening Port set to 7001.

- g. At Sizing & Scaling select the Extra Small NiFi Node Size and click Next.
- h. Add a KPI on the ListenHTTP processor to monitor how many bytes it is receiving, by clicking Add new KPI.

Make the following settings:

**KPI Scope**

Processor

**Processor Name**

ListenHTTP

**Metric to Track**

Bytes Received

Add new KPI
✕

Details

KPI Scope ?
Processor

Processor Name ?
ListenHTTP

Metric to Track ?

Bytes Received

METRIC DESCRIPTION:  
Number of bytes received from a source that is external from the flows

Alerts

☐ Trigger alert when metric is greater than
Value
MBytes

☐ Trigger alert when metric is less than
Value
MBytes

Alert will be triggered when metric is outside the boundary(s) for
Value
Minutes

Cancel
Add

i. Click Next.

j. Review the information provided and click Deploy.

Soon after the flow deployment has started, the client certificate and private key required for sending data to the NiFi flow become available for the flow deployment that is being created.

2. Collect the information required to configure your load balancer.

- a. Once the deployment has been created successfully, select it in the Deployments view and click Manage Deployment.
- b. In the Deployment Settings section, navigate to the **NiFi Configuration** tab to find information about the associated inbound connection endpoint.
- c. Copy the endpoint hostname and port and download the certificate and private key.

### 3. Create the curl request to validate connectivity to the HTTP inbound connection endpoint.

Using the endpoint hostname, port, client certificate and private key you can now construct a curl command to call the endpoint and validate connectivity:

```
curl -v -X POST https://[***ENDPOINT_HOSTNAME***]:7001/contentListener --key [***PATH/TO/
***]client-private-key-encoded --cert [***PATH/TO/***]client-certificate-encoded
```

You receive an HTTP 200 response code in a similar message, indicating that your client was able to securely connect to the inbound connection endpoint:

```
* Trying 10.36.84.149:7001...
* Connected to [***ENDPOINT_HOSTNAME***] (10.36.84.149) port 7001 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
* CAfile: /Users/mkohs/letsencrypt-stg-root-x1.pem
* CAPath: none
* (304) (OUT), TLS handshake, Client hello (1):
* (304) (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Request CERT (13):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS handshake, CERT verify (15):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
* subject: CN=dfx.dtefgqis.xcu2-8y8x.dev.cldr.work
* start date: May 11 21:02:20 2022 GMT
* expire date: Aug 9 21:02:19 2022 GMT
* subjectAltName: host "[***ENDPOINT_HOSTNAME***]" matched cert's
  "[***ENDPOINT_HOSTNAME***]"
* issuer: C=US; O=(STAGING) Let's Encrypt; CN=(STAGING) Artificial Apricot R3
* SSL certificate verify ok.
> POST /contentListener HTTP/1.1
> Host: [***ENDPOINT_HOSTNAME***]:7001
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 12 May 2022 00:45:57 GMT
< Content-Type: text/plain
< Content-Length: 0
< Server: Jetty(9.4.46.v20220331)
<
* Connection #0 to host [***ENDPOINT_HOSTNAME***] left intact
```

### 4. Use curl to post data to the HTTP inbound connection endpoint.

If you want to post events to the NiFi deployment you can add header and content definition to the curl request.

This example sends JSON data following a simple schema to the endpoint:

```
curl -v -X POST [***ENDPOINT_HOSTNAME***]:7001/contentListener \
--key [***PATH/TO/***]client-private-key-encoded \
--cert [***PATH/TO/***]client-certificate-encoded \
```

```
-H 'Content-Type: application/json' \  
-d '{"created_at":6453,"id":6453,"text":"This is my test event","timestamp_ms":34534,"id_store":12}'
```

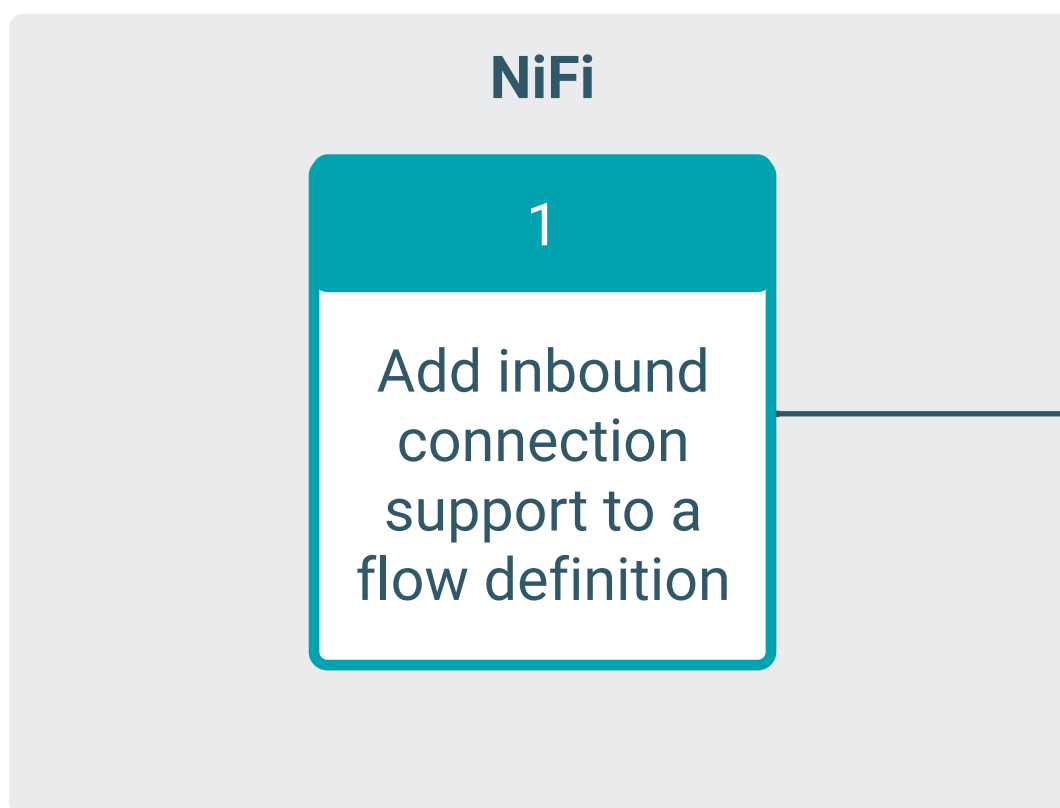
The NiFi deployment tries to validate the schema against the schema name and Schema Registry that you provided when deploying the ReadyFlow. If you provided dummy values, you receive a response indicating that the flow was unable to look up the schema.

### Related Information

[ReadyFlow overview: ListenHTTP to CDP Kafka](#)

## Using Inbound Connections with an external load balancer

Once a Cloudera Data Flow deployment with an Inbound Connection Endpoint is available, you can go on and connect an external load balancer to start sending data.



Inbound Connection Endpoints are created in Cloudera Data Flow with an internal Layer 4 (L4) load balancer (LB). Nevertheless, it is also possible to use your own native Layer 7 (L7) LB (Application Gateway on Azure, Application Load Balancer on AWS, respectively) in front of the Cloudera managed L4 LB.

Cloudera recommends achieving this by configuring your L7 LB to use the Cloudera Data Flow deployment LB as a backend. Enabling TLS between your LB and the Cloudera Data Flow LB is recommended, but mTLS is not possible for the backend connection. This means that your Listen Processor (e.g., ListenHTTP) in your NiFi flow cannot be configured with Client Auth = Required when using an external LB as a gateway.

You may configure the listening side of your LB and routing rules according to the requirements of your organization.

Alternatively, you may be required to use a L4 LB provided by your organization in front of the Cloudera managed LB. This is also possible, although Cloudera recommends directly using the Cloudera managed L4 LB when possible.



Typically, when using an external load balancer to act as a gateway, the internal managed load balancer should stay private. This can be accomplished by deselecting the “Use Public Endpoint” option when enabling Cloudera Data Flow for your environment, which limits Cloudera Data Flow to only use private subnets for all resources. If public access is needed, that would be done by exposing private resources via the external gateway load balancer.

### Configuration workflow

Currently, an Inbound Connection Endpoint can only be created during flow deployment, and cannot be reassigned without terminating the flow deployment for which it was created.

To configure an external load balancer, you need to go through the following steps:

- Design a flow to accept inbound connections.
  - If the flow will be used with a Layer 7 LB, a compatible Layer 7 processor (such as ListenHTTP or HandleHttpRequest) must be used.
  - Decide if TLS will be used between your LB and your flow deployment
    - Cloudera recommends enabling TLS, which is done by configuring your listen processor to use an SSL Context Service named Inbound SSL Context Service. Cloudera Data Flow will manage the certificates for this context service when deployed. mTLS between cloud native load balancers is generally not supported, so unless you know that your Gateway LB supports client certificates and mTLS for the backend connection, it is recommended to configure the Client Auth property for your Listen processor to something other than REQUIRED (that is, use NONE, AUTO, or WANT)
    - If you will terminate TLS at your Gateway LB, you can optionally choose to use an unencrypted connection on the backend, which you can do by configuring your listen processor with no SSL Context Service.
- Add the flow to the Cloudera Data Flow Catalog.
- Create a flow deployment, with an Inbound Connection Endpoint.



**Note:** If you have an existing endpoint left over after terminating a previous flow deployment, you can reuse it within the same environment.

- Download the client certificate and client key, note down the Endpoint Host Name, port number and protocol from the flow deployment where you want to send data using the external LB.
  1. Once the deployment has been created successfully, select it in the Deployments view and click Manage Deployment.
  2. In the Deployment Settings section, navigate to the **NiFi Configuration** tab to find information about the associated inbound connection endpoint.
  3. Copy the endpoint hostname and port and download the certificate and private key.



**Note:** Ports are flow-specific, all the other attributes are specific to the individual endpoint host.

- Create a Layer 7 Load Balancer in your environment and configure its backend using the endpoint information obtained from the flow deployment.

### Configure an Application Gateway in Azure

Learn about the settings required to set up an Azure Application Gateway to communicate with an Inbound Connection Endpoint.

#### About this task

Create an Azure Application Gateway service (you find it in the Networking services category) using the following settings:

## Procedure

1. Make the following Backend Pool settings:

### Backend Pool without targets

Set to No.

### Backend Targets

#### IP address or FQDN

Set to the `[*** INBOUND CONNECTION ENDPOINT HOSTNAME ***]` acquired from the NiFi settings of the flow deployment where you want to connect with your gateway.

For example, `my-endpoint.inbound.dfx.p8jdxchd.xcu2-8y8x.cloudera.site`.

For all other settings you can keep the default values.



#### Tip:

You can use a single gateway for multiple flow deployments, with rules for routing traffic to the correct flow deployment. For this scenario, setup multiple backend pools, one for each flow deployment.

2. Make the following Backend Settings:

- If your flow listen processor uses TLS (recommended):

#### Backend protocol:

HTTPS

#### Trusted root certificate

Yes

#### Backend port

Match the port of your HTTP Listen Processor.

For all other settings you can keep the default values.

- If your flow listen processor does not use TLS:

#### Backend protocol

HTTP

#### Backend port

Match the port of your HTTP Listen Processor.

For all other settings you can keep the default values.



**Tip:** You can use a single Application Gateway for multiple HTTP listen processors, with rules to route traffic to the correct listen processor port. For this scenario, setup multiple backend settings, one for each listen processor.

## Related Information

[Create an application gateway](#)

# Best practices for building custom components

Learn about general guidelines concerning the creation of custom NiFi archives (NARs).

The goal is to build your code once against a baseline version of Apache NiFi and it can be deployed to any flow you need using any version of Cloudera DataFlow powered by any NiFi deployment equal to or greater than the version it was built against, bar major version changes.

Apache NiFi extensions are packaged in NARs. A NAR allows several components and their dependencies to be packaged together into a single package. NiFi provides Maven archetypes for creating custom processor and

controller service bundle project structures. For detailed information, see the [Maven Projects for Extensions](#) Apache NiFi wiki page.

- If directly specifying the nifi-nar-maven-plugin, ensure you use the most recent version when compiling your custom code.
- If inheriting the nifi-nar-maven-plugin, Cloudera recommends that the parent version of nifi-nar-bundles has the same major and minor version as the selected NiFi runtime version. For example, If the CFM Version is 1.18.0.2.3.7.1 (NiFi Major Version: 1, NiFi Minor Version: 18) the recommended compilation version is 1.18.0 (The first two version numbers must be equal to, or less than the CFM version).

The screenshot shows the 'NiFi Configuration' page. A modal dialog titled 'Change NiFi Runtime Version' is open. The dialog has a 'Version Number' dropdown menu currently showing 'Version 1.18.0.2.3.7.1-1'. Below the dropdown, it says 'Using the latest version is recommended unless there is a specific need for a different version.' There are 'Cancel' and 'Apply' buttons at the bottom right of the dialog. In the background, the 'NiFi Configuration' page is visible, showing the 'NiFi Runtime Version' section with 'CURRENT VERSION Latest Version (1.18.0.2.3.8.0-11)' and a 'Change Version' link. Other sections include 'Autostart Behavior' (checked for automatic start), 'Inbound Connections' (unchecked for allowing data), and 'Custom NAR Configuration' (unchecked for using custom NARs).

```
<parent>
  <groupId>org.apache.nifi</groupId>
  <artifactId>nifi-nar-bundles</artifactId>
  <version>1.18.0.2.3.7.1-1</version>
</parent>
```

- Ensure your NAR pom only depends on another NAR pom for a controller service API NAR. Generally, do not extend from implementation NARs like nifi-standard-nar.
- Ensure your components jar pom marks API dependencies as provided in order to obtain them at runtime through the appropriate NAR dependency.

## Best practices for packaging custom Python processors [Technical Preview]

Depending on complexity and possible shared dependencies, you need to decide whether to create your custom processor as a single file or as a package.

This documentation describes your options on packaging your custom Python processor and making it available for flow deployments in Cloudera Data Flow, and is based on the official Apache NiFi 2 documentation. For additional best practices on writing a custom Python processor for a NiFi 2.x flow, consult the official Apache NiFi documentation. Python processors can be packaged either as a single Python file, or as a Python package.

## Single Python File

If the processor is simple and does not share dependencies with any other custom processor, it is easiest to have a single Python file named after the processor, like `CreateFlowFile.py`. In the single Python file format dependencies are specified directly in the processor.

For example:

```
class PandaProcessor(FlowFileTransform)
    class Java:
        implements = ['org.apache.nifi.pythonprocessor.FlowFileTransform']
    class ProcessorDetails:
        version = '0.0.1-SNAPSHOT',
        dependencies = ['pandas', 'numpy==1.20.0']
```

## Python package

If more than one custom Python processor uses the same dependencies, or if you have a helper module that you want to use in one or more Python processors, a Python package is required. Structure your code as follows:

```
my-python-package/
#
## __init__.py
#
## ProcessorA.py
#
## ProcessorB.py
#
## HelperModule.py
#
## requirements.txt
```

In this example, all requirements across the processors and helper modules appear in `requirements.txt`, and both `ProcessorA` and `ProcessorB` can reference code in the helper module in a way similar to the following:

```
from HelperModule import my_helper_function
```

When uploading a Python package to cloud storage for use in Cloudera Data Flow, add the package directory (`my-python-package` in this example) directly inside the cloud storage directory that you are going to specify during deployment.

For example, if you specify `s3a://bucket-name/custom-python` as your cloud storage directory in the wizard, the following files should exist in cloud storage:

```
s3://my-bucket/custom-python/my-python-package/__init__.py
s3://my-bucket/custom-python/my-python-package/ProcessorA.py
s3://my-bucket/custom-python/my-python-package/ProcessorB.py
s3://my-bucket/custom-python/my-python-package/HelperModule.py
s3://my-bucket/custom-python/my-python-package/requirements.txt
```

## Making the processor available for Cloudera Data Flow

In order to make a custom Python processor available to Cloudera Data Flow, upload it to cloud storage as described in [Preparing cloud storage to deploy custom processors](#).

In the deployment wizard, specify this directory as Custom Python processor Storage Location.

The screenshot shows the 'NiFi Configuration' page in Cloudera Data Flow. The 'NiFi Runtime Version' section has two options: 'Latest Version (1.27.0.2.3.15.0-8)' and 'nifi 2.x Tech Preview'. The 'Autostart Behavior' section has a checked checkbox for 'Automatically start flow upon success'. The 'Inbound Connections' section has an unchecked checkbox for 'Allow NiFi to receive data'. The 'Custom NAR Configuration' section has an unchecked checkbox for 'This flow deployment uses custom NAR'. The 'Custom Python Configuration' section has a checked checkbox for 'This flow deployment uses custom python processors'. A modal dialog titled 'Custom Python processor Storage Location' is open, showing fields for 'CDP Workload Username' (workload\_user), 'Password' (masked), 'Confirm Password' (masked), and 'Storage Location' (abfs:// my-bucket/custom-python). The dialog also has 'Save' and 'Cancel' buttons.

**NiFi Configuration**

NiFi Runtime Version

☐ Latest Version (1.27.0.2.3.15.0-8)

Change Version

1.x is the stable version of NiFi, compatible with the existing flows

**Autostart Behavior**

☒ Automatically start flow upon success

**Inbound Connections**

☐ Allow NiFi to receive data

**Custom NAR Configuration**

☐ This flow deployment uses custom NAR

**Custom Python Configuration**

☒ This flow deployment uses custom python processors

**Custom Python processor Storage Location**

To use custom Python processors, provide CDP access credentials and the path to their object store location. These settings will be used only for this deployment.

CDP Workload Username

workload\_user

Password

.....

Confirm Password

.....

Storage Location

abfs:// my-bucket/custom-python

Save Cancel

## Preparing cloud storage to deploy custom processors

To use a custom Apache NiFi processor or controller service in one of your Cloudera DataFlow flow deployments, add the NiFi Archive (NAR), Python file, or Python package containing the custom processor or controller service to a cloud storage location for later use during a flow deployment.

### Procedure

1. Create your cloud storage location.
2. Upload your NAR file, Python file, or Python package to the cloud storage location.

### 3. Configure access to your cloud provider storage in one of two ways:

- You have configured access to S3 buckets using ID Broker mapping.

If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

#### a. Access IDBroker mappings.

- To access IDBroker mappings in your environment, click **Actions Manage Access**.
  - Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click **Edit**.
- b. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the **Current Mappings** section by clicking the blue + sign.



**Note:** You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.

#### c. Click **Save and Sync**.

- You have configured access to S3 buckets with a RAZ enabled environment.

It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

- Ensure that Fine-grained access control is enabled for your Cloudera Data Flow environment.
- From the Ranger UI, navigate to the S3 repository.
- Create a policy to govern access to the S3 bucket and path used in your ingest workflow.



#### Tip:

The Path field must begin with a forward slash (/).

- Add the machine user that you have created for your ingest workflow to the policy you just created.

For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.

### 4. Note the workload user name and password, and cloud storage location to use in the Deployment Wizard.



**Tip:** If you want to provide a machine user as CDP Workload Username during flow deployment, make sure to note the full workload user name including the `srv_` prefix.

### What to do next

Once you have added the NAR files to a cloud storage location, you are ready to launch the Deployment Wizard and deploy a flow.

## Python scripts

Relying on Python scripts to perform data transformations within data flows is a common pattern for NiFi users. Cloudera Data Flow flow deployments come with Python 3 and the following custom pre-installed packages: `requests`, `urllib3`. You can design your data flows to use the pre-installed Python runtime as well as install additional custom packages which you might require.

### Upload and run Python scripts in flow deployments

If running your data flow requires executing a Python script, you have to upload it when creating your data flow deployment through the Deployment Wizard or the CLI. Follow these steps to configure your NiFi processors correctly and upload your Python script.

## Procedure

1. Create your Python script and save it as a file.

For example:

```
#!/usr/bin/python3
print("Hello, World!")
```

2. Open the flow definition which requires a Python script in NiFi.
3. Add and configure an ExecuteStreamCommand processor to run your script.

Make the following property settings:

### Command Arguments

provide #{Script}

### Command Path

provide python

Leave all other properties with their default values.



**Note:** If you need to upload additional supporting files for use by your script, add a dynamic property named Additional Resources referencing a parameter #{AdditionalResources}. The primary script may reference these files through the path /nifi-flow-assets/[*\*\*\*PARAMETER CONTEXT NAME\*\*\**]/[*\*\*\*PARAMETER NAME\*\*\**]/[*\*\*\*FILE NAME\*\*\**].@f0.

4. If you have edited your data flow in NiFi, download it as a flow definition and import it to Cloudera Data Flow. If you have edited your data flow in the Flow Designer, publish the flow to the Catalog.
5. Initiate a flow deployment from the Catalog. In the Parameters step of the Deployment Wizard, upload your Python script to the Script parameter. Upload additional files to the AdditionalResources parameter if applicable. Complete the Wizard and submit your deployment request.

## Results

Your Python script is uploaded to the flow deployment and executed as part of the data flow.

## Related Information

[Add a parameter in Flow Designer](#)

[Publish a draft to Catalog as a flow definition](#)

[Download a flow definition from NiFi](#)

[Import a flow definition to Cloudera Data Flow](#)

[Deploy a flow definition in Cloudera Data Flow](#)

## Install custom Python libraries in flow deployments

If your data flow requires custom Python packages you can modify your Python script to install these dependencies through the use of NiFi processors.

## Procedure

1. Create a Python script, to install the package you want to add:

```
#!/usr/bin/python3
try: import [***PACKAGE NAME***] as [***IMPORT AS***]
except ImportError:
    from pip._internal import main as pip
    pip(['install', '--user', '[***PACKAGE NAME***]'])
    import [***PACKAGE NAME***] as [***IMPORT AS***]
import sys
```

```
file = [***IMPORT AS***].read_csv(sys.stdin)
```

Replace `[***PACKAGE NAME***]` with the name of the package you want to import and `[***IMPORT AS***]` with a meaningful name you want the package to be called in your data flow.

```
#!/usr/bin/python3
try: import pandas as pd
except ImportError:
    from pip._internal import main as pip
    pip(['install', '--user', 'pandas'])
    import pandas as pd
import sys
file = pd.read_csv(sys.stdin)
```

2. Open the flow definition which requires custom packages in NiFi.
3. Add and configure an ExecuteStreamCommand processor to run your script.

Make the following property settings:

#### Command Arguments

provide `#{Script}`

#### Command Path

provide `python`

Leave all other properties with their default values.



**Note:** If you need to upload additional supporting files for use by your script, add a dynamic property named `Additional Resources` referencing a parameter `#{AdditionalResources}`. The primary script may reference these files through the path `/nifi-flow-assets/[***PARAMETER CONTEXT NAME***]/[***PARAMETER NAME***]/[***FILE NAME***]@f0`.

4. If you have edited your data flow in NiFi, download it as a flow definition and import it to Cloudera Data Flow. If you have edited your data flow in the Flow Designer, publish the flow to the Catalog.
5. Initiate a flow deployment from the Catalog. In the Parameters step of the Deployment Wizard, upload your Python script to the `Script` parameter. Upload additional files to the `AdditionalResources` parameter if applicable. Complete the Wizard and submit your deployment request.

## Results

Your Python script is uploaded to the flow deployment and the required custom libraries are installed when the script is executed as part of the data flow.