

Connecting to an Inbound Connection Endpoint

Date published: 2021-04-06

Date modified: 2024-06-03

The Cloudera logo, consisting of the word "CLOUDERA" in a bold, orange, sans-serif font. The letter "E" is stylized with a horizontal bar through its center.

Legal Notice

© Cloudera Inc. 2024. 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.

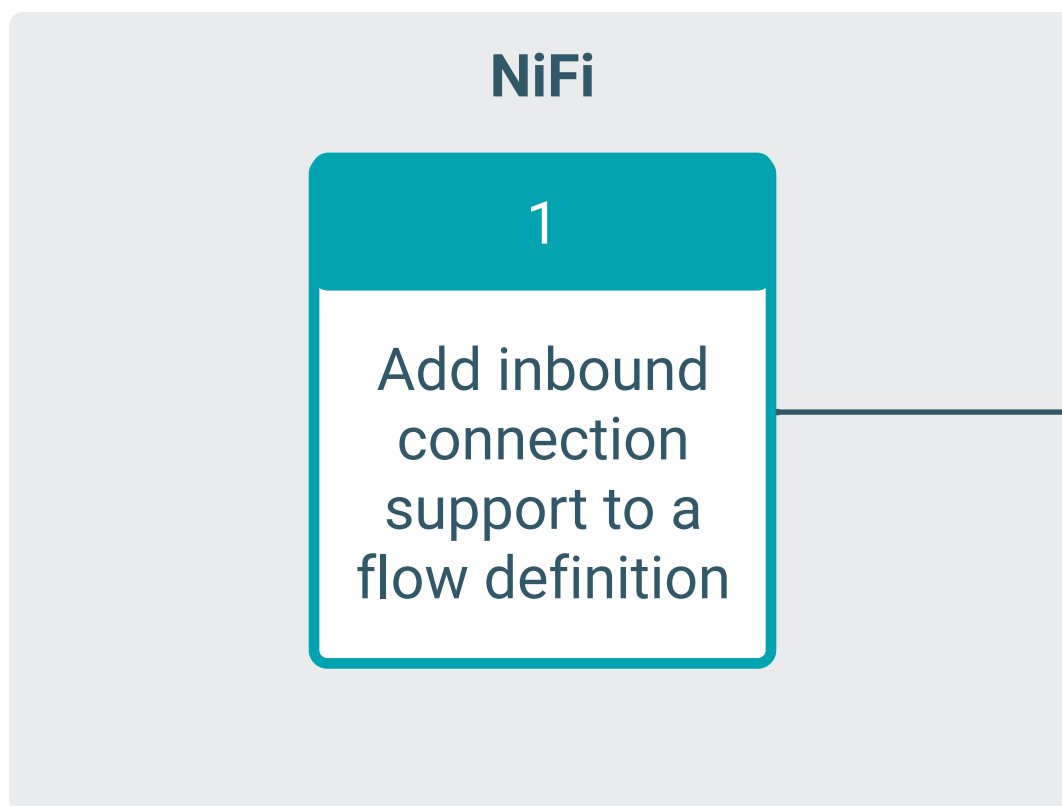
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES 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

Connecting applications to an endpoint.....	4
TLS keys and certificates.....	5
Tutorial: MiNiFi to Cloudera DataFlow flow deployment.....	5
Tutorial: Invoking an HTTP endpoint with curl.....	16
Using Inbound Connections with an external load balancer.....	21
Configure an Application Gateway in Azure.....	23

Connecting applications to an endpoint

Once a Cloudera DataFlow 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.

Tutorial: MiNiFi to Cloudera DataFlow flow deployment

This tutorial walks you through creating an inbound connection endpoint in Cloudera DataFlow 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 DataFlow environment.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Cloudera DataFlow 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 DataFlow.



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 DataFlow 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 DataFlow 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	? No value set ↑
Client Certificate	? /opt/minifi/minifi-test/client-certs/client-certificate-encoded... ↑
Passphrase	? No value set ↑
Private Key	? /opt/minifi/minifi-test/client-certs/client-private-key-encoded ↑
Use System Cert Store	? false ↑

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 DataFlow 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)' processor, which is circled in blue. 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 'Timer Driven' strategy and '1' concurrent tasks), 'Run Schedule*' (set to '1000 ms'), and 'Run Duration*' (a slider from 0ms to 2s). Below these is a 'Properties' table with various settings like 'Always Output Response' (false), 'Attributes to Send' (No value set), 'Connection Timeout' (5 s), 'Content-type' (application/plain-text), 'Disable Peer Verification' (false), 'Follow Redirects' (true), and 'HTTP Method' (POST). An 'Apply' button is at the bottom right.

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***]

Configuration

Settings

Service Name*

Client SSL Context Service

Properties

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

Apply

- 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.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 DataFlow 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

SERVICES

PARAMETERS

»

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

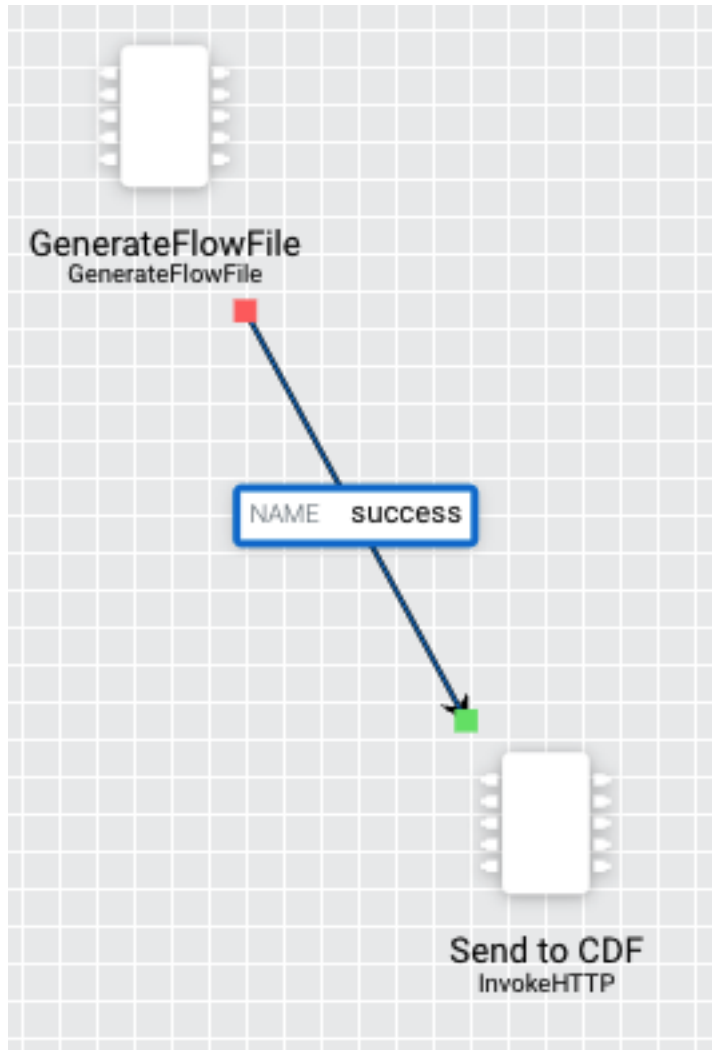
Run Duration*

0ms25ms50ms100ms250ms500ms1s2s

Properties

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

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 DataFlow Dashboard and click KPIs.

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

Related Information

[Configure an external application as a client to an Inbound Connection Endpoint](#)

[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 DataFlow environment.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Cloudera DataFlow 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}'
```

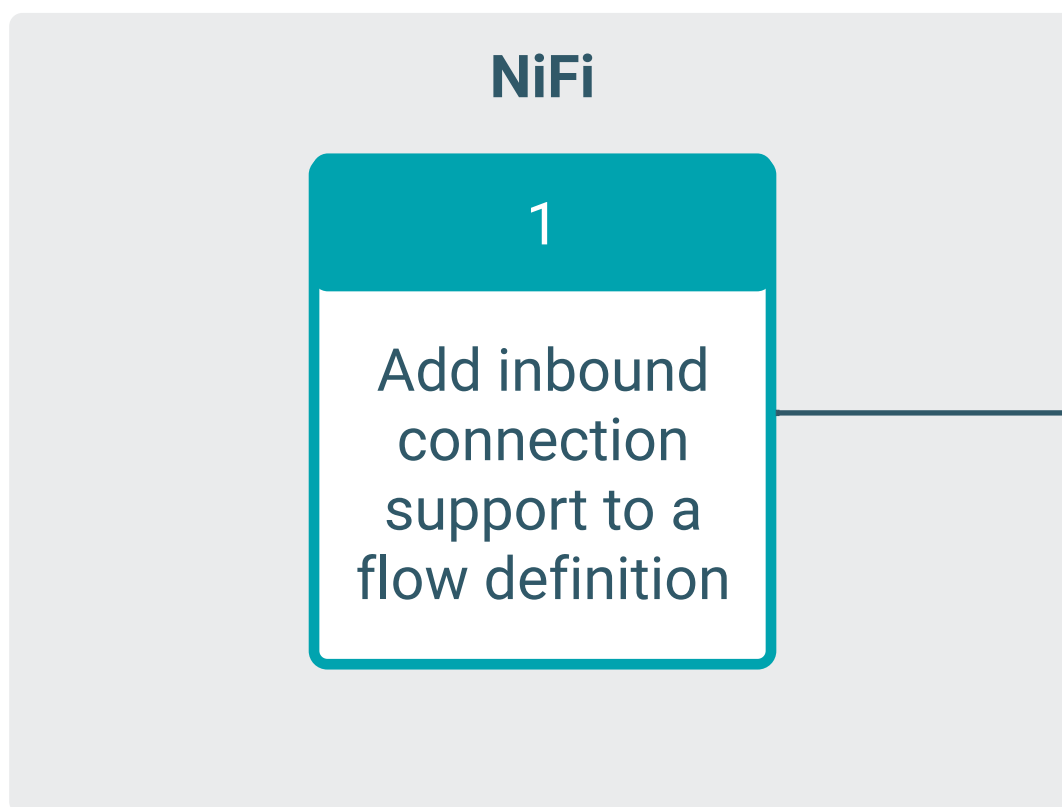
The NiFi deployment tries to validate the schema against the schema name and Schema Registry that you provided when deploying the ReadyFlow. If you provided dummy values, you receive a response indicating that the flow was unable to look up the schema.

Related Information

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

Using Inbound Connections with an external load balancer

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

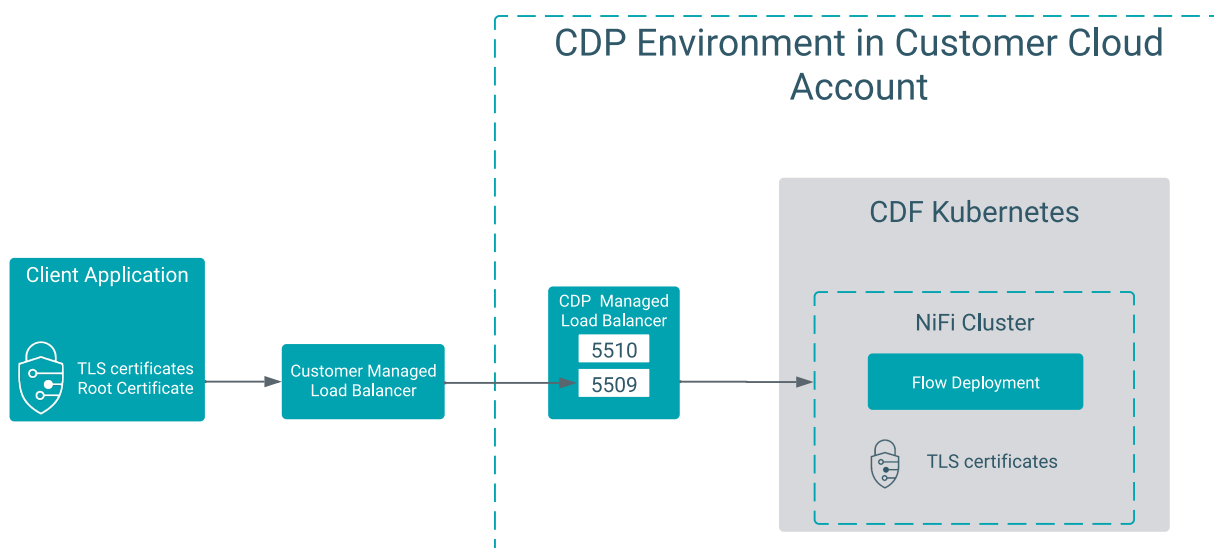


Inbound Connection Endpoints are created in Cloudera DataFlow with an internal Layer 4 (L4) load balancer (LB). Nevertheless, it is also possible to use your own native Layer 7 (L7) LB (Application Gateway on Azure, Application Load Balancer on AWS, respectively) in front of the Cloudera managed L4 LB.

Cloudera recommends achieving this by configuring your L7 LB to use the Cloudera DataFlow deployment LB as a backend. Enabling TLS between your LB and the Cloudera DataFlow 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 DataFlow for your environment, which limits Cloudera DataFlow 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 DataFlow 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 DataFlow 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](#)