Data Fabric – Internal Agent Guide

# Prerequisites

Before you can create an agent, you must have an existing environment (or tenant) to create the agent under.

Agents are created and configured within the Integration Portal, where the Data Fabric web resides.

There are three versions of the Integration Portal available:

Development/vNext – <https://integration.labs.epicor.com>

Canary/Prerelease – <https://canary.integration.epicor.com>

Production – <https://integration.epicor.com>

To start, login to your required version of the Integration Portal through Identity Provider (see the links above). Ensure that you have been setup within the Integration site before proceeding onto the next step.

# Launch Data Fabric

Once you are logged into the Integration Portal launch Data Fabric from the “Integrations” tab.

A screenshot of a computer

Description automatically generated

# Configure Agent

From the Data Fabric front-end, go to the “Agents” tab, click the + icon in the top right-hand corner to create a new agent.

A screenshot of a computer

Description automatically generated

Give your new agent a name, if this is for development purposes and only accessible to you then use your name as an identifier. Next choose the agent type, currently there are different agent types:

* Cloud – A SaaS managed offering of the agent, running in a docker container. This deployment requires some additional configuration not covered by this documentation and might not be visible depending on permissions granted.
* On-premise – Agent running in a local deployment, either as a service or standalone self-contained. Despite the name “on-premise” this does not limit you from running purely on-premise, it can be ran from anywhere, for example a VM in Azure.
* HTTP – An HTTP type agent can be used to send events via a REST API - a webhook can be posted to. This type only supports sending and not receiving events. This type of agent is intended for advanced integration scenarios and is not covered by this document.

For the purposes of this guide, we will select “On Premise” as the agent type.

Create connection(s) for your agent, again, for development purposes it is likely that you will only have one connection if you are implementing one side of an integration.

A screenshot of a computer

Description automatically generated

Currently, agents support the following three Connector types:

1. SQL
2. Mongo DB
3. File System

A screenshot of a computer

Description automatically generated

*If your required connector is not supported, please raise this with the Labs team so that we can extend support for it.*

## SQL Connector

### How it works

The SQL connector is designed to interact with 5 tables in MS SQL database, these tables are used to store the incoming and outgoing events as well as options and configuration.

Data Fabric will send events based on the agent’s configured routes, and the agent SQL connector will insert these events into the inbox table. From this point on, the system and its processor are responsible for consuming and handling these events.

Once an event has been processed, the status should be marked as processed, and the processed time updated. If there is an error processing the event, it should be marked as an error and a log inserted into the IntegrationInboxLog table including the InboxID of the related event.

Outbox processing is done in reverse, whereby the system inserts a row in the IntegrationOutbox for an event outbound. The agent polls, checking for new events to process and consume. Once events have been found, internally it will assign them to a consumer and they will be sent onto their destination through Data Fabric.

If there are any errors processing events the status will be marked with the error status, and a log inserted into the IntegrationOutboxLog table. This could occur when the event is validated against its corresponding schema, within the schema repository, accessible through the "Event Types" section of the user interface. When this happens, these logs will contain verbose information on why the event could not be sent.

### Usage

When using the SQL Connector, a valid SQL connection string must be supplied. Note that the SQL connection does not need to be exposed publicly to the internet, since typically the agent will be deployed next to the SQL server – meaning it might be a local hostname to your domain. The agent enforces encrypted connections, so ensure you have a valid certificate on the SQL Server. A full list of required parameters has been provided in the table below:

A screenshot of a computer

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Default | Description |
| Name | Yes |  | Distinguishable name for the connection. If the agent only has one connection, you may choose to use the same name as the agent. |
| Max Retries | Yes | 3 | Number of attempts to try to re-process the same outbox event (on failure). |
| Connection String | Yes |  | SQL Connection string. This connection string must include authentication information either in the form of SQL server credentials or as IntegratedSecurity. Please note, that TrustServerCertificate setting is also required to ensure correct operation of SSL connections |
| Default Schema | Yes | dbo | Schema to expect the tables to exist in. Depending on the system, this might not be the default. For example, in Kinetic this is “Inh”. |
| Purge Interval Seconds | Yes | 604800 (7 days) | How long processed events should be left before purging (in seconds) |
| Purge Inbox | No | False | Whether inbox events should be purged from the inbox collection. Note the default here is false because Data Fabric processors are usually responsible for purging of the inbox table. |
| Purge Outbox | No | True | Whether outbox events should be purged from the outbox collection. |
| Purge Batch Limit | Yes | 100 | How many events to try to purge in one batch. |
| Batch Size | Yes | 100 | Number of events to process in a single batch. Note that this number can be increased allowing for greater throughput at the expense of load. |
| Process Interval Seconds | Yes | 5 | Number of seconds to wait before polling for new outbox events. |
| Scopes | No |  | List of scopes to limit a connection to. When provided, routes can only be created when they include one of these defined scopes. |

## Mongo DB Connector

### How it works

The connector for Mongo DB works in a similar fashion to its SQL counterpart with the main difference being that collections are used in place of equivalent tables. The IDs that make up the primary key are GUIDs instead of numeric, which means that events are processed in order of their created date/time instead of the ID (as well the priority). This connector is currently only used by Connected Process Control since its database layer is using Mongo DB.

### Usage

A Mongo DB connection string must be supplied, with valid credentials. A full list of required parameters has been provided in the table below:

A screenshot of a computer

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Default | Description |
| Name | Yes |  | Distinguishable name for the connection. If the agent only has one connection, you may choose to use the same name as the agent. |
| Connection String | Yes |  | Mongo DB Connection string, these strings start with “mongodb://” you can find more information from <https://www.mongodb.com/docs/manual/reference/connection-string/>. |
| Database Name | Yes |  | Mongo Database to utilize. |
| Inbox Collection Name | Yes | IntegrationInbox | Name of the collection to use for Inbox events. |
| Inbox Log Collection Name | Yes | IntegrationInboxLog | Name of the collection to use for Inbox logs. |
| Outbox Collection Name | Yes | IntegrationOutbox | Name of the collection to use for Outbox events. |
| Outbox Log Collection Name | Yes | IntegrationOutboxLog | Name of the collection to use for Outbox logs. |
| Max Retries | Yes | 3 | Number of attempts to try to re-process the same outbox event (on failure). |
| Purge Interval Seconds | Yes | 604800 (7 days) | How long processed events should be left before purging (in seconds) |
| Purge Inbox | No | True | Whether inbox events should be purged from the inbox collection. |
| Purge Outbox | No | True | Whether outbox events should be purged from the outbox collection. |
| Purge Batch Limit | Yes | 100 | How many events to try to purge in one batch. |
| Batch Size | Yes | 100 | Number of events to process in a single batch. Note that this number can be increased allowing for greater throughput at the expense of load. |
| Process Interval Seconds | Yes | 5 | Number of seconds to wait before polling for new outbox events. |
| Server Selection Timeout Seconds | Yes | 15 | How long to wait before considering Mongo DB connection timed out. |
| Scopes | No |  | List of scopes to limit a connection to. When provided, routes can only be created when they include one of these defined scopes. |

## File System Connector

### How it works

Instead of using tables, this connector works with directories instead. There are a total of seven directories:

* Inbox – Events will arrive here based on the routes configured in the agent. Events are stored as two separate files, the data and metadata files.
* InboxLogs – Logs can be created, related to specific inbox events.
* InboxProcessed – Once an event is processed by the system, it should be moved to this directory.
* Outbox – Events should be added to this directory to be sent out.
* OutboxError – Events will be moved here when the agent fails to process them.
* OutboxLogs – The agent will create logs here when there are any errors processing events.
* OutboxProcessed – The agent will move events here after they have processed successfully.

Below is an example of an event that has been received:A close-up of numbers

Description automatically generated

The event is made up of two files, a metadata JSON file and the event data, a separate file suffixed with “\_data” with no extension.

The metadata will always contain the properties below (the values will be different though), defining some of the properties required for the agent and processor to process the data correctly.

{

    "Status": 0,

    "Scope": "",

    "EventType": "Test\_1.0",

    "CreatedDateTime": "2023-05-05T16:10:35.4332187Z",

    "ProcessedDateTime": null,

    "Key": "1",

    "CorrelationId": "76eac385-f85a-406a-be95-d0ea6e41364f",

    "FileExtension": "",

    "Retries": 0,

    "Id": "d067bb06-26f9-4a80-bd06-dbd9e088f287",

    "Priority": 3

  }

The File System Connector is currently only in use by one implementation, allowing communication between the Agent and Epicor HQ for an EDI integration.

### Usage

A “directory path” should be provided, ensure the user the agent is running under has read and write access to this directory. If the directory does not exist, the agent will create it automatically on start-up.

A screenshot of a computer

Description automatically generated

It's possible to override some of the defaults for the folder names, however it is not possible to specify different absolute directories, everything will be under the same top-level path specified in “Directory Path”.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Default | Description |
| Name | Yes |  | Distinguishable name for the connection. If the agent only has one connection, you may choose to use the same name as the agent. |
| Directory Path | Yes |  | Path to create integration related directories under. |
| Inbox Folder Name | Yes | Inbox | Inbox events directory name |
| Inbox Processed Folder Name | Yes | InboxProcessed | Directory where received inbox events should be moved into after being processed by the system. |
| Inbox Log Folder Name | Yes | InboxLogs | Directory containing any logs that related to inbox events. |
| Outbox Folder Name | Yes | Outbox | Directory agent will process and send events out from. |
| Outbox Error Folder Name | Yes | OutboxError | Directory agent will move events into when it when it encounters an error processing them. |
| Max Retries | Yes | 3 | Number of attempts to try to re-process the same outbox event (on failure). |
| Batch Size | Yes | 100 | Number of events to process in a single batch. Note that this number can be increased allowing for greater throughput at the expense of load. |
| Process Interval Seconds | Yes | 5 | Number of seconds to wait before polling for new outbox events. |

Select the connector type for your integration purposes and provide the required parameters as listed in usage. Save your agent and you will be taken to the agent page.

# Copy registration key

Once the agent is registered, the agent’s key can be copied from the top of page. The registration key is a one-time key used to setup the physical installation with the agent’s configuration. Once registration has been done, the agent will be tied to its virtual representation in the front-end.

A screenshot of a computer

Description automatically generated

Copy the “registration key” to the clipboard, using the button next to “registration key”.

# Install the Agent

The next stage is to download the agent.

A screenshot of a computer

Description automatically generated

Select your platform and runtime, for example Windows and select the latest version.

For Windows, you have the option of either downloading a standalone agent or using the Windows installer.

The installer allows you to setup agents as Windows services using an additional management tool to configure them. Alternatively, if you are simply testing or developing the standalone agent can be downloaded and run instead.

## Setup CLI (Standalone)

Download the agent using the “Download Agent” button.

Extract the files to a directory, for example “C:\DataFabricAgent”

Launch "Epicor.DataFabric.Agent.exe", at this point it will ask you for the registration key:

A computer screen shot of a computer error

Description automatically generated

Paste the registration key in from the previous step and press enter. On successfully validating the registration key with Data Fabric the agent will be fully registered and should then start communicating with the API. Please be aware that agents cannot be used across deployments – so download production agents to use for production and download development/vNext agents for use with development version of Data Fabric.

Note also that the registration key does not need to be kept, once an agent is registered the key is no longer valid. If you require re-registering an agent, simply go back to the agent page and click “unregister” to generate a new registration key. Doing this will invalidate existing agent connection though.

## Setup Installer

As an alternative to running the agent as standalone, it can be deployed through MSI installer. To begin, download the agent using the “Download Windows Installer” button.

Once downloaded, run the installer.

A screenshot of a computer

Description automatically generated

You have the option to install with or without a Windows service.

A screenshot of a computer

Description automatically generated

Provide a name for the agent, this does not necessarily need to be the same as the agent’s name, this is just an identifier for the Windows Service that gets installed. Typically name will help identify what the agent is intended for – so this could be name of the ERP deployment, such as “Live”, “Pilot” or another identification of the intended purpose of this agent.

Enter the registration key from the step when you created the agent.

Optionally you can provide a Windows username and password. The service will then run under those credentials rather than the standard Windows local service account that is used if credentials are left blank.

*Note that if you skip this step, you can still always run the “manager” afterwards to create the service as descibed later in this document.*

Click confirm, then click next and install the agent.

Once installed, the service will run in the background. It can be stopped or deleted through the Data Fabric Manager tool.

# Using the Manager

Alongside the agent, the installer also includes a service management tool, that can be used to add agents as Windows services and manage them within the same installation. This tool can be used as either a CLI or in terminal mode. Note that this manager tool needs to be ran elevated, since it needs elevated access to create and control Windows services.

A computer screen shot of a blue screen

Description automatically generated

## Adding a new Agent

A screenshot of a computer

Description automatically generated

Add a new agent, which will be installed as a Windows service.

You must provide the following:

* Name - This name will be used as a reference, it does not need to be the same as the agent’s name and will be suffixed onto the service.
* Registration Key - The registration key for the agent.

Optionally you can also provide:

* Description – The description can be replaced, and instead you can provide your own. This is the description that will be displayed in Windows Services.
* Username - A specific Windows user to run the service under.
* Password – A password for the Windows user, required when username is supplied.

A screenshot of a computer

Description automatically generated

Once added, the registered agent will be visible in the list of agents:A computer screen shot of a blue screen

Description automatically generated

The “Status” as displayed above is only the current service status, rather than the health status reported and displayed within Data Fabric’s front-end.

## Other actions

The service can be stopped or started by toggling the status:  
A computer screen shot of a message

Description automatically generated

To delete an agent service, click on the “Delete” button next to the agent.

You can also control the status of all the agents from the buttons above. For further explanation on each action, click the “Help” button.

# Troubleshooting

The agent will continuously write Windows logs under application. You can review these logs through Windows event viewer when you run into any problems. The agent will also report its health and telemetry to the Data Fabric frontend that can be reviewed under the agent page.

A screen shot of a graph

Description automatically generated

# Routes

To receive events from another system, routes must be configured under an agent.

It’s important to note that events will always be processed by Data Fabric regardless of whether there is a route configured for them. This is because routes are not created by the producer (agent that reads outgoing events from the Outbox), but by the consumer(s) (agent(s) that receive events from producers and write them into the Inbox) – and the setup of each route might be different between each connection.

Routes comprise of a source, and destination connection and can be created within the agent page, or on another tab named “Routes”. The routes tab lists all agents for the current tenant, whereas the agent only lists routes that are destined for the specific agent being currently viewed.

## Creating a route

To create a new route, go to the Agent page in the Data Fabric frontend. Scroll down to the “Routes” panel card and click the + icon.

A screenshot of a computer

Description automatically generated

Depending on your browsers tab width, the route will add a new row in the table, or a dialog will pop up as displayed below:

A screenshot of a computer

Description automatically generated

Select, or enter a **source** connection. This is the connection you want to receive events from.

Provide a **source scope**, in some cases this might be blank or empty. Scopes are used as part of the key that makes up an event. This might be the company, site or tenant it was sent from.

A screenshot of a computer

Description automatically generated

Enter an **event type**, note this can be a specific event, or a filter, using a wildcard **\*** character.

For example, you could specify an event type of “Kinetic.Order\_1.0”, allowing you to consume only that specific event, or you could enter “Kinetic.\*” which would then allow you to receive events where the event type starts with “Kinetic.”.

Select a **destination** connection, note if you only have one connection it will be populated for you automatically.

Provide a **destination scope**, in some cases this might be blank or empty. Scopes are used as part of the key that makes up an event. This might be the company, site or tenant it is being sent to.

**Action** can be ignored, currently only “read” type routes are supported. This is present to enable future capabilities of Data Fabric.

Once you are happy with your route, click save. Note that any changes made to routes can take up to 5 minutes to come into effect.

Exact details of event types and scopes will be supplied with the documentation of each individual integration.