Cloudera Edge Management 1.5.0

Managing Agent Manifest Resolution

Date published: 2019-04-15
Date modified: 2023-02-15

CLOUDERA

Legal Notice

Contents

Introduction to agent manifest resolution strategy

The manifest of an agent is its full and detailed list of capabilities. A manifest includes all extension components, including processors and controller services, and how they are configured.

When an agent first communicates to EFM through the C2 Protocol (heartbeats), the EFM server asks the agent to describe itself, and the agent supplies its manifest to the server.

An agent class is a group of agents. Currently, agent classes are defined by a unique property in each agent, and agents with matching class names are put in the same agent class in the EFM server.

An agent manifest is necessary in order to design a flow for an agent class, but because an agent class can contain many agents, the EFM server must decide which agent manifest to use for the entire agent class when loading the flow for that class in the flow designer. The logic that decides this is referred to as the *Agent Manifest Resolution Strategy*.

EFM supports multiple strategies to resolve agent manifest for a class at run time:

Strategy	Description
First In	This is the default manifest resolution strategy and binds an agent class to the first manifest reported for it.
Last In	This strategy updates the associated manifest to the most recently reported by any agent associated with the specified class.
Static	This strategy allows mapping of an agent class to a specific manifest ID.

The strategy is configurable at the application level and applies to all agent classes.

In efm.properties, set the global agent manifest resolution strategy:

```
# This property does not exist, so you will have to add it anywhere in efm.p
roperties
# The default is 'First In'
efm.manifest.strategy=Last In
```

Static mappings act as overrides to the global manifest resolution strategy. If a static mapping of agent class to manifest ID is configured, then the application uses the static manifest strategy for that class and ignore the globally configured strategy. Classes without a static mapping falls back to the global stratey (i.e., First In or Last In).

Static mappings are created through the REST API (or Swagger UI) and stored in the EFM database.

Here is an example of setting a mapping for a class to manifest using curl:

```
# Get all agent manifests
curl -X GET "http://localhost:10090/efm/api/agent-manifests" -H "accept: a
pplication/json"

# Make a note of the manifest identifer to use in the mapping
# Create a mapping
curl -X POST "http://localhost:10090/efm/api/agent-class-manifest-config" -H
 "accept: application/json" --data { "agentClassName": "MyAgentClass", "agen
tManifestId": "27165b44-c54a-4504-8d47-7e3bec421a00" }

# Later, update to map the agent class to a new manifest
curl -X PUT "http://localhost:10090/efm/api/agent-class-manifest-config" -H
 "accept: application/json" --data { "agentClassName": "MyAgentClass", "agen
tManifestId": "90af998a-f7ff-4422-b8fb-2ed08f273959" }
```

See the Swagger UI Section *Agent class manifest config* for more details.

> **Note:** When importing a flow for an agent class, the application uses static strategy irrespective of the globally configured strategy. In this scenario, if there is an existing mapping of agent class to manifest configured and if the input manifest does not match with the configured one, then an illegal state exception (HTTP 409 Conflict) error is thrown. If there is no existing mapping configured, then a mapping of agent class to the input manifest ID will be persisted.

Refreshing the agent manifest for existing flows

After upgrading agents that already have a flow defined for their agent class, the changes in the manifest (version and processor updates) are not instantly reflected in the existing flows. To make these changes appear, you must refresh the manifest using the Refresh Manifest option available in the EFM Flow Designer.

Before you begin

- Unforeseen issues can arise when refreshing the agent manifest. As a result, Cloudera recommends that you export the original flow definition before you continue with this task. You can export the flow definition with the REST API using the GET /designer/{agentClassName}/flows/export operation and endpoint. For more information, see Exporting and importing dataflows in CEM.
- Update all agents belonging to the same agent class. Inconsistencies between agent configuration can lead to issues with flow execution.

Procedure

1. In EFM, go to ⚙ Design.

2. Select the flow that belongs to the updated class and click Open.

3. Click Actions Refresh Manifest… .

   If there are new manifest definitions available for the class, the latest is used. Additionally, the differences between the existing and new manifest definition is presented to you in a modal window. The modal window only

lists newly added or removed processors and controller services. Version differences are not highlighted. This is because version differences are applicable to all processors following an upgrade. Here is an example:

Refresh Manifest   ✕

⚠ Refreshing this flow will rebase it to the latest manifest for the class **manifest-test**.

Differences between actual and latest manifest:

Added Processors:

org.apache.nifi.minifi.processors.Comp1ProcNEW

Removed Processors:

org.apache.nifi.minifi.processors.Comp1Proc4

org.apache.nifi.minifi.processors.Comp1Proc3

Added ControllerServices:

org.apache.nifi.minifi.controllers.Comp1CtrlSvcNEW

Removed ControllerServices:

org.apache.nifi.minifi.controllers.Comp1CtrlSvc2

Cancel   Refresh

4. Review the changes and click Refresh.

   The manifest is automatically refreshed after you click Refresh. All existing processors and services are updated in the flow. Any new processors and services become available in the list.

5. Update component parameters if there were any changes.

   Component parameter changes, for example, name changes, introduction of new parameters, and so on, must be corrected before publishing.

6. Publish the updated flow.

Results
The agent manifest for the selected flow is refreshed. New agent capabilities are reflected in the manifest.