KIP-494: Connect REST Endpoint to get Transformations (SMTs)

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: *here*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Connect has a REST API where clients can easily ask for which connectors are loaded by the Worker. SMTs are popular feature, and it would be great to have a programmatic way to ask the worker which SMTs it has loaded.

Currently the best way to verify which SMTs the worker has loaded is to visually parse logs. This sometimes impossible for a long-running worker, and tedious otherwise. It also prevents programmatic querying and verification that SMTs loaded.

Existing endpoint to get Connectors is actually called "GET /connector-pluings", implying it should be used for all plugins. In reality, Connectors are just one type of plugin: Other types include Simple Message Transformations (SMTs), Converters, HeaderConverters, ConfigProviders, and RestExtensions (loaded by the "DelegatingClassLoader").

This KIP will add HTTP endpoints for Transformations, Converters, HeaderConverters, ConfigProviders, and ConnectRestExtensions to Connect RestServers.

Public Interfaces

The new endpoints are specified below in bold in OpenApi format.

spec.yaml

```
# This already exists in production
/connector-plugins:
  get:
    # Existing behavior, will not repeat it here
    summary: Lists available Connector plugins

# Newly Proposed Endpoints
/plugins:
  get:
    summary: Returns plugin descriptions of all types
    responses:
      '200':
        description: A list of plugins
        content:
          application/json:
            schema:
              # Consistent with existing "/connector-plugins" endpoint
              type: array
```

```
              items:
                type: object
                properties:
                  class:
                    type: string
                    description: Plugin class name (e.g. "io.confluent.connect.hdfs.HdfsSinkConnector")
                  type:
                    type: string
                    description: The type of the plugin ("source"|"sink" for connectors, and converter, header-
converter, transformation, config-provider, connect-rest-extension for other types)
                  version:
                    type: string
                    description: Version of the connector

/plugins/{plugin-type}:
  get:
    summary: Returns plugin descriptions of the given type
    parameters:
      -in: path
      name: plugin-type
      schema:
        type: string
        required: true
        description: plugin-type can be one of connector, converter, header-converter, transformation, config-
provider, connect-rest-extension
    responses:
      '200':
        description: A list of plugins
        content:
          application/json:
            schema:
              # Consistent with existing "/connector-plugins" endpoint
              type: array
              items:
                type: object
                properties:
                  class:
                    type: string
                    description: Plugin class name (e.g. "io.confluent.connect.hdfs.HdfsSinkConnector")
                  type:
                    type: string
                    description: The type of the plugin ("source"|"sink" for connectors, and converter, header-
converter, transformation, config-provider, connect-rest-extension for other types)
                  version:
                    type: string
                    description: Version of the connector
      '400':
        description: Invalid request, such as unknown plugin type, as constructed with existing
ConnectRestException
        content:
          application/json:
            schema:
              # Consistent with https://docs.confluent.io/platform/current/connect/references/restapi.
html#status-and-errors
              type: object
              properties:
                error_code:
                  type: integer
                  description: HTTP Error code (e.g. 400).
                message:
                  type: string
                  description: Human readable message to suggest action to caller.
```

For example, clients can query SMTs with:

```
GET /plugins/transformation
200 OK
[
  {
    "class": "org.apache.kafka.connect.transforms.MaskField"
    "type": "transformation"
    "version": "1.0"
  },
  {
    "class": "org.apache.kafka.connect.transforms.HoistField"
    "type": "transformation"
    "version": "2.0"
  }
]
```

Proposed Changes

The new endpoints are described in the API Spec above.

The proposal is to implement them on a new "PluginsResource" analogous to the existing "ConnectorPluginsResource", and using the ConnectorPluginsResource's access to a Herder instance, by using Herder.plugins(), analogous to existing ConnectorPluginsResource. getConnectorPlugins().

Compatibility, Deprecation, and Migration Plan

This is a simple addition, so no compatibility or migration issues. Existing APIs are intentionally left unaltered for simplicity.

We will need to update the docs.

Forwards Compatibility

Future plugins will be expected to be reported in the generic `/plugins` endpoint. It would also be expected that the plugin-type be reported via their own `/plugins/{plugin-type}` endpoints.

Deprecation of existing endpoint GET /connector-plugins

It is further proposed that the existing endpoint `GET /connector-plugins` is deprecated. While there is no hurry to remove it from the codebase, the current `GET /connector-plugins` endpoint will be redundant to this new, more generally useful endpoint. Marking deprecation with the introduction of the new feature will give clients as much time as possible to migrate, giving the maintainers more flexibility to remove the old endpoint when they deem necessary.

Rejected Alternatives

Limiting scope of KIP to just SMTs

The scope could be limited to just adding HTTP Endpoints for just SMTs, which have the most pressing need. However, any design extending the current API to include SMTs would be designed to be elegantly extensible, and then most of the work to just include the other Plugin types is already done.

Updating existing endpoint for "GET /connector-plugins/" to include all plugins, as its name suggests

This strategy would break compatibility, and there are suitable alternatives without breaking changes.

Add implementation on existing ConnectorsPluginsResource

This idea is good, too, but leaned towards a new class to keep the naming very precise.