LOGPOINT

Integrations

Universal REST API for DirectorConsole UI

*V*3.3.0

CONTENTS

CHAPTER

ONE

UNIVERSAL REST API FETCHER FOR DIRECTOR CONSOLE UI

Universal REST API Fetcher provides a generic interface to fetch logs from cloud sources via REST APIs. The cloud sources can have multiple endpoints, and every configured source consumes one device license.

CHAPTER

TWO

INSTALLING UNIVERSAL REST API FETCHER

Prerequisites

1. Director Fabric v2.6.0 or later
2. Director Console v2.6.0 or later
3. Logpoint v7.5.0 or later

To install Universal REST API Fetcher in Director Console:

1. Log in to Director Console.
2. Click **ASSETS** in the navigation bar.
3. Select **Plugins** from the **Assets Type** drop-down.
4. Upload the Universal REST API Fetcher .pak file. Select **Replace Existing?** to replace the old file with new one.
5. Click **UPLOAD**.

Once uploaded, the .pak file is displayed in the list of available packages.

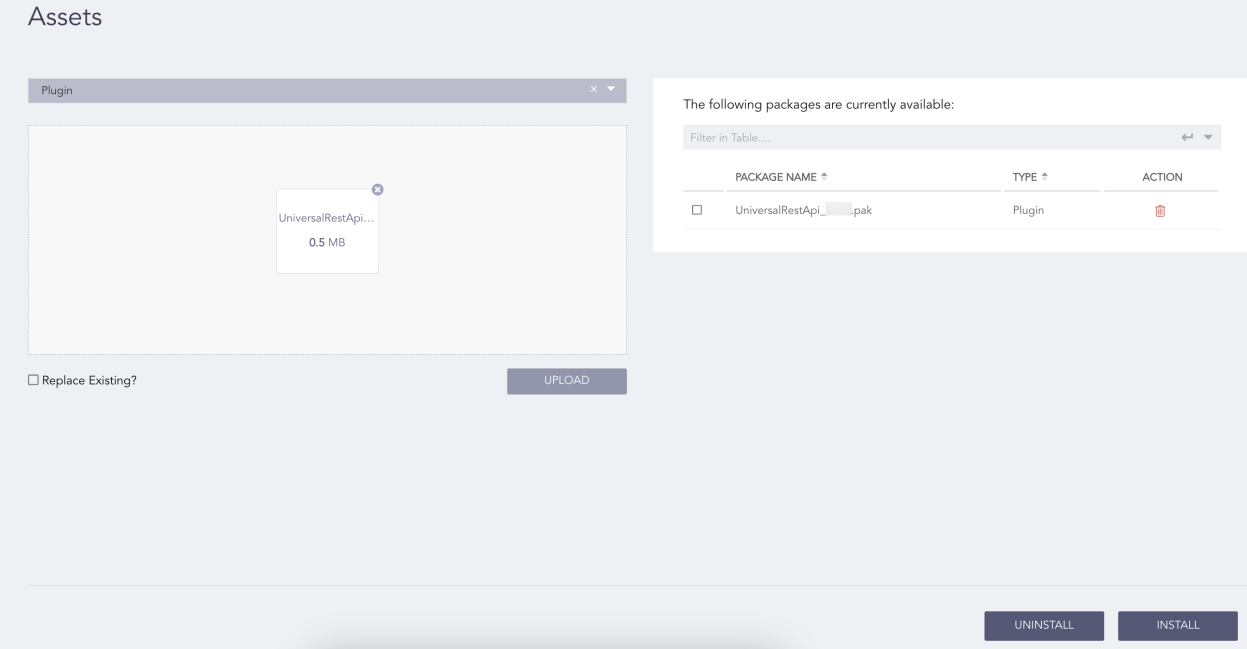

Fig. 1: Universal REST API Fetcher .pak file upload

6. Select Universal REST API Fetcher .pak from the list of available packages.
7. Click **INSTALL**.
8. Select the pool and Logpoint to install Universal REST API Fetcher. You can select multiple Logpoint from different pools.
9. Click **NEXT**.
10. Review your changes. You can go **BACK** to make necessary changes.

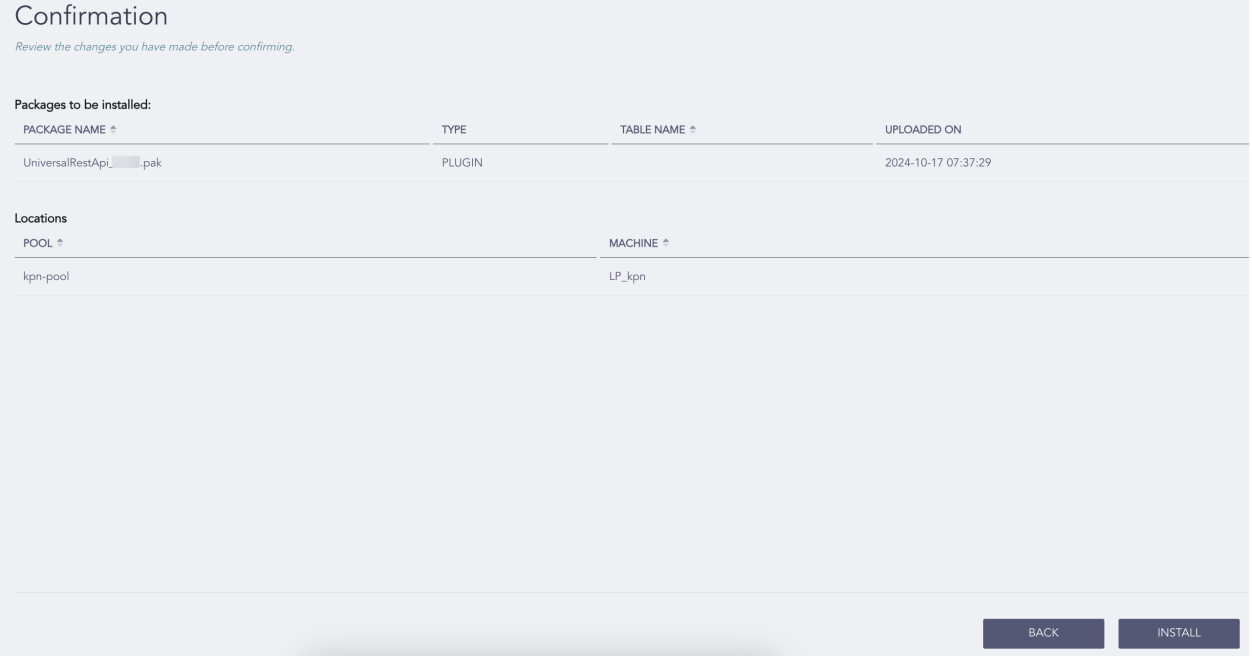

Fig. 2: CiscoUmbrella Installation Confirmation

11. Click **INSTALL** and click **OK** to confirm.

You are redirected to **TASKS**, which displays the installation progress.

CHAPTER

THREE

UNINSTALLING UNIVERSAL REST API FETCHER

To uninstall Universal REST API Fetcher, you must first remove its configurations and also uninstall Duo Security and Cybereason.

To remove Universal REST API Fetcher configurations:

1. Click **CONFIGURE** in the navigation bar.
2. Under **Entities**, click **LOG SOURCES**.
3. Click the (⋮) icon of Universal REST API Fetcher and click **Delete**.
4. Click **Delete**.

To uninstall Universal REST API Fetcher, Duo Security and Cybereason:

1. Click **ASSETS** in the navigation bar.
2. Click **UNINSTALL**.
3. Select the Logpoint where Universal REST API Fetcher, Duo Security, and Cybereason are installed. You can select multiple Logpoint from different pools.
4. Select **Universal REST API Fetcher, Duo Security and Cybereason** from the list of available packages.
5. Click **NEXT**.

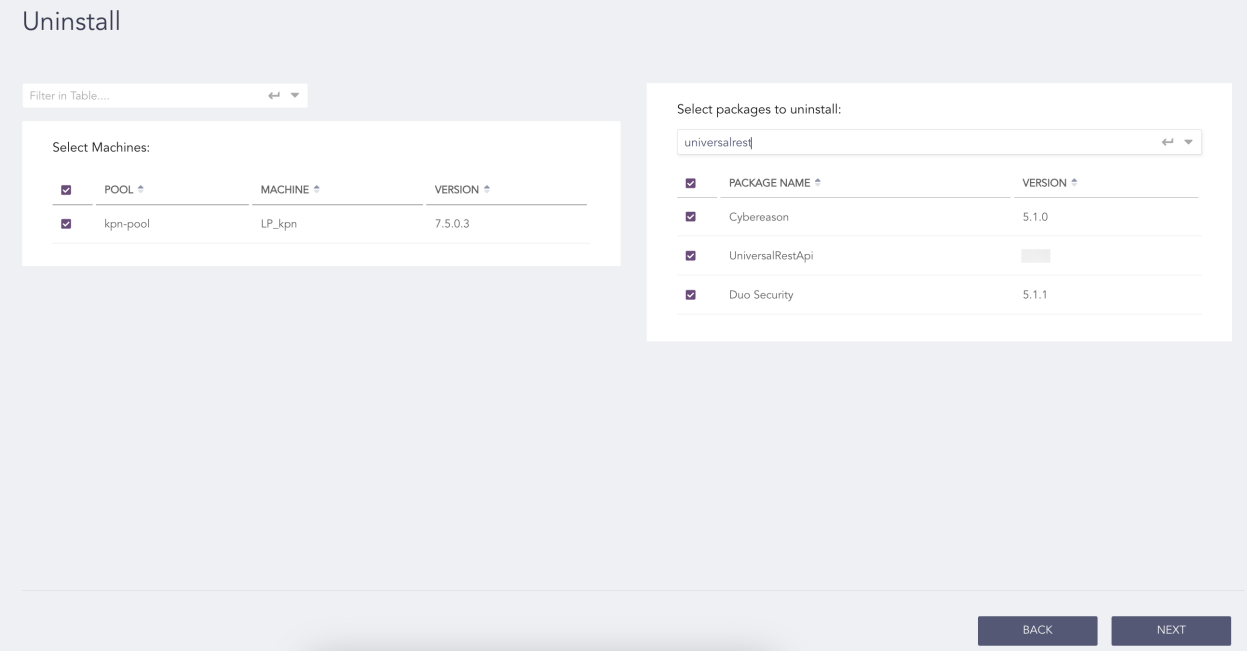

Fig. 1: Selecting Package

6. Review your changes. You can go **BACK** to make necessary changes.
7. Click **UNINSTALL** and click **OK** to confirm.

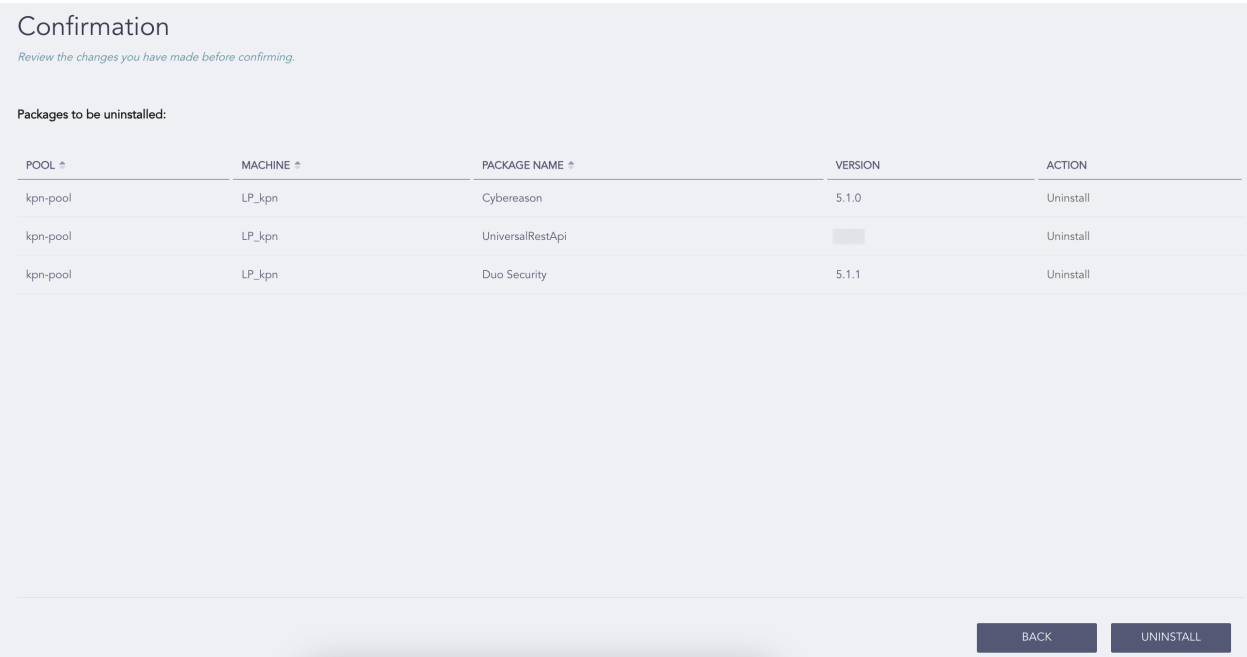

Fig. 2: Confirming the Changes

You are redirected to **TASKS**, which displays the uninstall progress.

CHAPTER

FOUR

CONFIGURING UNIVERSAL REST API FETCHER

You can configure Universal REST API Fetcher using its log source template. The template has pre-defined settings and configurations to fetch logs. However, some fields must be entered manually.

1. Click **CONFIGURE** in the navigation bar.
2. Under **Entities**, click **LOG SOURCES**.
3. Click **Create Log Source**.
4. Select **Universal REST API Fetcher**.
5. Select a Pool and Logpoint to configure the fetcher.

4.1 Source

In source, you can add details about the *REST APIs*, from where the Universal REST API Fetcher fetches logs for accurate identification, data formatting, and timestamping.

1. Click **Source**.
2. Enter the **Name**.
3. In **Base URL**, enter the API URL.
4. Enter **Request Timeout (secs)** for the API request.
5. In **Retry After(secs)**, enter the time to wait after an error or timeout.
6. Enter the frequency at which data is retrieved in **Fetch Interval (min)**.
7. Select the **Charset** and **Timezone**.

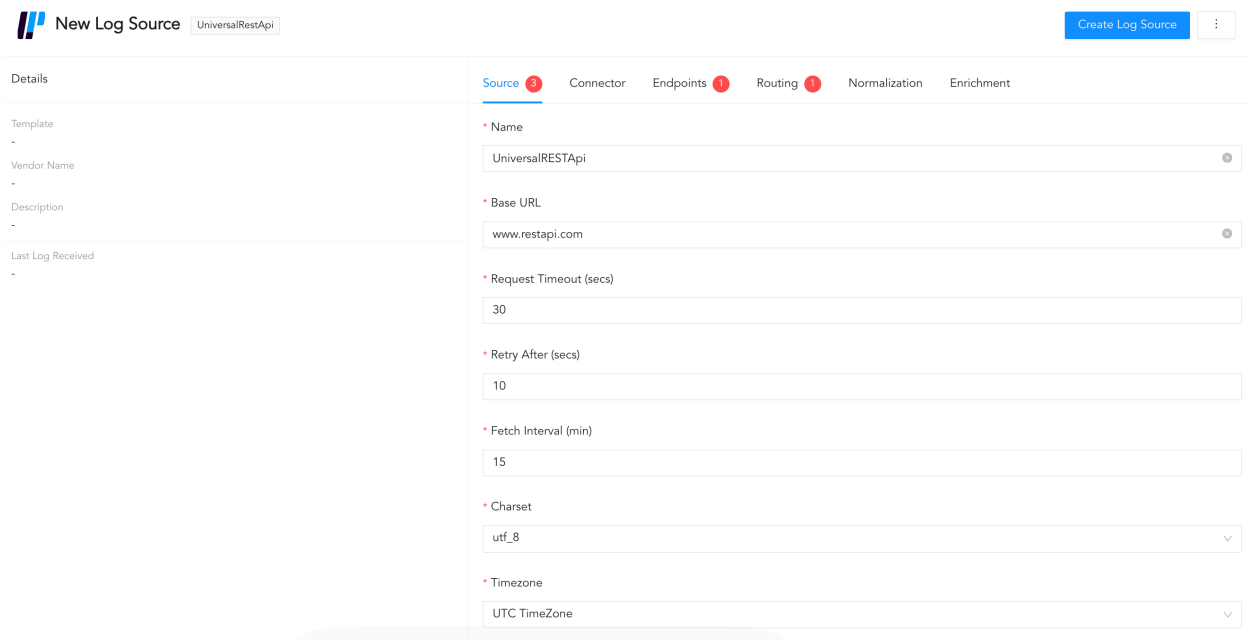

Fig. 1: Configuring Source

4.2 Connector

In connector, you can configure how Universal REST API Fetcher and *REST APIs* communicate with each other.

1. Click **Connector**.
2. Select the **Product**. It displays the integrations supported by Universal REST API Fetcher.
3. Select the **Authorization Type**.
    3.1. Select **No Auth** if no authentication is required.
    3.2. Select **Basic** to use a username and password to authenticate.
        3.2.1. Under **CREDENTIALS**, enter the **Username** and **Password**.
    3.3. Select **OAuth2** to authenticate using OAuth authentication.
        3.3.1. Under **OAUTH 2.0 BASIC INFORMATION**, enter the **Token URL** of the server.
        3.3.2. Select either **Client Credentials** or **Password Credentials** as the **Grant Type**.
            3.3.2.1. If you select **Client Credentials**, enter the OAuth secret password in **Client Secret**.
            3.3.2.2. If you select **Password Credentials**, enter the OAuth **Username** and **Password**.

3.3.3. In **Client ID**, enter the OAuth application ID or client ID.
3.3.4. In **API Key Prefix**, enter the prefix to add to the authorization header before the API Key or Token.
3.3.5. In **Client AUthentication**, select whether to send the client credentials as a basic auth header or in the body.
3.3.6. Enter the extra parameters key and its value in **ADDITIONAL BODY FOR OAUTH 2.0**.

3.4. Select the **API Key** to authenticate using an API Key.

3.4.1. In **Secret Key**, enter **API Key**. This API key is used in the authorization header.
3.4.2. In **API Key Prefix**, enter the prefix to add to the authorization header before API Key or Token.

3.5. Select **Digest** to authenticate using digest access authentication.

3.5.1. Under **CREDENTIALS**, enter the **Username** and **Password**.

3.6. Select **Custom** to authenticate using integration that applies custom authentication mechanisms and request handling.

4. Enter the custom headers in **Headers** as a key-value pair.
5. Enable **Enforce HTTPS certificate verification** to enable a secure connection.
6. Select **Enable Proxy** to use a proxy server.

6.1. Select either **HTTP** or **HTTPS** protocol.
6.2. Enter the proxy server **IP** address and the **PORT** number.

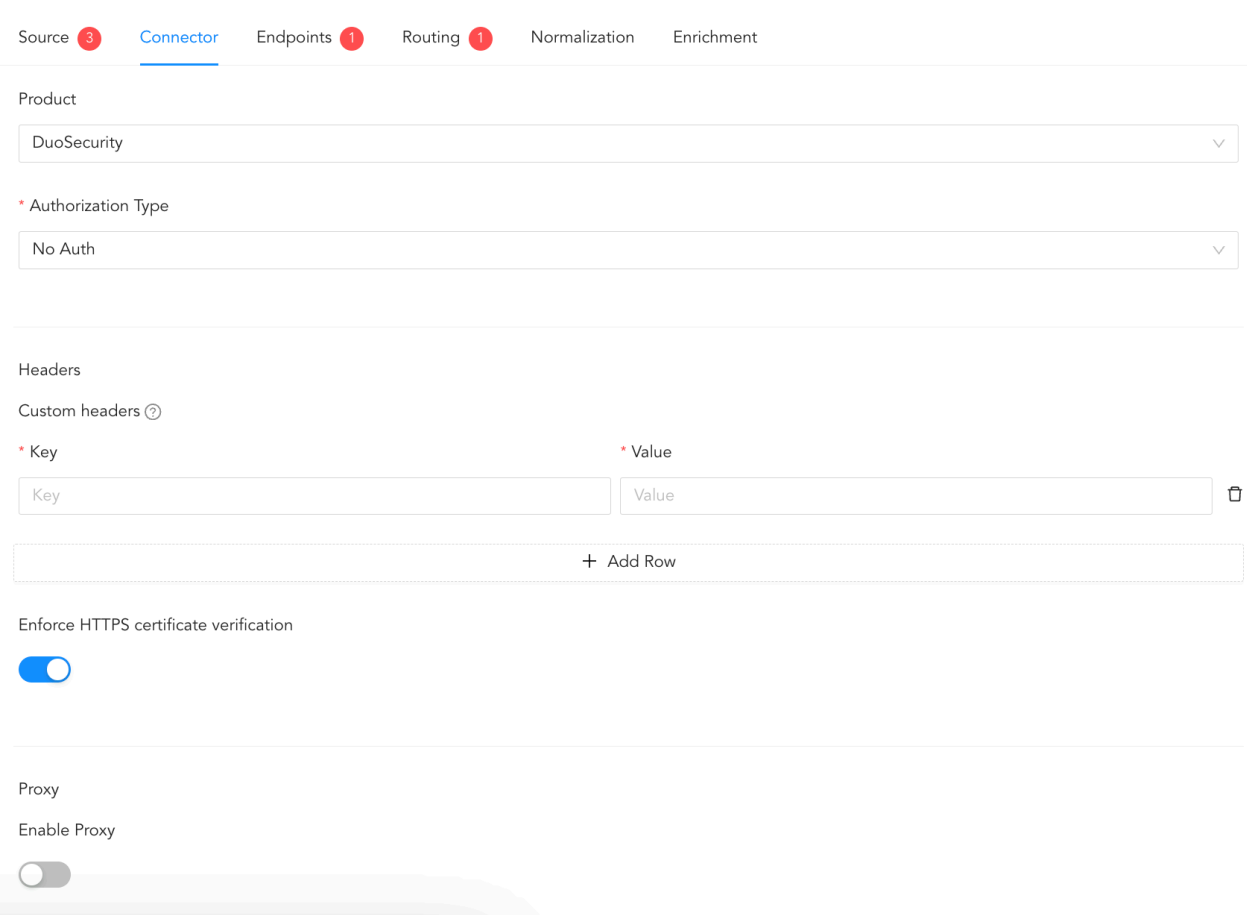

Fig. 2: Configuring Connector

4.3 Endpoints

In endpoints, you can configure details about the REST APIs endpoints.

1. Click **Endpoints** and **+ Add Row**.
2. Enter the endpoint's **Name**.
3. Select the request **Method** to call the endpoint.

    3.1. If you select **POST**, enter the **Post request body** in JSON format. For example:

```
{ "filters": [
        {
          "fieldName":"<field>",
          "operator": "<operator>",
          "values": "[value]",
```

```
    }
      ], "search": "<value>",
      "sortingFieldName": "<field>",
      "sortDirection": "<sort direction>",
      "limit": "<limit>",
      "offset": "<page number>"
}
```

Important: If your Post request body consists of an incremental value and the value is a date, then the value of *values* must be **[{{Start}}]**.

---

4. Enter the **Endpoint** part of the previously added Base URL.
5. Enter a **Description** for the endpoint.
6. In **Headers**, click **+ Add Row** and enter the custom headers in **Headers** as a key-value pair. The header parameters cannot be log filtering fields, such as start_date or end_date.
7. In **Query Parameters**, click **+ Add Row** and enter the request parameter's **Key** and **Value**.

   For example, if you are using the OData query filter, such as */api/alerts?$filter=(severity eq 'High') or (severity eq 'Medium')*, you must enter *$filter* as **Key** and *(severity eq 'High') or (severity eq 'Medium')* as **Value**.

   Query Parameters are sent in the request URL.

Important: If you need to provide the starting or end point of the log fetch, then they must be specified in either the Query parameters or Post request body. They must also be specified using Jinja template keywords, such as **{{Start}}** or **{{End}}**.

For example:

Post request body

```
{"filters:
[
{
"fieldName":"StartTimestamp",
"operator": "equals",
"values": {{Start}}
},
```

{
“fieldName”:”EndTimestamp”,
“operator”: “equals”,
“values”: {{End}} }
] }

Here, the field name **StartTimestamp** indicates the starting point of fetch, and **EndTimestamp** indicates the end point of fetch. These values are incremented dynamically in subsequent fetch attempts.

Query Parameters

StartTimestamp -> {{start}}

EndTimestamp -> {{end}}

---

8. In **Increment Value / Check Sum**, enter the increment field from the response of the RESTful API.

   For example, if the increment field is event_date and it is inside Events, then enter *Events.event_date*. The field is saved in CheckSum, a database that uses the field to record until data is fetched. This ensures there is no log duplication as Universal REST API Fetcher checks the CheckSum every time before fetching any new data.

9. Enter the **Response key**, which is an identifier to locate and parse logs within an API response.

10. Enter a **Custom Date Format** for API response. Universal REST API supports all date formats.

    Some of them are:

Date Type	Format	Example
UTC	%Y-%m-%dT%H:%M:%SZ	2023-04-27T07:18:52Z
ISO-8601	%Y-%m-%dT%H:%M:%S%z	2023-04-27T07:18:52+0000
RFC 2822	%a, %d %b %Y %H:%M:%S %z	Thu, 27 Apr 2023 07:18:52 +0000
RFC 850	%A, %d-%b-%y %H:%M:%S UTC	Thursday, 27-Apr-23 07:18:52 UTC

Continued on next page

Table 1 – continued from previous page

Date Type	Format	Example
RFC 1036	%a, %d %b %y %H:%M:%S %z	Thu, 27 Apr 23 07:18:52 +0000
RFC 1123	%a, %d %b %Y %H:%M:%S %z	Thu, 27 Apr 2023 07:18:52 +0000
RFC 822	%a, %d %b %y %H:%M:%S %z	Thu, 27 Apr 23 07:18:52 +0000
RFC 3339	%Y-%m-%dT%H:%M:%S%z	2023-04-27T07:18:52+00:00
ATOM	%Y-%m-%dT%H:%M:%S%z	2023-04-27T07:18:52+00:00
COOKIE	%A, %d-%b-%Y %H:%M:%S UTC	Thursday, 27-Apr-2023 07:18:52 UTC
RSS	%a, %d %b %Y %H:%M:%S %z	Thu, 27 Apr 2023 07:18:52 +0000
W3C	%Y-%m-%dT%H:%M:%S%z	2023-04-27T07:18:52+00:00
YYYY-DD-MM HH:MM:SS	%Y-%d-%m %H:%M:%S	2023-27-04 07:18:52
YYYY-DD-MM HH:MM:SS am/pm	%Y-%d-%m %I:%M:%S %p	2023-27-04 07:18:52 AM
DD-MM-YYYY HH:MM:SS	%d-%m-%Y %H:%M:%S	27-04-2023 07:18:52
MM-DD-YYYY HH:MM:SS	%m-%d-%Y %H:%M:%S	04-27-2023 07:18:52

11. In **Logs Filtering Parameters**, select the parameters to filter the incoming logs.

    11.1. Select a **Data format**.

        11.1.1. Select **ISO Date** to represent data using the International Standards Organization (ISO) format of "yyyy-MM-dd". Example: 2017-06-10. If you select **ISO Date**, then its value must be in the string format in the **Post request body**.

        11.1.2. Select **UNIX Epoch** to represent data using the UNIX epoch time format. It is a system for measuring time as the number of seconds that have elapsed since January 1, 1970, at 00:00:00 UTC (Coordinated Universal Time). Example: 1672475384.

        11.1.3. Select **UNIX Epoch (ms)** to represent data using the UNIX epoch time format with milliseconds precision. It is a system for measuring time as the number of milliseconds that have elapsed since January 1, 1970, at 00:00:00 UTC (Coordinated Universal Time). Example:1672475384000.

        11.1.4. Select **Custom Format** to define your own format for representing the data. The custom format can be created using *Date/Time patterns*.

11.1.5. Select a **Unique ID** to represent data using an unique ID. If you select **Unique ID** here, then its value must be in the number format in the **Post request body**.

12.2. Select an **Initial Fetch** date. Logs are fetched for the first time from this date.

13. In **Pagination Key**, enter the location of the following page URL from the response if the API supports pagination.

    For example, if the data from the RESTful API looks like the following, the pagination key is *metadata.links.next*.

```
"metadata": {

"links": {

        "self": "https://api.com/audit_logs",

        "next": "https://api.com/audit_logs?offset=500"

      }

      }
```

14. Click **Save Changes**.

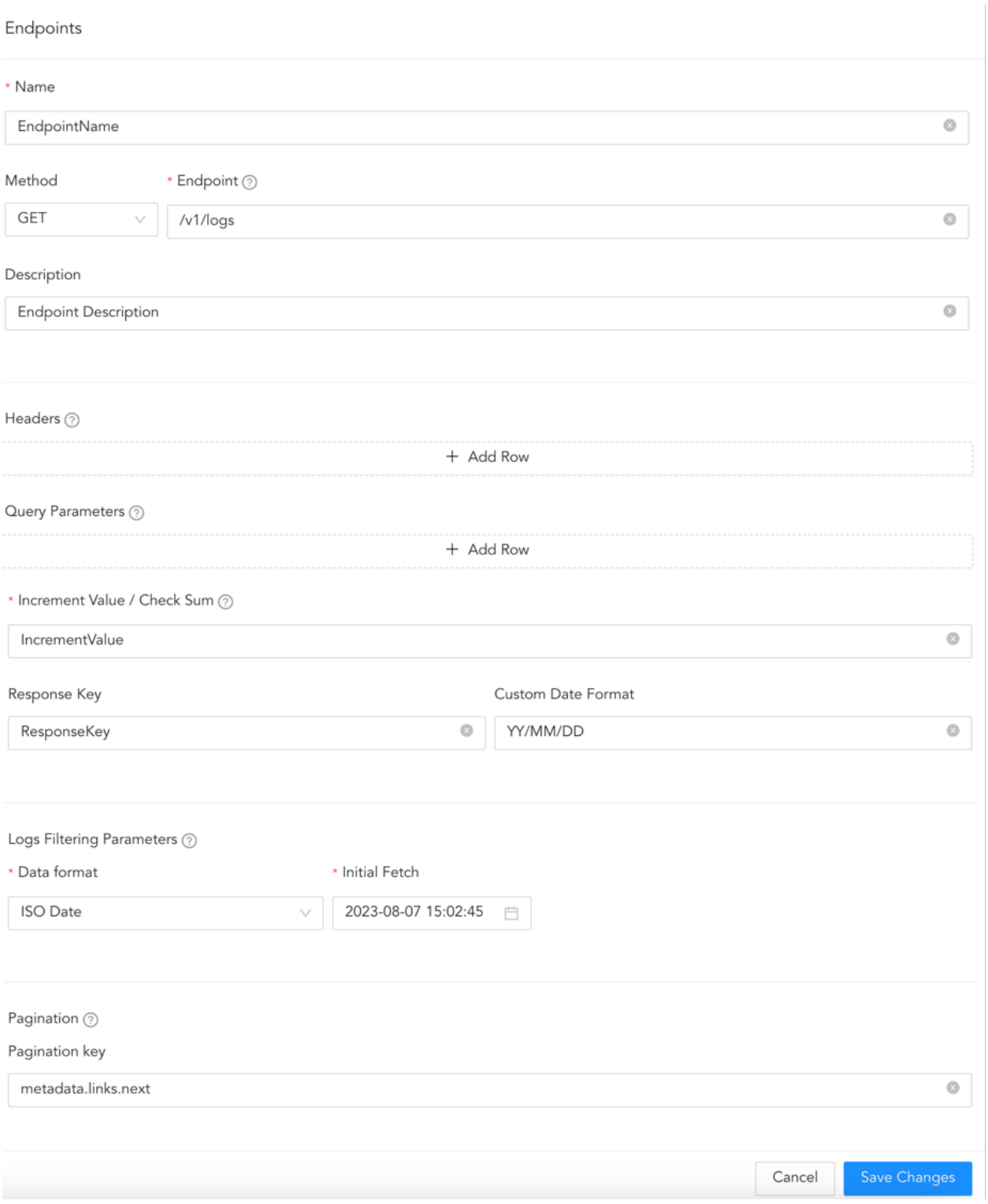

Fig. 3: Configuring Endpoint

To edit the endpoint configuration, click the (⋮) icon under **ACTION** and click **Edit**. Make the necessary changes and click **Save Changes**.

To delete the endpoint configuration, click the (⋮) icon under **ACTION** and click **Delete**.

4.4 Routing

In routing, you can create repos and routing criteria for Universal REST API Fetcher. Repos are locations where incoming logs are stored and routing criteria are created to determine the conditions under which these logs are sent to repos.

To create a repo:

1. Click **Routing** and **+ Create Repo**.
2. Enter a **Repo name**.
3. In **Path**, enter the location to store the incoming logs.
4. In **Retention (Days)**, enter the number of days logs are kept in a repository before they are automatically deleted.
5. In **Availability**, select the **Remote logpoint** and **Retention (Days)**.
6. Click **Create Repo**.

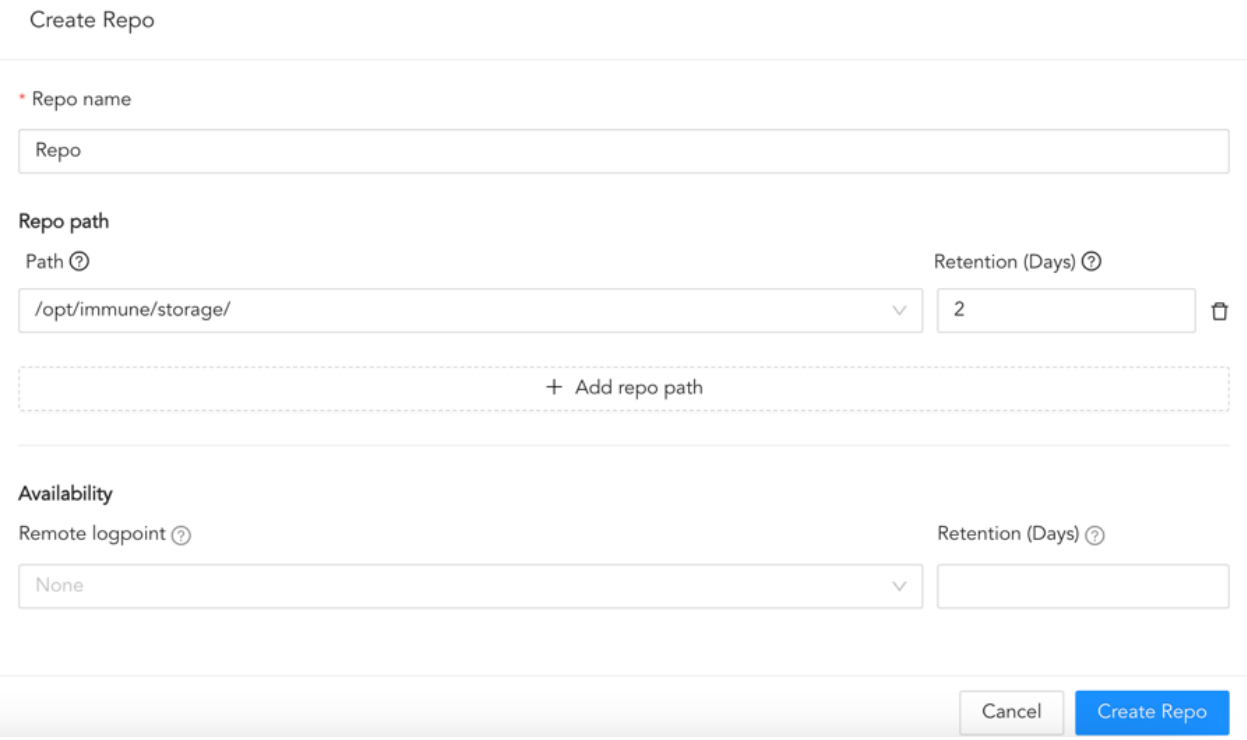

Fig. 4: Creating a Repo

In **Repo**, select the created repo to store logs.

To create Routing Criteria:

1. Click **+ Add row**.
2. Enter a **Key** and **Value**. The routing criteria is only applied to those logs which have this key-value pair.
3. Select an **Operation** for logs that have this key-value pair.

    3.1. Select **Store raw message** to store both the incoming and the normalized logs in the selected repo.
    3.2. Select **Discard raw message** to discard the incoming logs and store the normalized ones.
    3.3. Select **Discard entire event** to discard both the incoming and the normalized logs.
4. In **Repository**, select a repo to store logs.

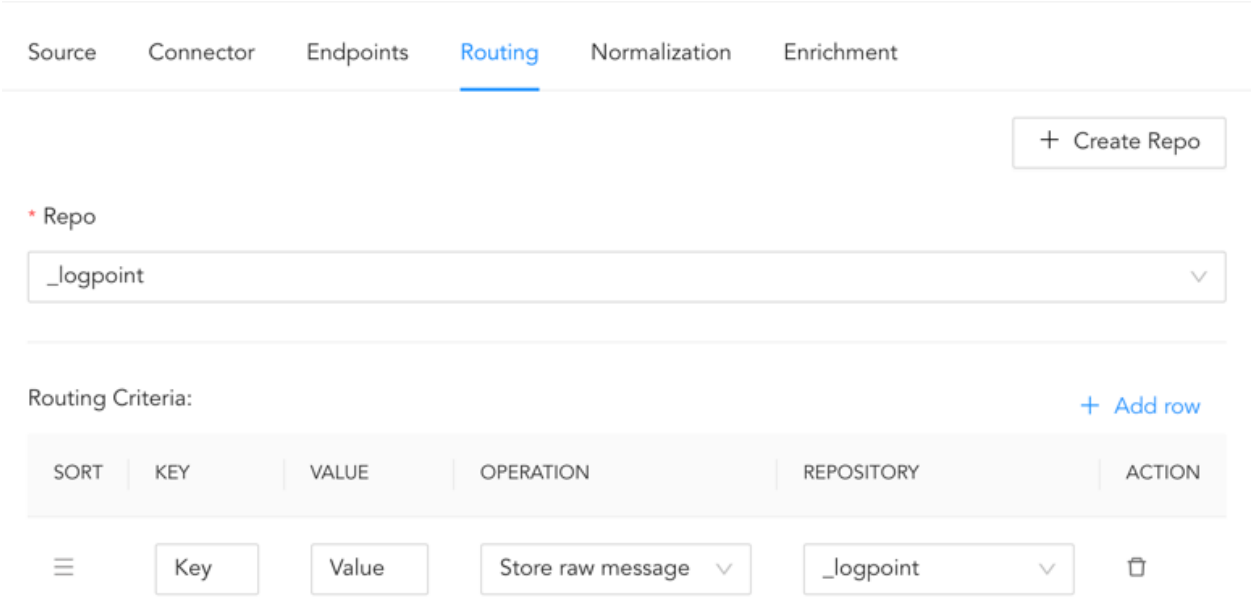

Fig. 5: Creating a Routing Criteria

Click the (🗑) icon under **Action** to delete the created routing criteria.

4.5 Normalization

In normalization, you can select normalizers for the incoming logs. Normalizers transform incoming logs into a standardized format for consistent and efficient analysis.

1. Click **Normalization**.
2. Select a **Normalizer** from the list and click the swap( ) icon.

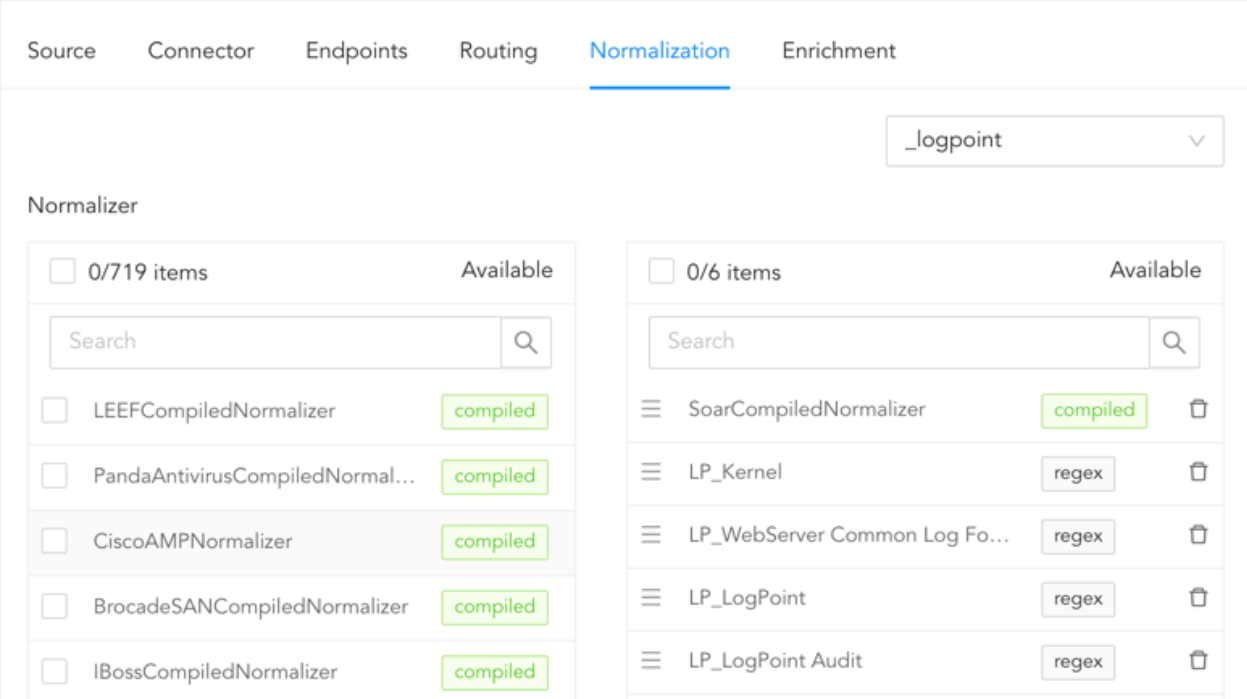

Fig. 6: Adding Normalizers

4.6 Enrichment

In enrichment, you can select an enrichment policy for the incoming logs. Enrichment policies are used to add additional information to a log, such as user information, device type or geolocation.

1. Click **Enrichment**.
2. Select an **Enrichment Policy**.

Click **Create Log Source** to save the configurations of Source, Connector, Endpoints, Routing, Normalization, and Enrichment.

You are redirected to **TASKS**, which displays the Universal REST API Fetcher configuration progress.