API Deployments Value Import Rules Patch v4

This is the latest version of the APIs.

Contents:

Create a list of value-based import rules for the specified deployment. Delete any previous rules applied to the same values.

NOTE: Import rules must be applied to individual deployments.

The generated rules apply to all flows that are imported into the Production instance after they have been created.

NOTE: Deployments pertain to Production instances of the Trifacta® platform. For more information, see *Overview of Deployment Manager*.

The response contains any previously created rules that have been deleted as a result of this change.

You can also make replacements in the import package based on object references. See *API Deployments Object Import Rules Patch v4*.

Version: v4

Required Permissions

NOTE: Each request to the Trifacta® platform must include authentication credentials. See *API Authentication*.

Request

Request Type: PATCH

Endpoint:

```
/v4/deployments/<id>/valueImportRules
```

where:

Parameter	Description
<id>	Internal identifier for the deployment

Request URI - Example:

```
/v4/deployments/4/valueImportRules
```

Request Body Example: Single value replacement

The following JSON array describes a single replacement rule for the S3 bucket name. In this case, the `wrangle-dev` bucket name has been replaced by the `wrangle-prod` bucket name, which means data is pulled in the Production deployment from the appropriate S3 bucket.

> **NOTE:** The executing user of any job must have access to any data source that is remapped in the new instance.

```
[
  {
    "type": "s3Bucket",
    "on": "wrangle-dev",
    "with": "wrangle-prod"
  }
]
```

Request Body Example: Multiple value replacements

The following JSON array describes two replacements for the `fileLocation` values. In this case, rules are applied in succession.

> **NOTE:** Rules are applied in the listed order. If you are applying multiple rules to the same object in the import package, the second rule must reference the expected changes applied by the first rule.

```
[
  {
    "type": "fileLocation",
    "on": "klamath",
    "with": "klondike"
  },
  {
    "type": "fileLocation",
    "on": "//dev//",
    "with": "/prod/"
  }
]
```

In the above:

- The first rule replaces the string `klamath` in the path to the source with the following value: `klondike`.
- The second rule performs a regular expression match on the string `/dev/`. Since the match is described using the regular expression syntax, the backslashes must be escaped. The replacement value is the following literal: `/prod/`.

You can specify matching values using the following types of matches:

Match Type	Example Syntax
string literal	`{"on":"d75255f0-a245-11e7-8618-adc1dbb4bed0"}`
regular expression	`{"on":"/[0-9a-zA-z]{8}-a245-11e7-8618-adc1dbb4bed0/"}`

NOTE: Use of Trifacta patterns is not supported.

For more information on patterns, see *Text Matching*.

Request Body Example: Replace database tables and paths

This example request includes replacements for a database table and its path (database name) in a relational publication.

NOTE: Rules are applied in the listed order. If you are applying multiple rules to the same object in the import package, the second rule must reference the expected changes applied by the first rule.

This type of replacement applies if the imported packages contain sources that are imported through two separate connections:

```
[
  {
    "type": "dbTableName",
    "on": "from_table_name",
    "with": "to_table_name"
  },
  {
    "type": "dbPath",
    "on": "from_path_element",
    "with": "to_path_element"
  },
]
```

Type	Description
`dbTableName`	Replaces the name of the table in the source (`on` value) with the new table name to use (`with` value).
`dbPath`	Replaces the path to the database in the source (`on` value) with the new path to use (`with` value). **NOTE:** The content of a dataset or output `dbPath` is an array. The regular expression for `on` is applied to every element in the `dbPath` value. Typically, there's only one element in the `dbPath` array. In some cases, there may be multiple elements, so be careful when specifying the `on` value.

Tip: The `on` parameter values can be provided as a regular expression.

Response

Response Status Code - Success: `200 - OK`

The response body contains any previously created rules that have been deleted as a result of this update.

Response Body Example: All new rule, no deletions

If the update does not overwrite any previous rules, then no rules are deleted. So, the response looks like the following:

```
{
    "deleted": {
        "data": []
    }
}
```

Response Body Example: Replace file location, delete previous rule

If you submit the request again, the response contains the rule definition of the previous update, which has been deleted.

```
{
    "deleted": {
        "data": [
            {
                "id": 1,
                "type": "s3Bucket",
                "on": "wrangle-dev",
                "with": "wrangle-prod",
                "createdAt": "2019-02-13T23:27:13.351Z",
                "updatedAt": "2019-02-13T23:27:13.351Z",
                "creator": {
                    "id": 7
                },
                "updater": {
                    "id": 7
                },
                "deployment": {
                    "id": 2
                }
            }
        ]
    }
}
```

Reference

Property	Description
id	Internal identifier for the value import rule
type	The type of value import rule: • `fileLocation` - the location of a specified file. • `s3Bucket` - location of the S3 bucket to modify
on	The specified literal or pattern to match.
with	The replacement value or pattern
createdAt	Timestamp for when the rule was created
updatedAt	Timestamp for when the rule was last updated
creator.id	Internal identifier of the user who created the rule
updater.id	Internal identifier of the user who last updated the rule
deployment.id	Internal identifier for the deployment from which the import rule was deleted.