

Connector APIs

[Connector \\[connectors.alauda](#) [ConnectorClass \\[connectors.alauda.io/v1apl](#)

Connector [connectors.alauda.io/v1alpha1]

Description

Connector is the Schema for the connectors API

Type

object

Specification

Property	Type	Description
<code>apiVersion</code>	<code>string</code>	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources
<code>kind</code>	<code>string</code>	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

Property	Type	Description
<code>metadata</code>	<code>ObjectMeta</code> ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
<code>spec</code>	<code>object</code>	ConnectorSpec defines the desired state of Connector
<code>status</code>	<code>object</code>	ConnectorStatus defines the observed state of Connector

.spec

Description

ConnectorSpec defines the desired state of Connector

Type

`object`

Property	Type	Description
<code>address</code>	<code>string</code>	Address is connector address
<code>auth</code>	<code>object</code>	Auth represents authenticate data of current connector
<code>connectorClassName</code>	<code>string</code>	ConnectorClassName of current connector

Property	Type	Description
<code>params</code>	<code>array</code>	Params information required for spec

`.spec.auth`

Description

Auth represents authenticate data of current connector

Type

`object`

Property	Type	Description
<code>name</code>	<code>string</code>	Name represent auth name that configured in ConnectorClass.spec.auth.types[].name
<code>params</code>	<code>array</code>	Params information required for authentication
<code>secretRef</code>	<code>object</code>	SecretRef secret reference when doing authentication

`.spec.auth.params`

Description

Params information required for authentication

Type

`array`

.spec.auth.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

object

Required

name

value

Property	Type	Description
name	string	
value		

.spec.auth.secretRef

Description

SecretRef secret reference when doing authentication

Type

object

Property	Type	Description
apiVersion	string	API version of the referent.
fieldPath	string	If referring to a piece of an object instead of an entire object, this string should contain a valid JSON/Go field access statement, such as <code>desiredState.manifest.containers[2]</code> . For example, if the object reference is to a container within a pod, this would take on a value like: <code>"spec.containers{name}"</code> (where

Property	Type	Description
		"name" refers to the name of the container that triggered the event) or if no container name is specified "spec.containers[2]" (container with index 2 in this pod). This syntax is chosen only to have some well-defined way of referencing a part of an object.
kind	string	Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds
name	string	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names
namespace	string	Namespace of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/
resourceVersion	string	Specific resourceVersion to which this reference is made, if any. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency
uid	string	UID of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#uids

.spec.params

Description

Params information required for spec

Type

array

.spec.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

object

Required

name

value

Property	Type	Description
name	string	
value		

.status

Description

ConnectorStatus defines the observed state of Connector

Type

object

Property	Type	Description
<code>annotations</code>	<code>object</code>	Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.
<code>api</code>	<code>object</code>	API contains the status information for the connector's api
<code>conditions</code>	<code>array</code>	Conditions the latest available observations of a resource's current state.
<code>observedGeneration</code>	<code>integer</code>	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.
<code>proxy</code>	<code>object</code>	Proxy contains the status information for the connector's proxy

.status.annotations

Description

Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.

Type

`object`

.status.api

Description

API contains the status information for the connector's api

Type

object

Property	Type	Description
path	string	Path provides the path for the connector API. it is the path of the connector API. it is used to construct the api path for the connector

.status.conditions

Description

Conditions the latest available observations of a resource's current state.

Type

array

.status.conditions[]

Description

Condition defines a readiness condition for a Knative resource. See: <https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#typical-status-properties>

Type

object

Required

status

type

Property	Type	Description
<code>lastTransitionTime</code>	<code>string</code>	LastTransitionTime is the last time the condition transitioned from one status to another. We use VolatileTime in place of metav1.Time to exclude this from creating equality.Semantic differences (all other things held constant).
<code>message</code>	<code>string</code>	A human readable message indicating details about the transition.
<code>reason</code>	<code>string</code>	The reason for the condition's last transition.
<code>severity</code>	<code>string</code>	Severity with which to treat failures of this type of condition. When this is not specified, it defaults to Error.
<code>status</code>	<code>string</code>	Status of the condition, one of True, False, Unknown.
<code>type</code>	<code>string</code>	Type of condition.

.status.proxy

Description

Proxy contains the status information for the connector's proxy

Type

`object`

Property	Type	Description
<code>httpAddress</code>	<code>object</code>	HTTPAddress provides the addressable HTTP endpoint for the connector proxy.

`.status.proxy.httpAddress`

Description

HTTPAddress provides the addressable HTTP endpoint for the connector proxy.

Type

`object`

Property	Type	Description
<code>CACerts</code>	<code>string</code>	CACerts is the Certification Authority (CA) certificates in PEM format according to https://www.rfc-editor.org/rfc/rfc7468 .
<code>audience</code>	<code>string</code>	Audience is the OIDC audience for this address.
<code>name</code>	<code>string</code>	Name is the name of the address.
<code>url</code>	<code>string</code>	

API Endpoints

The following API endpoints are available:

- `/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectors`

- **DELETE** : delete collection of Connector
- **GET** : list objects of kind Connector
- **POST** : create a new Connector
- `/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectors/{name}`
 - **DELETE** : delete the specified Connector
 - **GET** : read the specified Connector
 - **PATCH** : partially update the specified Connector
 - **PUT** : replace the specified Connector
- `/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectors/{name}/status`
 - **GET** : read status of the specified Connector
 - **PATCH** : partially update status of the specified Connector
 - **PUT** : replace status of the specified Connector

`/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectors`

HTTP method

DELETE

Description

delete collection of Connector

HTTP responses

HTTP code	Response body
200 - OK	Status schema
401 - Unauthorized	Empty

HTTP method

GET

Description

list objects of kind Connector

HTTP responses

HTTP code	Response body
200 - OK	<code>ConnectorList</code> schema
401 - Unauthorized	Empty

HTTP method

POST

Description

create a new Connector

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized <code>dryRun</code> directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
<code>fieldValidation</code>	<code>string</code>	<code>fieldValidation</code> instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a

Parameter	Type	Description
		BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
body	Connector schema	application/json formatted

HTTP responses

HTTP code	Response body
200 - OK	Connector schema
201 - Created	Connector schema
202 - Accepted	Connector schema
401 - Unauthorized	Empty

/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectors/{name}

HTTP method

DELETE

Description

delete the specified Connector

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	<code>Status</code> ↗ schema
202 - Accepted	<code>Status</code> ↗ schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified Connector

HTTP responses

HTTP code	Response body
200 - OK	<code>Connector</code> ↗ schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified Connector

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized <code>dryRun</code> directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
<code>fieldValidation</code>	<code>string</code>	<code>fieldValidation</code> instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a <code>BadRequest</code> error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	<code>Connector</code> schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified Connector

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized <code>dryRun</code> directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
<code>fieldValidation</code>	<code>string</code>	<code>fieldValidation</code> instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a <code>BadRequest</code> error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
<code>body</code>	<code>Connector</code> schema	<code>application/json</code> formatted

HTTP responses

HTTP code	Response body
200 - OK	<code>Connector</code> schema
201 - Created	<code>Connector</code> schema
401 - Unauthorized	Empty

/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectors/{name}/status

HTTP method

GET

Description

read status of the specified Connector

HTTP responses

HTTP code	Response body
200 - OK	<code>Connector</code> schema
401 - Unauthorized	Empty

HTTP method

PATCH

Description

partially update status of the specified Connector

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
<code>fieldValidation</code>	<code>string</code>	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a

Parameter	Type	Description
		warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	<code>Connector</code> schema
401 - Unauthorized	Empty

HTTP method

PUT

Description

replace status of the specified Connector

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
<code>fieldValidation</code>	<code>string</code>	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last

Parameter	Type	Description
		duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
body	Connector schema	application/json formatted

HTTP responses

HTTP code	Response body
200 - OK	Connector schema
201 - Created	Connector schema
401 - Unauthorized	Empty

ConnectorClass

[connectors.alauda.io/v1alpha1]

Description

ConnectorClass is the Schema for the connectorclasses API

Type

object

Specification

Property	Type	Description
<code>apiVersion</code>	<code>string</code>	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources

Property	Type	Description
<code>kind</code>	<code>string</code>	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds
<code>metadata</code>	<code>ObjectMeta</code>	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
<code>spec</code>	<code>object</code>	Spec defines the desired state of ConnectorClass
<code>status</code>	<code>object</code>	Status defines the actual state of ConnectorClass

.spec

Description

Spec defines the desired state of ConnectorClass

Type

`object`

Property	Type	Description
<code>address</code>	<code>object</code>	Address indicates address param constraints for this ConnectorClass of connectors we only support string param type

Property	Type	Description
<code>api</code>	<code>object</code>	API defines connectorclass plugin api address and openapi specification <code>api.ref</code> can be address of plugin api, it should be a kubernetes svc <code>api.uri</code> can be an absolute URL(non-empty scheme and non-empty host) pointing to the target or a relative URI. Relative URIs will be resolved using the base URI retrieved from Ref. <code>api.CACerts</code> and <code>api.audience</code> is not implemented now
<code>auth</code>	<code>object</code>	Auth indicates authentication constraints for this ConnectorClass of connectors
<code>authProbes</code>	<code>array</code>	AuthProbes defines authentication probe for this ConnectorClass of connectors
<code>configurations</code>	<code>array</code>	Configurations defines connectorclass configuration
<code>livenessProbe</code>	<code>object</code>	LivenessProbe defines liveness probe for this ConnectorClass of connectors
<code>params</code>	<code>array</code>	Params indicates param constraints for this ConnectorClass of connectors we only support string param type
<code>proxy</code>	<code>object</code>	Proxy defines the proxy configuration for this ConnectorClass. Specifies how network traffic should be

Property	Type	Description
		routed through a proxy server.

.spec.address

Description

Address indicates address param constraints for this ConnectorClass of connectors we only support string param type

Type

object

Required

name

Property	Type	Description
default		Default is the value a parameter takes if no input value is supplied.
description	string	Description is a user-facing description of the parameter that may be used to populate a UI.
enum	array	Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.
name	string	Name declares the name by which a parameter is referenced.

Property	Type	Description
<code>properties</code>	<code>object</code>	Properties is the JSON Schema properties to support key-value pairs parameter.
<code>type</code>	<code>string</code>	Type is the user-specified type of the parameter. The possible types are currently "string", "array" and "object", and "string" is the default.

`.spec.address.enum`

Description

Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.

Type

`array`

`.spec.address.enum[]`

Type

`string`

`.spec.address.properties`

Description

Properties is the JSON Schema properties to support key-value pairs parameter.

Type

`object`

`.spec.api`

Description

API defines connectorclass plugin api address and openapi specification `api.ref` can be address of plugin api, it should be a kubernetes svc `api.uri` can be an absolute URL(non-empty scheme and non-empty host) pointing to the target or a relative URI. Relative URIs will be resolved using the base URI retrieved from Ref. `api.CACerts` and `api.audience` is not implemented now

Type

object

Property	Type	Description
CACerts	string	CACerts are Certification Authority (CA) certificates in PEM format according to https://www.rfc-editor.org/rfc/rfc7468 [↗] . If set, these CAs are appended to the set of CAs provided by the Addressable target, if any.
audience	string	Audience is the OIDC audience. This need only be set, if the target is not an Addressable and thus the Audience can't be received from the Addressable itself. In case the Addressable specifies an Audience too, the Destinations Audience takes preference.
openapi		OpenAPI defines the openapi specification for the connectorclass api This OpenAPI specification can describe an API customized for the ConnectorClass, or it can describe the original tool's API exposed via the ConnectorClass's proxy. if client request path is not in the OpenAPI specification, it will just forward the request to proxy of the connectorclass.

Property	Type	Description
<code>permissions</code>	<code>object</code>	Permissions defines the request matching rules for different operations on the connectorclass api
<code>ref</code>	<code>object</code>	Ref points to an Addressable.
<code>uri</code>	<code>string</code>	URI can be an absolute URL(non-empty scheme and non-empty host) pointing to the target or a relative URI. Relative URIs will be resolved using the base URI retrieved from Ref.

`.spec.api.permissions`

Description

Permissions defines the request matching rules for different operations on the connectorclass api

Type

`object`

Property	Type	Description
<code>rego</code>	<code>string</code>	<p>Rego contains the Rego policy script that maps API requests to permission verbs. The script evaluates incoming requests and determines the required permission level.</p> <p>Requirements:</p> <ol style="list-style-type: none"> 1. Must be defined under the 'permissions' package 2. Must produce a 'result' object matching the PermissionMappingResult structure

Property	Type	Description
		<p>Input Variables:</p> <ul style="list-style-type: none">• request.path (string): Request path• request.method (string): HTTP method (GET, POST, PUT, DELETE, etc.)• request.headers (map<string, list>): Request headers• request.query (map<string, list>): Request query parameters• request.body (dynamic): Parsed JSON body (available when Content-Type is application/json) <p>Output:</p> <ul style="list-style-type: none">• result.verb (string): Permission verb - "read", "write", "delete", or "" (default to http verb mapping) <p>Example: package permissions import future.keywords.if</p> <pre>result = { "verb": "read" } if { startswith(input.request.path, "/search") input.request.method == "POST" } else = { "verb": "" }</pre>

.spec.api.ref

Description

Ref points to an Addressable.

Type

object

Required

kind

name

Property	Type	Description
<code>address</code>	<code>string</code>	Address points to a specific Address Name.
<code>apiVersion</code>	<code>string</code>	API version of the referent.
<code>group</code>	<code>string</code>	Group of the API, without the version of the group. This can be used as an alternative to the APIVersion, and then resolved using ResolveGroup. Note: This API is EXPERIMENTAL and might break anytime. For more details: https://github.com/knative/eventing/issues/5086
<code>kind</code>	<code>string</code>	Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds
<code>name</code>	<code>string</code>	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names
<code>namespace</code>	<code>string</code>	Namespace of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ This is optional field, it gets defaulted to the object holding it if left out.

.spec.auth

Description

Auth indicates authentication constraints for this ConnectorClass of connectors

Type

object

Property	Type	Description
types	array	Types represent the authentication types supported by connectors of the current connectorclass type When the array length is greater than 1, it means supporting multiple types, and the Connector can choose any one when using.

.spec.auth.types

Description

Types represent the authentication types supported by connectors of the current connectorclass type When the array length is greater than 1, it means supporting multiple types, and the Connector can choose any one when using.

Type

array

.spec.auth.types[]

Description

ConnectorClassAuthType represent the authentication types supported by connectors of the current connectorclass type

Type

object

Required

name

Property	Type	Description
<code>description</code>	<code>string</code>	Description is the description of the AuthType
<code>displayName</code>	<code>string</code>	DisplayName is the human readable name of the AuthType
<code>generator</code>	<code>object</code>	Generator specifies how to generate authentication data dynamically. Can be used to implement custom authentication logic.
<code>name</code>	<code>string</code>	Name of the AuthType Must be unique within the ConnectorClass.
<code>optional</code>	<code>boolean</code>	Optional indicates whether the authentication information is optional for this ConnectorClass of connectors the default value is false
<code>params</code>	<code>array</code>	Params declares the data fields included in this authentication type. For known types, the definition of included params is optional. If not defined, the conventional params will be used.
<code>secretType</code>	<code>string</code>	SecretType represents the secret type of the current authentication information follow k8s secret type definition. eg.kubernetes.io/basic-auth, kubernetes.io/ssh-auth, kubernetes.io/opaque

`.spec.auth.types[].generator`

Description

Generator specifies how to generate authentication data dynamically. Can be used to implement custom authentication logic.

Type

object

Property	Type	Description
<code>rego</code>	<code>string</code>	<p>Rego contains the Rego policy script that will be evaluated to generate authentication data. The script must define an 'auth' object that matches the following rules:</p> <ol style="list-style-type: none">1. Define its rules under the 'proxy' package2. Produce an 'auth' object containing AuthInjection structure.

`.spec.auth.types[].params`

Description

Params declares the data fields included in this authentication type. For known types, the definition of included params is optional. If not defined, the conventional params will be used.

Type

array

`.spec.auth.types[].params[]`

Description

ParamSpec defines arbitrary parameters needed beyond typed inputs (such as resources).

Type

object

Required

name

Property	Type	Description
default		Default is the value a parameter takes if no input value is supplied.
description	string	Description is a user-facing description of the parameter that may be used to populate a UI.
enum	array	Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.
name	string	Name declares the name by which a parameter is referenced.
properties	object	Properties is the JSON Schema properties to support key-value pairs parameter.
type	string	Type is the user-specified type of the parameter. The possible types are currently "string", "array" and "object", and "string" is the default.

.spec.auth.types[].params[].enum

Description

Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.

Type

array

.spec.auth.types[].params[].enum[]

Type

string

.spec.auth.types[].params[].properties

Description

Properties is the JSON Schema properties to support key-value pairs parameter.

Type

object

.spec.authProbes

Description

AuthProbes defines authentication probe for this ConnectorClass of connectors

Type

array

.spec.authProbes[]

Description

ConnectorClassAuthProbe represents network the detection configuration

Type

object

Required

authName

Property	Type	Description
authName	string	AuthName corresponds to <code>spec.auth.types[].name</code> , indicating the way to check for the corresponding authentication type
params	array	Params declares the data fields included in this probe it will use param value when probe
probe	object	Probe represents current detection configuration

`.spec.authProbes[].params`

Description

Params declares the data fields included in this probe it will use param value when probe

Type

array

`.spec.authProbes[].params[]`

Description

ParamSpec defines arbitrary parameters needed beyond typed inputs (such as resources).

Type

object

Required

name

Property	Type	Description
default		Default is the value a parameter takes if no input value is supplied.
description	string	Description is a user-facing description of the parameter that may be used to populate a UI.
enum	array	Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.
name	string	Name declares the name by which a parameter is referenced.
properties	object	Properties is the JSON Schema properties to support key-value pairs parameter.
type	string	Type is the user-specified type of the parameter. The possible types are currently "string", "array" and "object", and "string" is the default.

`.spec.authProbes[].params[].enum`

Description

Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.

Type

`array`

`.spec.authProbes[].params[].enum[]`

Type

`string`

`.spec.authProbes[].params[].properties`

Description

Properties is the JSON Schema properties to support key-value pairs parameter.

Type

`object`

`.spec.authProbes[].probe`

Description

Probe represents current detection configuration

Type

`object`

Property	Type	Description
<code>api</code>	<code>object</code>	API defines a network detection probe that uses the ConnectorClass API. The API endpoint for the probe is resolved from the ConnectorClass's <code>spec.api</code> configuration. If <code>spec.api</code> is not configured or is invalid, no API detection will be performed. The <code>Path</code> field within <code>APIProbeAction</code> specifies a relative path that is appended to the host of URI resolved from <code>spec.api</code> (e.g., <code>spec.api.uri</code>).

Property	Type	Description
<code>http</code>	<code>object</code>	Http represents the network detection using the http get method

`.spec.authProbes[].probe.api`

Description

API defines a network detection probe that uses the ConnectorClass API. The API endpoint for the probe is resolved from the ConnectorClass's `spec.api` configuration. If `spec.api` is not configured or is invalid, no API detection will be performed. The `Path` field within `APIProbeAction` specifies a relative path that is appended to the host of URI resolved from `spec.api` (e.g., `spec.api.uri`).

Type

`object`

Property	Type	Description
<code>path</code>	<code>string</code>	Path represents the API address accessed during the current detection it is relative path, it will be appended to the host of URI resolved from <code>spec.api</code> (e.g., <code>spec.api.uri</code>) when execute the probe

`.spec.authProbes[].probe.http`

Description

Http represents the network detection using the http get method

Type

`object`

Required

`path`

Property	Type	Description
<code>disableRedirect</code>	<code>boolean</code>	DisableRedirect indicates whether the probe should follow redirects, default is false
<code>host</code>	<code>string</code>	Host represents the tool address for the current detection. usually, it would be empty, it will use the <code>spec.address</code> of connector
<code>httpHeaders</code>	<code>array</code>	Custom headers to set in the request. HTTP allows repeated headers.
<code>method</code>	<code>string</code>	Method represents the HTTP method to use for the probe, support method: GET, POST. default is GET
<code>path</code>	<code>string</code>	Path represents the API address accessed during the current detection it is relative path, it will be appended to the host of URI resolved from <code>spec.address</code> of connector when execute the probe
<code>response</code>	<code>object</code>	Response defines the expected response validation rules. If not specified, the probe is considered successful if the request completes without error and returns a 2xx status code.
<code>scheme</code>	<code>string</code>	Scheme to use for connecting to the host. If empty:

Property	Type	Description
		<ul style="list-style-type: none"> When Host is empty or matches the connector's address, the scheme from the connector's address is used. Otherwise, defaults to http. If specified, this value will be used regardless of Host or connector's address.

`.spec.authProbes[].probe.http.httpHeaders`

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

array

`.spec.authProbes[].probe.http.httpHeaders[]`

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

object

Required

name

value

Property	Type	Description
name	string	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.

Property	Type	Description
value	string	The header field value

.spec.authProbes[].probe.http.response

Description

Response defines the expected response validation rules. If not specified, the probe is considered successful if the request completes without error and returns a 2xx status code.

Type

object

Property	Type	Description
cel	string	<p>CEL contains a CEL (Common Expression Language) expression for response validation. The expression must evaluate to a boolean value. Available variables:</p> <ul style="list-style-type: none"> response.statusCode (int): HTTP status code response.headers (map<string, list>): Response headers response.body (dynamic): Parsed JSON body (if Content-Type is application/json and body is valid JSON) response.bodyString (string): Response body as string <p>Example expressions:</p> <ul style="list-style-type: none"> response.statusCode == 200 && response.body.status == 'healthy' response.body.uptime > 60 && response.body.errors.size() == 0 response.body.contains('success') && !response.body.contains('error') response.headers['content-type'][0].startsWith('application/json')

Property	Type	Description
		<p>CEL is recommended for simple to medium complexity validations:</p> <ul style="list-style-type: none"> • Intuitive syntax similar to JavaScript/C/Python • Better performance for simple expressions • Native support in Kubernetes ecosystem • Lower learning curve <p>When both CEL and built-in rules (Contains, Regex, JSONPath) are specified, all rules must pass (AND logic).</p>

.spec.configurations

Description

Configurations defines connectorclass configuration

Type

array

.spec.configurations[]

Description

ConnectorClassConfiguration defines connectorclass configuration

Type

object

Property	Type	Description
annotations	object	<p>Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata. They are not queryable and should be preserved when modifying objects. More info:</p>

Property	Type	Description
		https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations ↗
<code>data</code>	<code>object</code>	Data contains the configuration data. Each key must consist of alphanumeric characters, '-', '_' or '.'.
<code>name</code>	<code>string</code>	Name of the configuration

`.spec.configurations[].annotations`

Description

Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata. They are not queryable and should be preserved when modifying objects. More info:

<https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations>

Type

`object`

`.spec.configurations[].data`

Description

Data contains the configuration data. Each key must consist of alphanumeric characters, '-', '_' or '.'.

Type

`object`

`.spec.livenessProbe`

Description

LivenessProbe defines liveness probe for this ConnectorClass of connectors

Type

object

Property	Type	Description
api	object	API defines a network detection probe that uses the ConnectorClass API. The API endpoint for the probe is resolved from the ConnectorClass's <code>spec.api</code> configuration. If <code>spec.api</code> is not configured or is invalid, no API detection will be performed. The <code>Path</code> field within <code>APIProbeAction</code> specifies a relative path that is appended to the host of URI resolved from <code>spec.api</code> (e.g., <code>spec.api.uri</code>).
http	object	Http represents the network detection using the http get method

.spec.livenessProbe.api

Description

API defines a network detection probe that uses the ConnectorClass API. The API endpoint for the probe is resolved from the ConnectorClass's `spec.api` configuration. If `spec.api` is not configured or is invalid, no API detection will be performed. The `Path` field within `APIProbeAction` specifies a relative path that is appended to the host of URI resolved from `spec.api` (e.g., `spec.api.uri`).

Type

object

Property	Type	Description
path	string	Path represents the API address accessed during the current detection it is relative path, it will be appended to the host of URI

Property	Type	Description
		resolved from <code>spec.api</code> (e.g., <code>spec.api.uri</code>) when execute the probe

.spec.livenessProbe.http

Description

Http represents the network detection using the http get method

Type

object

Required

path

Property	Type	Description
<code>disableRedirect</code>	boolean	DisableRedirect indicates whether the probe should follow redirects, default is false
<code>host</code>	string	Host represents the tool address for the current detection. usually, it would be empty, it will use the <code>spec.address</code> of connector
<code>httpHeaders</code>	array	Custom headers to set in the request. HTTP allows repeated headers.
<code>method</code>	string	Method represents the HTTP method to use for the probe, support method: GET, POST. default is GET

Property	Type	Description
<code>path</code>	<code>string</code>	Path represents the API address accessed during the current detection it is relative path, it will be appended to the host of URI resolved from <code>spec.address</code> of connector when execute the probe
<code>response</code>	<code>object</code>	Response defines the expected response validation rules. If not specified, the probe is considered successful if the request completes without error and returns a 2xx status code.
<code>scheme</code>	<code>string</code>	<p>Scheme to use for connecting to the host. If empty:</p> <ul style="list-style-type: none">• When Host is empty or matches the connector's address, the scheme from the connector's address is used.• Otherwise, defaults to http. If specified, this value will be used regardless of Host or connector's address.

`.spec.livenessProbe.http.httpHeaders`

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

`array`

`.spec.livenessProbe.http.httpHeaders[]`

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

object

Required

name

value

Property	Type	Description
name	string	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.
value	string	The header field value

.spec.livenessProbe.http.response

Description

Response defines the expected response validation rules. If not specified, the probe is considered successful if the request completes without error and returns a 2xx status code.

Type

object

Property	Type	Description
cel	string	CEL contains a CEL (Common Expression Language) expression for response validation. The expression must evaluate to a boolean value. Available variables: <ul style="list-style-type: none">response.statusCode (int): HTTP status coderesponse.headers (map<string, list>): Response headers

Property	Type	Description
		<ul style="list-style-type: none"> response.body (dynamic): Parsed JSON body (if Content-Type is application/json and body is valid JSON) response.bodyString (string): Response body as string <p>Example expressions:</p> <ul style="list-style-type: none"> response.statusCode == 200 && response.body.status == 'healthy' response.body.uptime > 60 && response.body.errors.size() == 0 response.body.contains('success') && !response.body.contains('error') response.headers['content-type'][0].startsWith('application/json') <p>CEL is recommended for simple to medium complexity validations:</p> <ul style="list-style-type: none"> Intuitive syntax similar to JavaScript/C/Python Better performance for simple expressions Native support in Kubernetes ecosystem Lower learning curve <p>When both CEL and built-in rules (Contains, Regex, JSONPath) are specified, all rules must pass (AND logic).</p>

.spec.params

Description

Params indicates param constraints for this ConnectorClass of connectors we only support string param type

Type

array

.spec.params[]

Description

ParamSpec defines arbitrary parameters needed beyond typed inputs (such as resources).

Type

object

Required

name

Property	Type	Description
default		Default is the value a parameter takes if no input value is supplied.
description	string	Description is a user-facing description of the parameter that may be used to populate a UI.
enum	array	Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.
name	string	Name declares the name by which a parameter is referenced.
properties	object	Properties is the JSON Schema properties to support key-value pairs parameter.
type	string	Type is the user-specified type of the parameter. The possible types are currently "string", "array" and "object", and "string" is the default.

`.spec.params[].enum`

Description

Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.

Type

array

`.spec.params[].enum[]`

Type

string

`.spec.params[].properties`

Description

Properties is the JSON Schema properties to support key-value pairs parameter.

Type

object

`.spec.proxy`

Description

Proxy defines the proxy configuration for this ConnectorClass. Specifies how network traffic should be routed through a proxy server.

Type

object

Property	Type	Description
<code>CACerts</code>	<code>string</code>	CACerts are Certification Authority (CA) certificates in PEM format according to https://www.rfc-editor.org/rfc/rfc7468 .

Property	Type	Description
		If set, these CAs are appended to the set of CAs provided by the Addressable target, if any.
<code>audience</code>	<code>string</code>	Audience is the OIDC audience. This need only be set, if the target is not an Addressable and thus the Audience can't be received from the Addressable itself. In case the Addressable specifies an Audience too, the Destinations Audience takes preference.
<code>authExtractor</code>	<code>object</code>	AuthExtractor specifies the method used to extract authentication data from incoming requests. The AuthExtractor is responsible for extracting the token required to perform proxy permission validation.
<code>ref</code>	<code>object</code>	Ref points to an Addressable.
<code>uri</code>	<code>string</code>	URI can be an absolute URL(non-empty scheme and non-empty host) pointing to the target or a relative URI. Relative URIs will be resolved using the base URI retrieved from Ref.

.spec.proxy.authExtractor

Description

AuthExtractor specifies the method used to extract authentication data from incoming requests. The AuthExtractor is responsible for extracting the token required to perform proxy permission validation.

Type

object

Property	Type	Description
		Rego contains the Rego policy script that will be evaluated to extract proxy authentication data from the request. The script must define an 'auth' object that matches the following rules:
rego	string	<ol style="list-style-type: none"> 1. Define its rules under the 'proxy' package 2. Produce an 'auth' object has structure same as ProxyAuthExtractorResult. Example: package proxy auth = { "token": input.request.headers["Private-Token"][0] }

.spec.proxy.ref

Description

Ref points to an Addressable.

Type

object

Required

kind

name

Property	Type	Description
address	string	Address points to a specific Address Name.
apiVersion	string	API version of the referent.
group	string	Group of the API, without the version of the group. This can be used as an alternative to the APIVersion, and then resolved

Property	Type	Description
		using ResolveGroup. Note: This API is EXPERIMENTAL and might break anytime. For more details: https://github.com/knative/eventing/issues/5086
kind	string	Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds
name	string	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names
namespace	string	Namespace of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ This is optional field, it gets defaulted to the object holding it if left out.

.status

Description

Status defines the actual state of ConnectorClass

Type

object

Property	Type	Description
annotations	object	Annotations is additional Status fields for the Resource to save some additional State as well as

Property	Type	Description
		convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.
<code>api</code>	<code>object</code>	API represents status of connectorclass api it will resolved based on <code>spec.api</code> if <code>spec.api</code> is empty or invalid, it will not be set if current field is empty, the connectorclass cannot provides any api service.
<code>conditions</code>	<code>array</code>	Conditions the latest available observations of a resource's current state.
<code>observedGeneration</code>	<code>integer</code>	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.
<code>proxy</code>	<code>object</code>	Proxy represents status of connectorclass proxy it will resolved based on <code>spec.proxy</code> if <code>spec.proxy</code> is empty or invalid, it will not be set if current field is empty, the connectorclass cannot provides any proxy service.

.status.annotations

Description

Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.

Type

object

.status.api

Description

API represents status of connectorclass api it will resolved based on `spec.api` if `spec.api` is empty or invalid, it will not be set if current field is empty, the connectorclass cannot provides any api service.

Type

object

Property	Type	Description
address	object	Address is a single Addressable address.

.status.api.address

Description

Address is a single Addressable address.

Type

object

Property	Type	Description
CACerts	string	CACerts is the Certification Authority (CA) certificates in PEM format according to https://www.rfc-editor.org/rfc/rfc7468 .
audience	string	Audience is the OIDC audience for this address.

Property	Type	Description
<code>name</code>	<code>string</code>	Name is the name of the address.
<code>url</code>	<code>string</code>	

.status.conditions

Description

Conditions the latest available observations of a resource's current state.

Type

`array`

.status.conditions[]

Description

Condition defines a readiness condition for a Knative resource. See: <https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#typical-status-properties>

Type

`object`

Required

`status`

`type`

Property	Type	Description
<code>lastTransitionTime</code>	<code>string</code>	LastTransitionTime is the last time the condition transitioned from one status to another. We use VolatileTime in place of metav1.Time to exclude this from creating equality.Semantic differences (all other things held constant).
<code>message</code>	<code>string</code>	A human readable message indicating details about the transition.
<code>reason</code>	<code>string</code>	The reason for the condition's last transition.
<code>severity</code>	<code>string</code>	Severity with which to treat failures of this type of condition. When this is not specified, it defaults to Error.
<code>status</code>	<code>string</code>	Status of the condition, one of True, False, Unknown.
<code>type</code>	<code>string</code>	Type of condition.

.status.proxy

Description

Proxy represents status of connectorclass proxy it will resolved based on `spec.proxy` if `spec.proxy` is empty or invalid, it will not be set if current field is empty, the connectorclass cannot provides any proxy service.

Type

object

Property	Type	Description
httpAddress	object	HttpAddress is a single Addressable address.

.status.proxy.httpAddress

Description

HttpAddress is a single Addressable address.

Type

object

Property	Type	Description
CACerts	string	CACerts is the Certification Authority (CA) certificates in PEM format according to https://www.rfc-editor.org/rfc/rfc7468 .
audience	string	Audience is the OIDC audience for this address.
name	string	Name is the name of the address.
url	string	

API Endpoints

The following API endpoints are available:

- `/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorclasses`
 - **DELETE** : delete collection of ConnectorClass
 - **GET** : list objects of kind ConnectorClass
 - **POST** : create a new ConnectorClass
- `/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorclasses/{name}`
 - **DELETE** : delete the specified ConnectorClass
 - **GET** : read the specified ConnectorClass
 - **PATCH** : partially update the specified ConnectorClass
 - **PUT** : replace the specified ConnectorClass
- `/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorclasses/{name}/status`
 - **GET** : read status of the specified ConnectorClass
 - **PATCH** : partially update status of the specified ConnectorClass
 - **PUT** : replace status of the specified ConnectorClass

`/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorclasses`

HTTP method

DELETE

Description

delete collection of ConnectorClass

HTTP responses

HTTP code	Response body
200 - OK	Status ✓ schema
401 - Unauthorized	Empty

HTTP method

GET

Description

list objects of kind ConnectorClass

HTTP responses

HTTP code	Response body
200 - OK	<code>ConnectorClassList</code> schema
401 - Unauthorized	Empty

HTTP method

POST

Description

create a new ConnectorClass

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
<code>fieldValidation</code>	<code>string</code>	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default

Parameter	Type	Description
		in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
body	ConnectorClass schema	application/json formatted

HTTP responses

HTTP code	Response body
200 - OK	ConnectorClass schema
201 - Created	ConnectorClass schema
202 - Accepted	ConnectorClass schema
401 - Unauthorized	Empty

/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorclasses/{name}

HTTP method

DELETE

Description

delete the specified ConnectorClass

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized <code>dryRun</code> directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	<code>Status</code> ↗ schema
202 - Accepted	<code>Status</code> ↗ schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified ConnectorClass

HTTP responses

HTTP code	Response body
200 - OK	<code>ConnectorClass</code> ↗ schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified ConnectorClass

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized <code>dryRun</code> directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
<code>fieldValidation</code>	<code>string</code>	<code>fieldValidation</code> instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a <code>BadRequest</code> error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	<code>ConnectorClass</code> schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified `ConnectorClass`

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized <code>dryRun</code> directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
<code>fieldValidation</code>	<code>string</code>	<code>fieldValidation</code> instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a <code>BadRequest</code> error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
<code>body</code>	<code>ConnectorClass</code> schema	<code>application/json</code> formatted

HTTP responses

HTTP code	Response body
200 - OK	<code>ConnectorClass</code> schema
201 - Created	<code>ConnectorClass</code> schema
401 - Unauthorized	Empty

/apis/connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorclasses/{name}/status

HTTP method

GET

Description

read status of the specified ConnectorClass

HTTP responses

HTTP code	Response body
200 - OK	<code>ConnectorClass</code> schema
401 - Unauthorized	Empty

HTTP method

PATCH

Description

partially update status of the specified ConnectorClass

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
<code>fieldValidation</code>	<code>string</code>	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a

Parameter	Type	Description
		warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	<code>ConnectorClass</code> schema
401 - Unauthorized	Empty

HTTP method

PUT

Description

replace status of the specified ConnectorClass

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
<code>fieldValidation</code>	<code>string</code>	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last

Parameter	Type	Description
		duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
body	ConnectorClass schema	application/json formatted

HTTP responses

HTTP code	Response body
200 - OK	ConnectorClass schema
201 - Created	ConnectorClass schema
401 - Unauthorized	Empty