Connector APIs

Connector\[connectors.katanor ConnectorClass\[connectors.katanomi.dev/v1

Connector[connectors.katanomi.dev/v1alpha1]

Description

Connector is the Schema for the connectors API

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	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 ↗
`kind`	`string`	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:

Property	Type	Description
		https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	ConnectorSpec defines the desired state of Connector
`status`	`object`	ConnectorStatus defines the observed state of Connector

.spec

Description

ConnectorSpec defines the desired state of Connector

Type

`object`

Property	Type	Description
`address`	`string`	Address is connector address
`auth`	`object`	Auth represents authenticate data of current connector

Property	Type	Description
`connectorClassName`	`string`	ConnectorClassName of current connector

.spec.auth

Description

Auth represents authenticate data of current connector

Type

`object`

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

.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 desiredState.manifest.containers[2]. For example, if the object reference is to a container within a pod, this would take on a value like: "spec.containers{name}" (where "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/ ↗

Property	Type	Description
`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 ↗

.status

Description

ConnectorStatus defines the observed state of Connector

Type

`object`

Property	Type	Description
`annotations`	`object`	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.

Property	Type	Description
`conditions`	`array`	Conditions the latest available observations of a resource's current state.
`observedGeneration`	`integer`	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.
`proxy`	`object`	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.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
`lastTransitionTime`	`string`	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).
`message`	`string`	A human readable message indicating details about the transition.
`reason`	`string`	The reason for the condition's last transition.
`severity`	`string`	Severity with which to treat failures of this type of condition. When this is not specified, it defaults to Error.
`status`	`string`	Status of the condition, one of True, False, Unknown.

Property	Type	Description
`type`	`string`	Type of condition.

.status.proxy

Description

Proxy contains the status information for the connector's proxy

Type

`object`

Property	Type	Description
`httpAddress`	`object`	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
`CACerts`	`string`	CACerts is the Certification Authority (CA) certificates in PEM format according to https://www.rfc-editor.org/rfc/rfc7468↗.

Property	Type	Description
`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}/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	`ConnectorList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new Connector

Query parameters

Parameter	Type	Description
`dryRun`	`string`	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
`fieldValidation`	`string`	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 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

HTTP code	Response body
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
`dryRun`	`string`	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	`Status↗` schema
202 - Accepted	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified Connector

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified Connector

Query parameters

Parameter	Type	Description
`dryRun`	`string`	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
`fieldValidation`	`string`	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 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

Parameter	Type	Description
		will contain all unknown and duplicate fields encountered.

HTTP responses

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

HTTP method

`PUT`

Description

replace the specified Connector

Query parameters

Parameter	Type	Description
`dryRun`	`string`	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
`fieldValidation`	`string`	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 in v1.23+ - Strict: This will fail the

Parameter	Type	Description
		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

/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	`Connector` schema

HTTP code	Response body
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified Connector

Query parameters

Parameter	Type	Description
`dryRun`	`string`	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
`fieldValidation`	`string`	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 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	`Connector` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified Connector

Query parameters

Parameter	Type	Description
`dryRun`	`string`	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
`fieldValidation`	`string`	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 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.katanomi.dev/v1alpha1]

Description

ConnectorClass is the Schema for the connectorclasses API

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	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 ↗
`kind`	`string`	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:

Property	Type	Description
		https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	Spec defines the desired state of ConnectorClass
`status`	`object`	Status defines the actual state of ConnectorClass

.spec

Description

Spec defines the desired state of ConnectorClass

Type

`object`

Property	Type	Description
`address`	`object`	Address indicates address param constraints for this ConnectorClass of connectors we only support string param type
`api`	`object`	API defines connectorclass plugin api address `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

Property	Type	Description
		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
`auth`	`object`	Auth indicates authentication constraints for this ConnectorClass of connectors
`authProbes`	`array`	AuthProbes defines authentication probe for this ConnectorClass of connectors
`configurations`	`array`	Configurations defines connectorclass configuration
`livenessProbe`	`object`	LivenessProbe defines liveness probe for this ConnectorClass of connectors
`proxy`	`object`	Proxy defines the proxy configuration for this ConnectorClass. Specifies how network traffic should be 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`	`object`	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.address.default

Description

Default is the value a parameter takes if no input value is supplied.

Type

`object`

Required

`arrayVal` `objectVal` `stringVal` `type`

Property	Type	Description
`arrayVal`	`array`	
`objectVal`	`object`	
`stringVal`	`string`	
`type`	`string`	ParamType indicates the type of an input parameter; Used to distinguish between a single string and an array of strings.

.spec.address.default.arrayVal

Type

`array`

.spec.address.default.arrayVal[]

Type

`string`

.spec.address.default.objectVal

Type

`object`

.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 `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.
`ref`	`object`	Ref points to an Addressable.
`uri`	`string`	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.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 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.

.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
`description`	`string`	Description is the description of the AuthType
`displayName`	`string`	DisplayName is the human readable name of the AuthType
`generator`	`object`	Generator specifies how to generate authentication data dynamically. Can be used to implement custom authentication logic.
`name`	`string`	Name of the AuthType Must be unique within the ConnectorClass.
`optional`	`boolean`	Optional indicates whether the authentication information is optional for this ConnectorClass of connectors the default value is false
`params`	`array`	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.
`secretType`	`string`	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
`rego`	`string`	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: 1. Define its rules under the 'proxy' package 2. 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`	`object`	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[].default

Description

Default is the value a parameter takes if no input value is supplied.

Type

`object`

Required

`arrayVal` `objectVal` `stringVal` `type`

Property	Type	Description
`arrayVal`	`array`	
`objectVal`	`object`	
`stringVal`	`string`	
`type`	`string`	ParamType indicates the type of an input parameter; Used to distinguish between a single string and an array of strings.

.spec.auth.types[].params[].default.arrayVal

Type

`array`

.spec.auth.types[].params[].default.arrayVal[]

Type

`string`

.spec.auth.types[].params[].default.objectVal

Type

`object`

.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 `spec.auth.types[].name`, 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`	`object`	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[].default

Description

Default is the value a parameter takes if no input value is supplied.

Type

`object`

Required

`arrayVal` `objectVal` `stringVal` `type`

Property	Type	Description
`arrayVal`	`array`	
`objectVal`	`object`	
`stringVal`	`string`	
`type`	`string`	ParamType indicates the type of an input parameter; Used to distinguish between a single string and an array of strings.

.spec.authProbes[].params[].default.arrayVal

Type

`array`

.spec.authProbes[].params[].default.arrayVal[]

Type

`string`

.spec.authProbes[].params[].default.objectVal

Type

`object`

.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
`http`	`object`	Http represents the network detection using the http get method More Info: TODO:

.spec.authProbes[].probe.http

Description

Http represents the network detection using the http get method More Info: TODO:

Type

`object`

Required

`path`

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

Property	Type	Description
`method`	`string`	Method represents the HTTP method to use for the probe, support method: GET, POST. default is GET
`path`	`string`	Path represents the API address accessed during the current detection
`scheme`	`string`	Scheme to use for connecting to the host. If empty: • 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.
`value`	`string`	The header field value

.spec.configurations

Description

Configurations defines connectorclass configuration

Type

`array`

.spec.configurations[]

Description

ConnectorClassConfiguration defines connectorclass configuration

Type

`object`

Property	Type	Description
`annotations`	`object`	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

Property	Type	Description
		should be preserved when modifying objects. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations ↗
`data`	`object`	Data contains the configuration data. Each key must consist of alphanumeric characters, '-', '_' or '.'.
`name`	`string`	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
`http`	`object`	Http represents the network detection using the http get method More Info: TODO:

.spec.livenessProbe.http

Description

Http represents the network detection using the http get method More Info: TODO:

Type

`object`

Required

`path`

Property	Type	Description
`disableRedirect`	`boolean`	DisableRedirect indicates whether the probe should follow redirects, default is false
`host`	`string`	Host represents the tool address for the current detection. usually, it would be empty, it will use the `spec.address` of connector

Property	Type	Description
`httpHeaders`	`array`	Custom headers to set in the request. HTTP allows repeated headers.
`method`	`string`	Method represents the HTTP method to use for the probe, support method: GET, POST. default is GET
`path`	`string`	Path represents the API address accessed during the current detection
`scheme`	`string`	Scheme to use for connecting to the host. If empty: • 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.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
`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.

Property	Type	Description
`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.
`ref`	`object`	Ref points to an Addressable.
`uri`	`string`	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.ref

Description

Ref points to an Addressable.

Type

`object`

Required

`kind` `name`

Property	Type	Description
`address`	`string`	Address points to a specific Address Name.

Property	Type	Description
`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 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 convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.
`api`	`object`	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.
`conditions`	`array`	Conditions the latest available observations of a resource's current state.
`observedGeneration`	`integer`	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.
`proxy`	`object`	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.

.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.
`name`	`string`	Name is the name of the address.
`url`	`string`	

.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
`lastTransitionTime`	`string`	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).
`message`	`string`	A human readable message indicating details about the transition.
`reason`	`string`	The reason for the condition's last transition.
`severity`	`string`	Severity with which to treat failures of this type of condition. When this is not specified, it defaults to Error.
`status`	`string`	Status of the condition, one of True, False, Unknown.
`type`	`string`	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

HTTP code	Response body
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind ConnectorClass

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorClassList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new ConnectorClass

Query parameters

Parameter	Type	Description
`dryRun`	`string`	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
`fieldValidation`	`string`	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.

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
`dryRun`	`string`	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	`Status↗` schema
202 - Accepted	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified ConnectorClass

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorClass

Query parameters

Parameter	Type	Description
`dryRun`	`string`	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
`fieldValidation`	`string`	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 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	`ConnectorClass` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorClass

Query parameters

Parameter	Type	Description
`dryRun`	`string`	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
`fieldValidation`	`string`	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 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

/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	`ConnectorClass` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorClass

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further

Parameter	Type	Description
		processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	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 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	`ConnectorClass` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorClass

Query parameters

Parameter	Type	Description
`dryRun`	`string`	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
`fieldValidation`	`string`	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 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

HTTP code	Response body
401 - Unauthorized	Empty