Connector APIs

Connector \[accesspolicies.alau Connector \[accessrequests.ala Connector \

ConnectorClass \[connectors.a Connector \[resourceinterfaces.alauda.io/v1al

Connector [accesspolicies.alauda.io/v1alpha1]

Description

AccessPolicy defines the access strategy for Connectors in a namespace. It specifies which Connectors are covered and what permissions are granted, either automatically (defaultPermission) or after approval checks pass (checkGrantedPermission).

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

Property	Type	Description
		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 ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	AccessPolicySpec defines the desired state of AccessPolicy.
`status`	`object`	AccessPolicyStatus defines the observed state of AccessPolicy.

.spec

Description

AccessPolicySpec defines the desired state of AccessPolicy.

Type

`object`

Property	Type	Description
`checkGrantedPermission`	`object`	CheckGrantedPermission defines permissions granted only after approval checks pass.

Property	Type	Description
`connector`	`object`	Connector specifies which Connectors this policy applies to. If empty, the policy applies to all Connectors in the namespace.
`defaultPermission`	`object`	DefaultPermission defines the Role and RoleBinding automatically granted without any approval check.

.spec.checkGrantedPermission

Description

CheckGrantedPermission defines permissions granted only after approval checks pass.

Type

`object`

Required

`spec`

Property	Type	Description
`spec`	`object`	Spec contains the check rules and the permissions to grant after all checks pass.

.spec.checkGrantedPermission.spec

Description

Spec contains the check rules and the permissions to grant after all checks pass.

Type

`object`

Required

`checks` `roleTemplate`

Property	Type	Description
`checks`	`array`	Checks is the list of approval check rules.
`roleTemplate`	`object`	RoleTemplate defines the rules for the generated Role.

.spec.checkGrantedPermission.spec.checks

Description

Checks is the list of approval check rules.

Type

`array`

.spec.checkGrantedPermission.spec.checks[]

Description

CheckRule defines a check rule that must pass for a permission to be granted. it contains either a reference to a CheckRuleSpec stored in a ConfigMap or the CheckRuleSpec itself. you can specify either Ref or Spec, but not both.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the identifier of this check rule, referenced in AccessRequest status.
`ref`	`object`	Ref is a reference to a CheckRuleSpec stored in a ConfigMap.
`spec`	`object`	Spec contains the check rule specification.

.spec.checkGrantedPermission.spec.checks[].ref

Description

Ref is a reference to a CheckRuleSpec stored in a ConfigMap.

Type

`object`

Required

`configMap`

Property	Type	Description
`configMap`	`object`	ConfigMap references the ConfigMap containing the CheckRuleSpec.

.spec.checkGrantedPermission.spec.checks[].ref.configMap

Description

ConfigMap references the ConfigMap containing the CheckRuleSpec.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.checkGrantedPermission.spec.checks[].spec

Description

Spec contains the check rule specification.

Type

`object`

Required

`selector`

Property	Type	Description
`selector`	`object`	Selector specifies how to find the Check Duck Type resource.
`state`	`object`	State configures how the check result is computed. If empty, the default duck-type field status.state is used.

.spec.checkGrantedPermission.spec.checks[].spec.selector

Description

Selector specifies how to find the Check Duck Type resource.

Type

`object`

Required

`objectRef`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.
`objectRef`	`object`	ObjectRef specifies the reference to the object to check against. kind and apiVersion are required to distinguish different duck types

.spec.checkGrantedPermission.spec.checks[].spec.selector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.checkGrantedPermission.spec.checks[].spec.selector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.checkGrantedPermission.spec.checks[].spec.selector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.checkGrantedPermission.spec.checks[].spec.selector.matchExpressions[].values[]

Type

`string`

.spec.checkGrantedPermission.spec.checks[].spec.selector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.checkGrantedPermission.spec.checks[].spec.selector.objectRef

Description

ObjectRef specifies the reference to the object to check against. kind and apiVersion are required to distinguish different duck types

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 ↗

Property	Type	Description
`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.checkGrantedPermission.spec.checks[].spec.state

Description

State configures how the check result is computed. If empty, the default duck-type field status.state is used.

Type

`object`

Property	Type	Description
`rego`	`string`	Rego is an OPA Rego script (package "approval") that receives the full check resource as input and must output status = {"state": "approved|rejected|pending|passed"}. If empty, the default duck-type field status.state is used.

.spec.checkGrantedPermission.spec.roleTemplate

Description

RoleTemplate defines the rules for the generated Role.

Type

`object`

Property	Type	Description
`ref`	`object`	Ref specifies a reference to a RoleTemplate

.spec.checkGrantedPermission.spec.roleTemplate.ref

Description

Ref specifies a reference to a RoleTemplate

Type

`object`

Property	Type	Description
`configMap`	`object`	ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of

Property	Type	Description
		rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

.spec.checkGrantedPermission.spec.roleTemplate.ref.configMap

Description

ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.connector

Description

Connector specifies which Connectors this policy applies to. If empty, the policy applies to all Connectors in the namespace.

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.
`names`	`array`	Names is an explicit list of resource names to match.

.spec.connector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.connector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.connector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.connector.matchExpressions[].values[]

Type

`string`

.spec.connector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.connector.names

Description

Names is an explicit list of resource names to match.

Type

`array`

.spec.connector.names[]

Type

`string`

.spec.defaultPermission

Description

DefaultPermission defines the Role and RoleBinding automatically granted without any approval check.

Type

`object`

Required

`bindingTemplate` `roleTemplate`

Property	Type	Description
`bindingTemplate`	`object`	BindingTemplate defines the subjects for the generated RoleBinding.
`roleTemplate`	`object`	RoleTemplate defines the rules to include in the generated Role.

.spec.defaultPermission.bindingTemplate

Description

BindingTemplate defines the subjects for the generated RoleBinding.

Type

`object`

Property	Type	Description
`serviceAccounts`	`array`	ServiceAccounts is the list of service account templates to bind.

.spec.defaultPermission.bindingTemplate.serviceAccounts

Description

ServiceAccounts is the list of service account templates to bind.

Type

`array`

.spec.defaultPermission.bindingTemplate.serviceAccounts[]

Description

ServiceAccountTemplate defines a template for binding ServiceAccounts. it extends rbacv1.Subject with dynamic label-based selectors.

Type

`object`

Property	Type	Description
`names`	`array`	Names is the list of service account names to bind.
`namespaceSelector`	`object`	NamespaceSelector selects Namespaces by label and/or name.

.spec.defaultPermission.bindingTemplate.serviceAccounts[].names

Description

Names is the list of service account names to bind.

Type

`array`

.spec.defaultPermission.bindingTemplate.serviceAccounts[].names[]

Type

`string`

.spec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector

Description

NamespaceSelector selects Namespaces by label and/or name.

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.
`names`	`array`	Names is an explicit list of resource names to match.

.spec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This

array is replaced during a strategic merge patch.

Type

`array`

.spec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.matchExpressions[].values[]

Type

`string`

.spec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.names

Description

Names is an explicit list of resource names to match.

Type

`array`

.spec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.names[]

Type

`string`

.spec.defaultPermission.roleTemplate

Description

RoleTemplate defines the rules to include in the generated Role.

Type

`object`

Property	Type	Description
`ref`	`object`	Ref specifies a reference to a RoleTemplate

.spec.defaultPermission.roleTemplate.ref

Description

Ref specifies a reference to a RoleTemplate

Type

`object`

Property	Type	Description
`configMap`	`object`	ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

.spec.defaultPermission.roleTemplate.ref.configMap

Description

ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.status

Description

AccessPolicyStatus defines the observed state of AccessPolicy.

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.
`matchedConnectors`	`array`	MatchedConnectors records the Connector names matched by spec.connector.
`observedGeneration`	`integer`	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.

.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.matchedConnectors

Description

MatchedConnectors records the Connector names matched by spec.connector.

Type

`array`

.status.matchedConnectors[]

Description

LocalObjectReference contains enough information to let you locate the referenced object inside the same namespace.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

API Endpoints

The following API endpoints are available:

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

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

HTTP method

`DELETE`

Description

delete collection of AccessPolicy

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 AccessPolicy

HTTP responses

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

HTTP method

`POST`

Description

create a new AccessPolicy

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`	`AccessPolicy` schema	`application/json` formatted

HTTP responses

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

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

HTTP method

`DELETE`

Description

delete the specified AccessPolicy

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 AccessPolicy

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified AccessPolicy

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	`AccessPolicy` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified AccessPolicy

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`	`AccessPolicy` schema	`application/json` formatted

HTTP responses

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

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

HTTP method

`GET`

Description

read status of the specified AccessPolicy

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified AccessPolicy

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	`AccessPolicy` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified AccessPolicy

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`	`AccessPolicy` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`AccessPolicy` schema
201 - Created	`AccessPolicy` schema

HTTP code	Response body
401 - Unauthorized	Empty

Connector [accessrequests.alauda.io/v1alpha1]

Description

AccessRequest represents a subject's access application for a specific Connector, scoped to the lifecycle of a context object (Pod). It tracks matched AccessPolicies, approval check states, and authorization status via conditions.

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

Property	Type	Description
		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 ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	AccessRequestSpec defines the desired state of AccessRequest.
`status`	`object`	AccessRequestStatus records the observed state of AccessRequest.

.spec

Description

AccessRequestSpec defines the desired state of AccessRequest.

Type

`object`

Required

`connectorRef` `context` `subject`

Property	Type	Description
`connectorRef`	`object`	ConnectorRef references the target Connector in the same namespace. Only Name is required; Namespace is

Property	Type	Description
		always the same as the AccessRequest.
`context`	`object`	Context provides lifecycle context for this request. Currently only Kind=Pod is supported.
`subject`	`object`	Subject is the identity requesting access (typically a ServiceAccount).

.spec.connectorRef

Description

ConnectorRef references the target Connector in the same namespace. Only Name is required; Namespace is always the same as the AccessRequest.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.context

Description

Context provides lifecycle context for this request. Currently only Kind=Pod is supported.

Type

`object`

Required

`objectRef`

Property	Type	Description
`objectRef`	`object`	ObjectRef points to the lifecycle object (e.g., a Pod). Currently only Kind=Pod is supported.

.spec.context.objectRef

Description

ObjectRef points to the lifecycle object (e.g., a Pod). Currently only Kind=Pod is supported.

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

Property	Type	Description
		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.subject

Description

Subject is the identity requesting access (typically a ServiceAccount).

Type

`object`

Required

`kind` `name`

Property	Type	Description
`apiGroup`	`string`	APIGroup holds the API group of the referenced subject. Defaults to "" for ServiceAccount subjects. Defaults to "rbac.authorization.k8s.io" for User and Group subjects.
`kind`	`string`	Kind of object being referenced. Values defined by this API group are "User", "Group", and "ServiceAccount". If the Authorizer does not recognized the kind value, the Authorizer should report an error.
`name`	`string`	Name of the object being referenced.
`namespace`	`string`	Namespace of the referenced object. If the object kind is non-namespace, such as "User" or "Group", and this value is not empty the Authorizer should report an error.

.status

Description

AccessRequestStatus records the observed state of AccessRequest.

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.
`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.
`policies`	`array`	Policies holds the matched AccessPolicy status list. Full AccessPolicy snapshots are stored to prevent policy changes from affecting in-flight authorization decisions.

.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.

Property	Type	Description
`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.policies

Description

Policies holds the matched AccessPolicy status list. Full AccessPolicy snapshots are stored to prevent policy changes from affecting in-flight authorization decisions.

Type

`array`

.status.policies[]

Description

AccessPolicyMatchedStatus records a matched AccessPolicy and its check results.

Type

`object`

Required

`name` `policySpec`

Property	Type	Description
`matchedChecks`	`array`	MatchedChecks records the matched Check Duck Type resources and their states.
`name`	`string`	Name is the AccessPolicy name, used as the list map key.
`permissionSync`	`object`	PermissionSync records policy-level permission synchronization condition.
`policySpec`	`object`	PolicySpec is the full AccessPolicy spec snapshot at match time.

.status.policies[].matchedChecks

Description

MatchedChecks records the matched Check Duck Type resources and their states.

Type

`array`

.status.policies[].matchedChecks[]

Description

MatchedCheck records one matched Check Duck Type resource instance.

Type

`object`

Required

`condition` `name` `ref`

Property	Type	Description
`condition`	`object`	Condition records the computed approval condition of this check.
`name`	`string`	Name matches CheckRule.name in the AccessPolicy.
`ref`	`object`	Ref identifies the matched Check Duck Type resource instance.

.status.policies[].matchedChecks[].condition

Description

Condition records the computed approval condition of this check.

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).

Property	Type	Description
`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.policies[].matchedChecks[].ref

Description

Ref identifies the matched Check Duck Type resource instance.

Type

`object`

Property	Type	Description
`apiVersion`	`string`	API version of the referent.

Property	Type	Description
`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/ ↗
`resourceVersion`	`string`	Specific resourceVersion to which this reference is made, if any. More info: https://git.k8s.io/community/contributors/devel/sig-

Property	Type	Description
		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.policies[].permissionSync

Description

PermissionSync records policy-level permission synchronization condition.

Type

`object`

Required

`lastTransitionTime` `message` `reason` `status` `type`

Property	Type	Description
`lastTransitionTime`	`string`	lastTransitionTime is the last time the condition transitioned from one status to another. This should be when the underlying condition changed. If that is not known, then using the time when the API field changed is acceptable.
`message`	`string`	message is a human readable message indicating details about the transition. This may be an empty string.

Property	Type	Description
`observedGeneration`	`integer`	observedGeneration represents the .metadata.generation that the condition was set based upon. For instance, if .metadata.generation is currently 12, but the .status.conditions[x].observedGeneration is 9, the condition is out of date with respect to the current state of the instance.
`reason`	`string`	reason contains a programmatic identifier indicating the reason for the condition's last transition. Producers of specific condition types may define expected values and meanings for this field, and whether the values are considered a guaranteed API. The value should be a CamelCase string. This field may not be empty.
`status`	`string`	status of the condition, one of True, False, Unknown.
`type`	`string`	type of condition in CamelCase or in foo.example.com/CamelCase.

.status.policies[].policySpec

Description

PolicySpec is the full AccessPolicy spec snapshot at match time.

Type

`object`

Property	Type	Description
`checkGrantedPermission`	`object`	CheckGrantedPermission defines permissions granted only after approval checks pass.
`connector`	`object`	Connector specifies which Connectors this policy applies to. If empty, the policy applies to all Connectors in the namespace.
`defaultPermission`	`object`	DefaultPermission defines the Role and RoleBinding automatically granted without any approval check.

.status.policies[].policySpec.checkGrantedPermission

Description

CheckGrantedPermission defines permissions granted only after approval checks pass.

Type

`object`

Required

`spec`

Property	Type	Description
`spec`	`object`	Spec contains the check rules and the permissions to grant after all checks pass.

.status.policies[].policySpec.checkGrantedPermission.spec

Description

Spec contains the check rules and the permissions to grant after all checks pass.

Type

`object`

Required

`checks` `roleTemplate`

Property	Type	Description
`checks`	`array`	Checks is the list of approval check rules.
`roleTemplate`	`object`	RoleTemplate defines the rules for the generated Role.

.status.policies[].policySpec.checkGrantedPermission.spec.checks

Description

Checks is the list of approval check rules.

Type

`array`

.status.policies[].policySpec.checkGrantedPermission.spec.checks[]

Description

CheckRule defines a check rule that must pass for a permission to be granted. it contains either a reference to a CheckRuleSpec stored in a ConfigMap or the CheckRuleSpec itself. you can specify either Ref or Spec, but not both.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the identifier of this check rule, referenced in AccessRequest status.
`ref`	`object`	Ref is a reference to a CheckRuleSpec stored in a ConfigMap.
`spec`	`object`	Spec contains the check rule specification.

.status.policies[].policySpec.checkGrantedPermission.spec.checks[].ref

Description

Ref is a reference to a CheckRuleSpec stored in a ConfigMap.

Type

`object`

Required

`configMap`

Property	Type	Description
`configMap`	`object`	ConfigMap references the ConfigMap containing the CheckRuleSpec.

.status.policies[].policySpec.checkGrantedPermission.spec.checks[].ref.configMap

Description

ConfigMap references the ConfigMap containing the CheckRuleSpec.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.status.policies[].policySpec.checkGrantedPermission.spec.checks[].spec

Description

Spec contains the check rule specification.

Type

`object`

Required

`selector`

Property	Type	Description
`selector`	`object`	Selector specifies how to find the Check Duck Type resource.
`state`	`object`	State configures how the check result is computed. If empty, the default duck-type field status.state is used.

.status.policies[].policySpec.checkGrantedPermission.spec.checks[].spec.selector

Description

Selector specifies how to find the Check Duck Type resource.

Type

`object`

Required

`objectRef`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is

Property	Type	Description
		"key", the operator is "In", and the values array contains only "value". The requirements are ANDed.
`objectRef`	`object`	ObjectRef specifies the reference to the object to check against. kind and apiVersion are required to distinguish different duck types

.status.policies[].policySpec.checkGrantedPermission.spec.checks[].spec.selector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.status.policies[].policySpec.checkGrantedPermission.spec.checks[].spec.selector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.status.policies[].policySpec.checkGrantedPermission.spec.checks[].spec.selector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.status.policies[].policySpec.checkGrantedPermission.spec.checks[].spec.selector.matchExpressions[].values[]

Type

`string`

.status.policies[].policySpec.checkGrantedPermission.spec.checks[].spec.selector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.status.policies[].policySpec.checkGrantedPermission.spec.checks[].spec.selector.objectRef

Description

ObjectRef specifies the reference to the object to check against. kind and apiVersion are required to distinguish different duck types

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

Property	Type	Description
		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 ↗

.status.policies[].policySpec.checkGrantedPermission.spec.checks[].spec.state

Description

State configures how the check result is computed. If empty, the default duck-type field status.state is used.

Type

`object`

Property	Type	Description
`rego`	`string`	Rego is an OPA Rego script (package "approval") that receives the full check resource as input and must output status = {"state": "approved|rejected|pending|passed"}. If empty, the default duck-type field status.state is used.

.status.policies[].policySpec.checkGrantedPermission.spec.roleTemplate

Description

RoleTemplate defines the rules for the generated Role.

Type

`object`

Property	Type	Description
`ref`	`object`	Ref specifies a reference to a RoleTemplate

.status.policies[].policySpec.checkGrantedPermission.spec.roleTemplate.ref

Description

Ref specifies a reference to a RoleTemplate

Type

`object`

Property	Type	Description
`configMap`	`object`	ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

.status.policies[].policySpec.checkGrantedPermission.spec.roleTemplate.ref.configMap

Description

ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info:

Property	Type	Description
		https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.status.policies[].policySpec.connector

Description

Connector specifies which Connectors this policy applies to. If empty, the policy applies to all Connectors in the namespace.

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.
`names`	`array`	Names is an explicit list of resource names to match.

.status.policies[].policySpec.connector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.status.policies[].policySpec.connector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.status.policies[].policySpec.connector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.status.policies[].policySpec.connector.matchExpressions[].values[]

Type

`string`

.status.policies[].policySpec.connector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.status.policies[].policySpec.connector.names

Description

Names is an explicit list of resource names to match.

Type

`array`

.status.policies[].policySpec.connector.names[]

Type

`string`

.status.policies[].policySpec.defaultPermission

Description

DefaultPermission defines the Role and RoleBinding automatically granted without any approval check.

Type

`object`

Required

`bindingTemplate` `roleTemplate`

Property	Type	Description
`bindingTemplate`	`object`	BindingTemplate defines the subjects for the generated RoleBinding.
`roleTemplate`	`object`	RoleTemplate defines the rules to include in the generated Role.

.status.policies[].policySpec.defaultPermission.bindingTemplate

Description

BindingTemplate defines the subjects for the generated RoleBinding.

Type

`object`

Property	Type	Description
`serviceAccounts`	`array`	ServiceAccounts is the list of service account templates to bind.

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts

Description

ServiceAccounts is the list of service account templates to bind.

Type

`array`

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts[]

Description

ServiceAccountTemplate defines a template for binding ServiceAccounts. it extends rbacv1.Subject with dynamic label-based selectors.

Type

`object`

Property	Type	Description
`names`	`array`	Names is the list of service account names to bind.
`namespaceSelector`	`object`	NamespaceSelector selects Namespaces by label and/or name.

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts[].names

Description

Names is the list of service account names to bind.

Type

`array`

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts[].names[]

Type

`string`

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector

Description

NamespaceSelector selects Namespaces by label and/or name.

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is

Property	Type	Description
		"key", the operator is "In", and the values array contains only "value". The requirements are ANDed.
`names`	`array`	Names is an explicit list of resource names to match.

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.matchExpressions[].values[]

Type

`string`

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.names

Description

Names is an explicit list of resource names to match.

Type

`array`

.status.policies[].policySpec.defaultPermission.bindingTemplate.serviceAccounts[].namespaceSelector.names[]

Type

`string`

.status.policies[].policySpec.defaultPermission.roleTemplate

Description

RoleTemplate defines the rules to include in the generated Role.

Type

`object`

Property	Type	Description
`ref`	`object`	Ref specifies a reference to a RoleTemplate

.status.policies[].policySpec.defaultPermission.roleTemplate.ref

Description

Ref specifies a reference to a RoleTemplate

Type

`object`

Property	Type	Description
`configMap`	`object`	ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

.status.policies[].policySpec.defaultPermission.roleTemplate.ref.configMap

Description

ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

API Endpoints

The following API endpoints are available:

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

- `PATCH`: partially update status of the specified AccessRequest
- `PUT`: replace status of the specified AccessRequest

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

HTTP method

`DELETE`

Description

delete collection of AccessRequest

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind AccessRequest

HTTP responses

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

HTTP method

`POST`

Description

create a new AccessRequest

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`	`AccessRequest` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`AccessRequest` schema

HTTP code	Response body
201 - Created	`AccessRequest` schema
202 - Accepted	`AccessRequest` schema
401 - Unauthorized	Empty

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

HTTP method

`DELETE`

Description

delete the specified AccessRequest

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 AccessRequest

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified AccessRequest

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.

HTTP responses

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

HTTP method

`PUT`

Description

replace the specified AccessRequest

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

Parameter	Type	Description
		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`	`AccessRequest` schema	`application/json` formatted

HTTP responses

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

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

HTTP method

`GET`

Description

read status of the specified AccessRequest

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified AccessRequest

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	`AccessRequest` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified AccessRequest

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

Parameter	Type	Description
`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`	`AccessRequest` schema	`application/json` formatted

HTTP responses

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

Connector [connectors.alauda.io/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 ↗

Property	Type	Description
`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: 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

Property	Type	Description
`auth`	`object`	Auth represents authenticate data of current connector
`connectorClassName`	`string`	ConnectorClassName of current connector
`params`	`array`	Params information required for spec

.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`	`array`	Params information required for authentication
`secretRef`	`object`	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.

Property	Type	Description
`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/ ↗
`resourceVersion`	`string`	Specific resourceVersion to which this reference is made, if any. More info: https://git.k8s.io/community/contributors/devel/sig-

Property	Type	Description
		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
`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 contains the status information for the connector's api
`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.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
`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.alauda.io/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 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

Property	Type	Description
		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
`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
`params`	`array`	Params indicates param constraints for this ConnectorClass of connectors we only support string param type
`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
`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.
`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
`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.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.

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 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.

Property	Type	Description
`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.
`permissions`	`object`	Permissions defines the request matching rules for different operations on the connectorclass api
`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.permissions

Description

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

Type

`object`

Property	Type	Description
`rego`	`string`	Rego contains the Rego policy script that maps API requests to permission verbs. The script evaluates incoming requests and determines the required permission level. Requirements: 1. Must be defined under the 'permissions' package 2. Must produce a 'result' object matching the PermissionMappingResult structure Input Variables: - 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) Output: - result.verb (string): Permission verb - "read", "write", "delete", or "" (default to http verb mapping) Example: package permissions import future.keywords.if result = { "verb": "read" } if { startswith(input.request.path, "/search") input.request.method == "POST" } else = { "verb": "" }

.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 ↗

Property	Type	Description
`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
`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.
`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.

Property	Type	Description
`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[].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.

Type

`object`

.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
`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.
`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[].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.

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
`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.
`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
`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[].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.

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
`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 `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`).
`http`	`object`	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
`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 resolved from `spec.api` (e.g., `spec.api.uri`) 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
`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

Property	Type	Description
		`spec.address` of connector
`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 it is relative path, it will be appended to the host of URI resolved from `spec.address` of connector when execute the probe
`response`	`object`	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.
`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.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`	CEL contains a CEL (Common Expression Language) expression for response validation. The expression must evaluate to a boolean value. Available variables: • 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 Example expressions: • 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') CEL is recommended for simple to medium complexity validations: • Intuitive syntax similar to JavaScript/C/Python • Better performance for simple expressions • Native support in Kubernetes ecosystem • Lower learning curve

Property	Type	Description
		When both CEL and built-in rules (Contains, Regex, JSONPath) are specified, all rules must pass (AND logic).

.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 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 '.'.

Property	Type	Description
`name`	`string`	Name of the configuration
`params`	`array`	Params declares the parameters this configuration accepts from configuration.params at CSI mount time. Each entry is validated and defaults are applied before template rendering. A param with Default == nil is considered required; mount fails if absent.

.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.configurations[].params

Description

Params declares the parameters this configuration accepts from configuration.params at CSI mount time. Each entry is validated and defaults are applied before template rendering. A param with Default == nil is considered required; mount fails if absent.

Type

`array`

.spec.configurations[].params[]

Description

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

Type

`object`

Required

`name`

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.
`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.

Property	Type	Description
`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.configurations[].params[].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.

Type

`object`

.spec.configurations[].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.configurations[].params[].enum[]

Type

`string`

.spec.configurations[].params[].properties

Description

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

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 `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`).
`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 resolved from `spec.api` (e.g., `spec.api.uri`) 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
`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 it is relative path, it will be appended to the host of URI resolved from `spec.address` of connector when execute the probe
`response`	`object`	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.
`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.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:

Property	Type	Description
		• 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 Example expressions: • 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') CEL is recommended for simple to medium complexity validations: • Intuitive syntax similar to JavaScript/C/Python • Better performance for simple expressions • Native support in Kubernetes ecosystem • Lower learning curve When both CEL and built-in rules (Contains, Regex, JSONPath) are specified, all rules must pass (AND logic).

.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
`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.
`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.

Property	Type	Description
`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[].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.

Type

`object`

.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
`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.
`authExtractor`	`object`	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.
`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.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`	`string`	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: 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

Property	Type	Description
		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

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.
`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↗.

Property	Type	Description
`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

Property	Type	Description
		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
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 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`	`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

HTTP code	Response body
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 processing of the request. Valid values are: - All: all dry run stages will be processed

Parameter	Type	Description
`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

Parameter	Type	Description
		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

Connector [resourceinterfaces.alauda.io/v1alpha1]

Description

ResourceInterface is the Schema for the resourceinterfaces API ResourceInterface defines a reusable template for creating PipelineIntegrations with specific parameters and outputs. It supports inheritance through the extends field and provides a way to standardize resource definitions across different connector implementations.

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 ↗

Property	Type	Description
`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: 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 ResourceInterface

.spec

Description

Spec defines the desired state of ResourceInterface

Type

`object`

Property	Type	Description
`attributes`	`array`	Attributes defines the attributes that will be available when creating PipelineIntegration instances.
`configurations`	`object`	Configuration defines the additional configuration for this ResourceInterface, which will be passed to the PipelineIntegration instance as annotations. Each key

Property	Type	Description
		maps to a configuration entry containing a value and optional metadata.
`params`	`array`	Params defines the parameters required to construct a PipelineIntegration. These parameters will be used when creating PipelineIntegration instances.
`workspaces`	`array`	Workspaces defines the workspaces that will be available when creating PipelineIntegration instances.

.spec.attributes

Description

Attributes defines the attributes that will be available when creating PipelineIntegration instances.

Type

`array`

.spec.attributes[]

Description

ResourceInterfaceAttributeSpec defines attribute specification for ResourceInterface

Type

`object`

Required

`name`

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.
`default`		Default is the value a parameter takes if no input value is supplied.
`dependsOn`	`array`	DependsOn defines the dependencies of this parameter. it could be the name of other parameters or connectors
`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.
`expression`	`string`	Expression defines a custom expression to compute the value. The expression can access connector and params of current PipelineIntegration.
`name`	`string`	Name declares the name by which a parameter is referenced.

Property	Type	Description
`parameterize`	`object`	Parameterize defines the parameterization configuration for this parameter.
`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.attributes[].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.

Type

`object`

.spec.attributes[].dependsOn

Description

DependsOn defines the dependencies of this parameter. it could be the name of other parameters or connectors

Type

`array`

.spec.attributes[].dependsOn[]

Type

`string`

.spec.attributes[].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.attributes[].enum[]

Type

`string`

.spec.attributes[].parameterize

Description

Parameterize defines the parameterization configuration for this parameter.

Type

`object`

Property	Type	Description
`disable`	`boolean`	Disable indicates whether parameterization is disabled for this parameter/workspace.

Property	Type	Description
`name`	`string`	Name is the default parameter name when this parameter/workspace is parameterized during PipelineResource construction.

.spec.attributes[].properties

Description

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

Type

`object`

.spec.configurations

Description

Configuration defines the additional configuration for this ResourceInterface, which will be passed to the PipelineIntegration instance as annotations. Each key maps to a configuration entry containing a value and optional metadata.

Type

`object`

Property	Type	Description
`multipleMergeConnectors`	`object`	MultipleMergeConnectors defines the configuration for enabling multiple connectors merge in PipelineIntegration. When this configuration is enabled, multiple connectors will be merged when creating PipelineIntegration.

.spec.configurations.multipleMergeConnectors

Description

MultipleMergeConnectors defines the configuration for enabling multiple connectors merge in PipelineIntegration. When this configuration is enabled, multiple connectors will be merged when creating PipelineIntegration.

Type

`object`

Property	Type	Description
`description`	`string`	Description provides a human-readable description of this configuration entry.
`enabled`	`boolean`	Enabled indicates whether this configuration is enabled.

.spec.params

Description

Params defines the parameters required to construct a PipelineIntegration. These parameters will be used when creating PipelineIntegration instances.

Type

`array`

.spec.params[]

Description

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

Type

`object`

Required

`name`

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.
`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.

Property	Type	Description
`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[].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.

Type

`object`

.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.workspaces

Description

Workspaces defines the workspaces that will be available when creating PipelineIntegration instances.

Type

`array`

.spec.workspaces[]

Description

ResourceInterfaceWorkspaceSpec defines a workspace specification for ResourceInterface

Type

`object`

Required

`name`

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. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations ↗
`description`	`string`	Description provides a human-readable description of the workspace.

Property	Type	Description
`name`	`string`	Name of the workspace.
`value`	`object`	Value defines the volume source for this workspace.
`workspaceMapping`	`object`	WorkspaceMapping defines the workspace mapping configuration for this workspace.

.spec.workspaces[].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. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations

Type

`object`

.spec.workspaces[].value

Description

Value defines the volume source for this workspace.

Type

`object`

Property	Type	Description
`configMap`	`object`	ConfigMap represents a configMap that should popu workspace.
`csi`	`object`	CSI (Container Storage Interface) represents ephem storage that is handled by certain external CSI drive
`emptyDir`	`object`	EmptyDir represents a temporary directory that shar Task's lifetime. More info: https://kubernetes.io/docs/concepts/storage/volumes ↗ Either this OR PersistentVolumeClaim can be use
`persistentVolumeClaim`	`object`	PersistentVolumeClaimVolumeSource represents a to a PersistentVolumeClaim in the same namespace this OR EmptyDir can be used.
`projected`	`object`	Projected represents a projected volume that should this workspace.
`secret`	`object`	Secret represents a secret that should populate this workspace.
`subPath`	`string`	SubPath is optionally a directory on the volume whic be used for this binding (i.e. the volume will be mour sub directory).

Property	Type	Description
`volumeClaimTemplate`	`object`	VolumeClaimTemplate is a template for a claim that created in the same namespace.

.spec.workspaces[].value.configMap

Description

ConfigMap represents a configMap that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode is optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths

Property	Type	Description
		must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional specify whether the ConfigMap or its keys must be defined

.spec.workspaces[].value.configMap.items

Description

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].value.configMap.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].value.csi

Description

CSI (Container Storage Interface) represents ephemeral storage that is handled by certain external CSI drivers.

Type

`object`

Required

`driver`

Property	Type	Description
`driver`	`string`	driver is the name of the CSI driver that handles this volume. Consult with your admin for the correct name as registered in the cluster.
`fsType`	`string`	fsType to mount. Ex. "ext4", "xfs", "ntfs". If not provided, the empty value is passed to the associated CSI driver which will determine the default filesystem to apply.
`nodePublishSecretRef`	`object`	nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secret references are passed.
`readOnly`	`boolean`	readOnly specifies a read-only configuration for the volume. Defaults to false (read/write).
`volumeAttributes`	`object`	volumeAttributes stores driver-specific properties that are passed to the CSI driver. Consult your driver's documentation for supported values.

.spec.workspaces[].value.csi.nodePublishSecretRef

Description

nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secret references are passed.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.workspaces[].value.csi.volumeAttributes

Description

volumeAttributes stores driver-specific properties that are passed to the CSI driver. Consult your driver's documentation for supported values.

Type

`object`

.spec.workspaces[].value.emptyDir

Description

EmptyDir represents a temporary directory that shares a Task's lifetime. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir Either this OR

PersistentVolumeClaim can be used.

Type

`object`

Property	Type	Description
`medium`	`string`	medium represents what type of storage medium should back this directory. The default is "" which means to use the node's default medium. Must be an empty string (default) or Memory. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir ↗
`sizeLimit`		sizeLimit is the total amount of local storage required for this EmptyDir volume. The size limit is also applicable for memory medium. The maximum usage on memory medium EmptyDir would be the minimum value between the SizeLimit specified here and the sum of memory limits of all containers in a pod. The default is nil which means that the limit is undefined. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir ↗

.spec.workspaces[].value.persistentVolumeClaim

Description

PersistentVolumeClaimVolumeSource represents a reference to a PersistentVolumeClaim in the same namespace. Either this OR EmptyDir can be used.

Type

`object`

Required

`claimName`

Property	Type	Description
`claimName`	`string`	claimName is the name of a PersistentVolumeClaim in the same namespace as the pod using this volume. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims ↗
`readOnly`	`boolean`	readOnly Will force the ReadOnly setting in VolumeMounts. Default false.

.spec.workspaces[].value.projected

Description

Projected represents a projected volume that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode are the mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

Property	Type	Description
`sources`	`array`	sources is the list of volume projections. Each entry in this list handles one source.

.spec.workspaces[].value.projected.sources

Description

sources is the list of volume projections. Each entry in this list handles one source.

Type

`array`

.spec.workspaces[].value.projected.sources[]

Description

Projection that may be projected along with other supported volume types. Exactly one of these fields must be set.

Type

`object`

Property	Type	Description
`clusterTrustBundle`	`object`	ClusterTrustBundle allows a pod to access the `.spec.trustBundle` field of ClusterTrustBundle objects in an auto-updating file. Alpha, gated by the ClusterTrustBundleProjection feature gate. ClusterTrustBundle objects can either be selected by name, or by the combination of signer name and a label selector.

Property	Type	Description
		Kubelet performs aggressive normalization of the PEM contents written into the pod filesystem. Esoteric PEM features such as inter-block comments and block headers are stripped. Certificates are deduplicated. The ordering of certificates within the file is arbitrary, and Kubelet may change the order over time.
`configMap`	`object`	configMap information about the configMap data to project
`downwardAPI`	`object`	downwardAPI information about the downwardAPI data to project
`secret`	`object`	secret information about the secret data to project
`serviceAccountToken`	`object`	serviceAccountToken is information about the serviceAccountToken data to project

.spec.workspaces[].value.projected.sources[].clusterTrustBundle

Description

ClusterTrustBundle allows a pod to access the `.spec.trustBundle` field of ClusterTrustBundle objects in an auto-updating file. Alpha, gated by the ClusterTrustBundleProjection feature gate. ClusterTrustBundle objects can either be selected by name, or by the combination of signer name and a label selector. Kubelet performs aggressive normalization of the PEM contents written into the pod filesystem.

Esoteric PEM features such as inter-block comments and block headers are stripped. Certificates are deduplicated. The ordering of certificates within the file is arbitrary, and Kubelet may change the order over time.

Type

`object`

Required

`path`

Property	Type	Description
`labelSelector`	`object`	Select all ClusterTrustBundles that match this label selector. Only has effect if signerName is set. Mutually-exclusive with name. If unset, interpreted as "match nothing". If set but empty, interpreted as "match everything".
`name`	`string`	Select a single ClusterTrustBundle by object name. Mutually-exclusive with signerName and labelSelector.
`optional`	`boolean`	If true, don't block pod startup if the referenced ClusterTrustBundle(s) aren't available. If using name, then the named ClusterTrustBundle is allowed not to exist. If using signerName, then the combination of signerName and labelSelector is allowed to match zero ClusterTrustBundles.
`path`	`string`	Relative path from the volume root to write the bundle.
`signerName`	`string`	Select all ClusterTrustBundles that match this signer name. Mutually-exclusive with name. The contents of all

Property	Type	Description
		selected ClusterTrustBundles will be unified and deduplicated.

.spec.workspaces[].value.projected.sources[].clusterTrustBundle.labelSelector

Description

Select all ClusterTrustBundles that match this label selector. Only has effect if signerName is set. Mutually-exclusive with name. If unset, interpreted as "match nothing". If set but empty, interpreted as "match everything".

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

.spec.workspaces[].value.projected.sources[].clusterTrustBundle.labelSelector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.workspaces[].value.projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.workspaces[].value.projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.workspaces[].value.projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[].values[]

Type

`string`

.spec.workspaces[].value.projected.sources[].clusterTrustBundle.labelSelector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.workspaces[].value.projected.sources[].configMap

Description

configMap information about the configMap data to project

Type

`object`

Property	Type	Description
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional specify whether the ConfigMap or its keys must be defined

.spec.workspaces[].value.projected.sources[].configMap.items

Description

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not

be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].value.projected.sources[].configMap.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element

Property	Type	Description
		'..'. May not start with the string '..'.

.spec.workspaces[].value.projected.sources[].downwardAPI

Description

downwardAPI information about the downwardAPI data to project

Type

`object`

Property	Type	Description
`items`	`array`	Items is a list of DownwardAPIVolume file

.spec.workspaces[].value.projected.sources[].downwardAPI.items

Description

Items is a list of DownwardAPIVolume file

Type

`array`

.spec.workspaces[].value.projected.sources[].downwardAPI.items[]

Description

DownwardAPIVolumeFile represents information to create the file containing the pod field

Type

`object`

Required

`path`

Property	Type	Description
`fieldRef`	`object`	Required: Selects a field of the pod: only annotations, labels, name, namespace and uid are supported.
`mode`	`integer`	Optional: mode bits used to set permissions on this file, must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	Required: Path is the relative path name of the file to be created. Must not be absolute or contain the '..' path. Must be utf-8 encoded. The first item of the relative path must not start with '..'
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, requests.cpu and requests.memory) are currently supported.

.spec.workspaces[].value.projected.sources[].downwardAPI.items[].fieldRef

Description

Required: Selects a field of the pod: only annotations, labels, name, namespace and uid are supported.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.workspaces[].value.projected.sources[].downwardAPI.items[].resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, requests.cpu and requests.memory) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.workspaces[].value.projected.sources[].secret

Description

secret information about the secret data to project

Type

`object`

Property	Type	Description
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional field specify whether the Secret or its key must be defined

.spec.workspaces[].value.projected.sources[].secret.items

Description

items if unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].value.projected.sources[].secret.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].value.projected.sources[].serviceAccountToken

Description

serviceAccountToken is information about the serviceAccountToken data to project

Type

`object`

Required

`path`

Property	Type	Description
`audience`	`string`	audience is the intended audience of the token. A recipient of a token must identify itself with an identifier specified in the audience of the token, and otherwise should reject the token. The audience defaults to the identifier of the apiserver.
`expirationSeconds`	`integer`	expirationSeconds is the requested duration of validity of the service account token. As the token approaches expiration, the kubelet volume plugin will proactively rotate the service account token. The kubelet will start trying to rotate the token if the token is older than 80 percent of its time to live or if the token is older than 24 hours.Defaults to 1 hour and must be at least 10 minutes.
`path`	`string`	path is the path relative to the mount point of the file to project the token into.

.spec.workspaces[].value.secret

Description

Secret represents a secret that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode is Optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`items`	`array`	items If unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`optional`	`boolean`	optional field specify whether the Secret or its keys must be defined
`secretName`	`string`	secretName is the name of the secret in the pod's namespace to use. More info: https://kubernetes.io/docs/concepts/storage/volumes#secret ↗

.spec.workspaces[].value.secret.items

Description

items If unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].value.secret.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

Property	Type	Description
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].value.volumeClaimTemplate

Description

VolumeClaimTemplate is a template for a claim that will be created in the same namespace.

Type

`object`

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: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

Property	Type	Description
`metadata`	`ObjectMeta`	Standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
`spec`	`object`	spec defines the desired characteristics of a volume requested by a pod author. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims
`status`	`object`	status represents the current information/status of a persistent volume claim. Read-only. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims

.spec.workspaces[].value.volumeClaimTemplate.spec

Description

spec defines the desired characteristics of a volume requested by a pod author. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims

Type

`object`

Property	Type	Description
`accessModes`	`array`	accessModes contains the desired access mod volume should have. More info: https://kubernetes.io/docs/concepts/storage/per volumes#access-modes-1 ↗
`dataSource`	`object`	dataSource field can be used to specify either: • An existing VolumeSnapshot object (snapshot.storage.k8s.io/VolumeSnapshot) • An existing PVC (PersistentVolumeClaim) If provisioner or an external controller can sup the specified data source, it will create a nev volume based on the contents of the specifie source. When the AnyVolumeDataSource fe gate is enabled, dataSource contents will be to dataSourceRef, and dataSourceRef conte be copied to dataSource when dataSourceRef.namespace is not specified. namespace is specified, then dataSourceRe not be copied to dataSource.
`dataSourceRef`	`object`	dataSourceRef specifies the object from which populate the volume with data, if a non-empty v is desired. This may be any object from a non-e API group (non core object) or a PersistentVolumeClaim object. When this field i specified, volume binding will only succeed if th of the specified object matches some installed v populator or dynamic provisioner. This field will the functionality of the dataSource field and as s both fields are non-empty, they must have the s

Property	Type	Description
		value. For backwards compatibility, when name isn't specified in dataSourceRef, both fields (dataSource and dataSourceRef) will be set to t same value automatically if one of them is empt the other is non-empty. When namespace is sp in dataSourceRef, dataSource isn't set to the sa value and must be empty. There are three impo differences between dataSource and dataSourc • While dataSource only allows two specific ty objects, dataSourceRef allows any non-core as well as PersistentVolumeClaim objects. • While dataSource ignores disallowed values (dropping them), dataSourceRef preserves a values, and generates an error if a disallowe value is specified. • While dataSource only allows local objects, dataSourceRef allows objects in any names (Beta) Using this field requires the AnyVolumeDataSource feature gate to be en (Alpha) Using the namespace field of dataSourceRef requires the CrossNamespaceVolumeDataSource featur to be enabled.

Property	Type	Description
`resources`	`object`	resources represents the minimum resources th volume should have. If RecoverVolumeExpansionFailure feature is ena users are allowed to specify resource requireme that are lower than previous value but must still higher than capacity recorded in the status field claim. More info: https://kubernetes.io/docs/concepts/storage/per volumes#resources ↗
`selector`	`object`	selector is a label query over volumes to consid binding.
`storageClassName`	`string`	storageClassName is the name of the StorageC required by the claim. More info: https://kubernetes.io/docs/concepts/storage/per volumes#class-1 ↗
`volumeAttributesClassName`	`string`	volumeAttributesClassName may be used to se VolumeAttributesClass used by this claim. If spe the CSI driver will create or update the volume v attributes defined in the corresponding VolumeAttributesClass. This has a different pur than storageClassName, it can be changed afte claim is created. An empty string value means t VolumeAttributesClass will be applied to the cla it's not allowed to reset this field to empty string is set. If unspecified and the PersistentVolumeC unbound, the default VolumeAttributesClass wil by the persistentvolume controller if it exists. If t

Property	Type	Description
		resource referred to by volumeAttributesClass d not exist, this PersistentVolumeClaim will be se Pending state, as reflected by the modifyVolum field, until such as a resource exists. More info: https://kubernetes.io/docs/concepts/storage/vol attributes-classes/ (Beta) Using this field requ the VolumeAttributesClass feature gate to be er (off by default).
`volumeMode`	`string`	volumeMode defines what type of volume is rec by the claim. Value of Filesystem is implied whe included in claim spec.
`volumeName`	`string`	volumeName is the binding reference to the PersistentVolume backing this claim.

.spec.workspaces[].value.volumeClaimTemplate.spec.accessModes

Description

accessModes contains the desired access modes the volume should have. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#access-modes-1

Type

`array`

.spec.workspaces[].value.volumeClaimTemplate.spec.accessModes[]

Type

`string`

.spec.workspaces[].value.volumeClaimTemplate.spec.dataSource

Description

dataSource field can be used to specify either: * An existing VolumeSnapshot object (snapshot.storage.k8s.io/VolumeSnapshot) * An existing PVC (PersistentVolumeClaim) If the provisioner or an external controller can support the specified data source, it will create a new volume based on the contents of the specified data source. When the AnyVolumeDataSource feature gate is enabled, dataSource contents will be copied to dataSourceRef, and dataSourceRef contents will be copied to dataSource when dataSourceRef.namespace is not specified. If the namespace is specified, then dataSourceRef will not be copied to dataSource.

Type

`object`

Required

`kind` `name`

Property	Type	Description
`apiGroup`	`string`	APIGroup is the group for the resource being referenced. If APIGroup is not specified, the specified Kind must be in the core API group. For any other third-party types, APIGroup is required.
`kind`	`string`	Kind is the type of resource being referenced
`name`	`string`	Name is the name of resource being referenced

.spec.workspaces[].value.volumeClaimTemplate.spec.dataSourceRef

Description

dataSourceRef specifies the object from which to populate the volume with data, if a non-empty volume is desired. This may be any object from a non-empty API group (non core object) or a PersistentVolumeClaim object. When this field is specified, volume binding will only succeed if the type of the specified object matches some installed volume populator or dynamic provisioner. This field will replace the functionality of the dataSource field and as such if both fields are non-empty, they must have the same value. For backwards compatibility, when namespace isn't specified in dataSourceRef, both fields (dataSource and dataSourceRef) will be set to the same value automatically if one of them is empty and the other is non-empty. When namespace is specified in dataSourceRef, dataSource isn't set to the same value and must be empty. There are three important differences between dataSource and dataSourceRef: * While dataSource only allows two specific types of objects, dataSourceRef allows any non-core object, as well as PersistentVolumeClaim objects. * While dataSource ignores disallowed values (dropping them), dataSourceRef preserves all values, and generates an error if a disallowed value is specified. * While dataSource only allows local objects, dataSourceRef allows objects in any namespaces. (Beta) Using this field requires the AnyVolumeDataSource feature gate to be enabled. (Alpha) Using the namespace field of dataSourceRef requires the CrossNamespaceVolumeDataSource feature gate to be enabled.

Type

`object`

Required

`kind` `name`

Property	Type	Description
`apiGroup`	`string`	APIGroup is the group for the resource being referenced. If APIGroup is not specified, the specified Kind must be in the core API group. For any other third-party types, APIGroup is required.

Property	Type	Description
`kind`	`string`	Kind is the type of resource being referenced
`name`	`string`	Name is the name of resource being referenced
`namespace`	`string`	Namespace is the namespace of resource being referenced Note that when a namespace is specified, a gateway.networking.k8s.io/ReferenceGrant object is required in the referent namespace to allow that namespace's owner to accept the reference. See the ReferenceGrant documentation for details. (Alpha) This field requires the CrossNamespaceVolumeDataSource feature gate to be enabled.

.spec.workspaces[].value.volumeClaimTemplate.spec.resources

Description

resources represents the minimum resources the volume should have. If RecoverVolumeExpansionFailure feature is enabled users are allowed to specify resource requirements that are lower than previous value but must still be higher than capacity recorded in the status field of the claim. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#resources

Type

`object`

Property	Type	Description
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.spec.workspaces[].value.volumeClaimTemplate.spec.resources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.workspaces[].value.volumeClaimTemplate.spec.resources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an

implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.workspaces[].value.volumeClaimTemplate.spec.selector

Description

selector is a label query over volumes to consider for binding.

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

.spec.workspaces[].value.volumeClaimTemplate.spec.selector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.workspaces[].value.volumeClaimTemplate.spec.selector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.workspaces[].value.volumeClaimTemplate.spec.selector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.workspaces[].value.volumeClaimTemplate.spec.selector.matchExpressions[].values[]

Type

`string`

.spec.workspaces[].value.volumeClaimTemplate.spec.selector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.workspaces[].value.volumeClaimTemplate.status

Description

status represents the current information/status of a persistent volume claim. Read-only. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims

Type

`object`

Property	Type	Description
`accessModes`	`array`	accessModes contains the actual acces volume backing the PVC has. More info https://kubernetes.io/docs/concepts/sto volumes#access-modes-1 ↗
`allocatedResourceStatuses`	`object`	allocatedResourceStatuses stores statu being resized for the given PVC. Key n standard Kubernetes label syntax. Valid either: * Un-prefixed keys: - storage - th the volume. * Custom resources must u implementation-defined prefixed names "example.com/my-custom-resource" Ap values - keys that are unprefixed or hav kubernetes.io prefix are considered res hence may not be used. ClaimResourceStatus can be in any of states: - ControllerResizeInProgress: S resize controller starts resizing the volu plane. - ControllerResizeFailed: State s has failed in resize controller with a terr NodeResizePending: State set when re has finished resizing the volume but fur volume is needed on the node. - NodeResizeInProgress: State set wher resizing the volume. - NodeResizeFaile when resizing has failed in kubelet with error. Transient errors don't set NodeRe example: if expanding a PVC for more field can be one of the following states: pvc.status.allocatedResourceStatus['st "ControllerResizeInProgress" - pvc.status.allocatedResourceStatus['st

Property	Type	Description
		"ControllerResizeFailed" - pvc.status.allocatedResourceStatus['st "NodeResizePending" - pvc.status.allocatedResourceStatus['st "NodeResizeInProgress" - pvc.status.allocatedResourceStatus['st "NodeResizeFailed" When this field is r means that no resize operation is in pro given PVC. A controller that receives PVC update w unknown resourceName or ClaimReso should ignore the update for the purpos designed. For example - a controller tha responsible for resizing capacity of the ignore PVC updates that change other associated with PVC. This is an alpha field and requires enab RecoverVolumeExpansionFailure featu
`allocatedResources`	`object`	allocatedResources tracks the resource a PVC including its capacity. Key name standard Kubernetes label syntax. Valid either: * Un-prefixed keys: - storage - th the volume. * Custom resources must u implementation-defined prefixed names "example.com/my-custom-resource" Ap values - keys that are unprefixed or hav kubernetes.io prefix are considered res hence may not be used. Capacity reported here may be larger th capacity when a volume expansion ope requested. For storage quota, the large

Property	Type	Description
		allocatedResources and PVC.spec.res If allocatedResources is not set, PVC.s alone is used for quota calculation. If a expansion capacity request is lowered, allocatedResources is only lowered if t expansion operations in progress and i volume capacity is equal or lower than capacity. A controller that receives PVC update v unknown resourceName should ignore the purpose it was designed. For exam controller that only is responsible for re of the volume, should ignore PVC upda change other valid resources associate This is an alpha field and requires enab RecoverVolumeExpansionFailure featu
`capacity`	`object`	capacity represents the actual resource underlying volume.
`conditions`	`array`	conditions is the current Condition of pe volume claim. If underlying persistent v resized then the Condition will be set to

Property	Type	Description
`currentVolumeAttributesClassName`	`string`	currentVolumeAttributesClassName is t name of the VolumeAttributesClass the When unset, there is no VolumeAttribut to this PersistentVolumeClaim This is a requires enabling VolumeAttributesClas by default).
`modifyVolumeStatus`	`object`	ModifyVolumeStatus represents the sta ControllerModifyVolume operation. Wh there is no ModifyVolume operation bei This is a beta field and requires enablir VolumeAttributesClass feature (off by d
`phase`	`string`	phase represents the current phase of PersistentVolumeClaim.

.spec.workspaces[].value.volumeClaimTemplate.status.accessModes

Description

accessModes contains the actual access modes the volume backing the PVC has. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#access-modes-1

Type

`array`

.spec.workspaces[].value.volumeClaimTemplate.status.accessModes[]

Type

`string`

.spec.workspaces[].value.volumeClaimTemplate.status.allocatedResourceStatuses

Description

allocatedResourceStatuses stores status of resource being resized for the given PVC. Key names follow standard Kubernetes label syntax. Valid values are either: * Un-prefixed keys: - storage - the capacity of the volume. * Custom resources must use implementation-defined prefixed names such as "example.com/my-custom-resource" Apart from above values - keys that are unprefixed or have kubernetes.io prefix are considered reserved and hence may not be used. ClaimResourceStatus can be in any of following states: - ControllerResizeInProgress: State set when resize controller starts resizing the volume in control-plane. - ControllerResizeFailed: State set when resize has failed in resize controller with a terminal error. - NodeResizePending: State set when resize controller has finished resizing the volume but further resizing of volume is needed on the node. - NodeResizeInProgress: State set when kubelet starts resizing the volume. - NodeResizeFailed: State set when resizing has failed in kubelet with a terminal error. Transient errors don't set NodeResizeFailed. For example: if expanding a PVC for more capacity - this field can be one of the following states: - pvc.status.allocatedResourceStatus['storage'] = "ControllerResizeInProgress" - pvc.status.allocatedResourceStatus['storage'] = "ControllerResizeFailed" - pvc.status.allocatedResourceStatus['storage'] = "NodeResizePending" - pvc.status.allocatedResourceStatus['storage'] = "NodeResizeInProgress" - pvc.status.allocatedResourceStatus['storage'] = "NodeResizeFailed" When this field is not set, it means that no resize operation is in progress for the given PVC. A controller that receives PVC update with previously unknown resourceName or ClaimResourceStatus should ignore the update for the purpose it was designed. For example - a controller that only is responsible for resizing capacity of the volume, should ignore PVC updates that change other valid resources associated with PVC. This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

Type

`object`

.spec.workspaces[].value.volumeClaimTemplate.status.allocatedResources

Description

allocatedResources tracks the resources allocated to a PVC including its capacity. Key names follow standard Kubernetes label syntax. Valid values are either: * Un-prefixed keys: - storage - the capacity of the volume. * Custom resources must use implementation-defined prefixed names such as "example.com/my-custom-resource" Apart from above values - keys that are unprefixed or have kubernetes.io prefix are considered reserved and hence may not be used. Capacity reported here may be larger than the actual capacity when a volume expansion operation is requested. For storage quota, the larger value from allocatedResources and PVC.spec.resources is used. If allocatedResources is not set, PVC.spec.resources alone is used for quota calculation. If a volume expansion capacity request is lowered, allocatedResources is only lowered if there are no expansion operations in progress and if the actual volume capacity is equal or lower than the requested capacity. A controller that receives PVC update with previously unknown resourceName should ignore the update for the purpose it was designed. For example - a controller that only is responsible for resizing capacity of the volume, should ignore PVC updates that change other valid resources associated with PVC. This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

Type

`object`

.spec.workspaces[].value.volumeClaimTemplate.status.capacity

Description

capacity represents the actual resources of the underlying volume.

Type

`object`

.spec.workspaces[].value.volumeClaimTemplate.status.conditions

Description

conditions is the current Condition of persistent volume claim. If underlying persistent volume is being resized then the Condition will be set to 'Resizing'.

Type

`array`

.spec.workspaces[].value.volumeClaimTemplate.status.conditions[]

Description

PersistentVolumeClaimCondition contains details about state of pvc

Type

`object`

Required

`status` `type`

Property	Type	Description
`lastProbeTime`	`string`	lastProbeTime is the time we probed the condition.
`lastTransitionTime`	`string`	lastTransitionTime is the time the condition transitioned
`message`	`string`	message is the human-readable message indicating de
`reason`	`string`	reason is a unique, this should be a short, machine und the reason for condition's last transition. If it reports "Re underlying persistent volume is being resized.

Property	Type	Description
`status`	`string`	Status is the status of the condition. Can be True, False https://kubernetes.io/docs/reference/kubernetes-api/con resources/persistent-volume-claim-v1/#:~:text=state%20of%20pvc-,conditions.status,-(strin
`type`	`string`	Type is the type of the condition. More info: https://kubernetes.io/docs/reference/kubernetes-api/con resources/persistent-volume-claim-v1/#:~:text=set%20to%20%27ResizeStarted%27.-,Pers contains%20details%20about ↗

.spec.workspaces[].value.volumeClaimTemplate.status.modifyVolumeStatus

Description

ModifyVolumeStatus represents the status object of ControllerModifyVolume operation. When this is unset, there is no ModifyVolume operation being attempted. This is a beta field and requires enabling VolumeAttributesClass feature (off by default).

Type

`object`

Required

`status`

Property	Type	Description
`status`	`string`	status is the status of the ControllerModifyVolume operation. It can be in any of following states:

Property	Type	Description
		• Pending Pending indicates that the PersistentVolumeClaim cannot be modified due to unmet requirements, such as the specified VolumeAttributesClass not existing. • InProgress InProgress indicates that the volume is being modified. • Infeasible Infeasible indicates that the request has been rejected as invalid by the CSI driver. To resolve the error, a valid VolumeAttributesClass needs to be specified. Note: New statuses can be added in the future. Consumers should check for unknown statuses and fail appropriately.
`targetVolumeAttributesClassName`	`string`	targetVolumeAttributesClassName is the name of the VolumeAttributesClass the PVC currently being reconciled

.spec.workspaces[].workspaceMapping

Description

WorkspaceMapping defines the workspace mapping configuration for this workspace.

Type

`object`

Property	Type	Description
`name`	`string`	Name is the default parameter name when this workspace is mapped during PipelineIntegration construction.

API Endpoints

The following API endpoints are available:

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

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

HTTP method

`DELETE`

Description

delete collection of ResourceInterface

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ResourceInterface

HTTP responses

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

HTTP method

`POST`

Description

create a new ResourceInterface

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

Parameter	Type	Description
		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`	`ResourceInterface` schema	`application/json` formatted

HTTP responses

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

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

HTTP method

`DELETE`

Description

delete the specified ResourceInterface

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 ResourceInterface

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ResourceInterface

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	`ResourceInterface` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ResourceInterface

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`	`ResourceInterface` schema	`application/json` formatted

HTTP responses

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