Networking APIs

HTTPRoute [httproutes.gateway.networking.k8s.io/v1]

Description

HTTPRoute provides a way to route HTTP requests. This includes the capability to match requests by hostname, path, header, or query param. Filters can be used to specify additional processing steps. Backends specify where matching requests should be routed.

Type

`object`

Required

`spec`

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 HTTPRoute.
`status`	`object`	Status defines the current state of HTTPRoute.

.spec

Description

Spec defines the desired state of HTTPRoute.

Type

`object`

Property	Type	Description
`hostnames`	`array`	Hostnames defines a set of hostnames that should match against the HTTP Host header to select a HTTPRoute used to process the request. Implementations MUST ignore any port value specified in the HTTP Host header while performing a

Property	Type	Description
		match and (absent of any applicable header modification configuration) MUST forward this header unmodified to the backend. Valid values for Hostnames are determined by RFC 1123 definition of a hostname with 2 notable exceptions: 1. IPs are not allowed. 2. A hostname may be prefixed with a wildcard label (`*.`). The wildcard label must appear by itself as the first label. If a hostname is specified by both the Listener and HTTPRoute, there must be at least one intersecting hostname for the HTTPRoute to be attached to the Listener. For example: • A Listener with `test.example.com` as the hostname matches HTTPRoutes that have either not specified any hostnames, or have specified at least one of `test.example.com` or `*.example.com`. • A Listener with `*.example.com` as the hostname matches HTTPRoutes that have either not specified any hostnames or have specified at least one hostname that matches the Listener hostname. For example, `*.example.com`, `test.example.com`, and `foo.test.example.com` would all match. On the other hand, `example.com` and `test.example.net` would not match. Hostnames that are prefixed with a wildcard label (`*.`) are interpreted as a suffix match. That means that a match for `*.example.com` would match both `test.example.com`, and `foo.test.example.com`, but not `example.com`. If both the Listener and HTTPRoute have specified hostnames, any HTTPRoute hostnames that do not match the Listener hostname MUST be ignored. For example, if a Listener

Property	Type	Description
		specified `*.example.com`, and the HTTPRoute specified `test.example.com` and `test.example.net`, `test.example.net` must not be considered for a match. If both the Listener and HTTPRoute have specified hostnames, and none match with the criteria above, then the HTTPRoute is not accepted. The implementation must raise an 'Accepted' Condition with a status of `False` in the corresponding RouteParentStatus. In the event that multiple HTTPRoutes specify intersecting hostnames (e.g. overlapping wildcard matching and exact matching hostnames), precedence must be given to rules from the HTTPRoute with the largest number of: • Characters in a matching non-wildcard hostname. • Characters in a matching hostname. If ties exist across multiple Routes, the matching precedence rules for HTTPRouteMatches takes over. Support: Core
`parentRefs`	`array`	ParentRefs references the resources (usually Gateways) that a Route wants to be attached to. Note that the referenced parent resource needs to allow this for the attachment to be complete. For Gateways, that means the Gateway needs to allow attachment from Routes of this kind and namespace. For Services, that means the Service must either be in the same namespace for a "producer" route, or the mesh implementation must support and allow "consumer" routes for the referenced Service. ReferenceGrant is not applicable for governing ParentRefs to Services - it is not possible to create a "producer" route for a Service in a different namespace from the Route.

Property	Type	Description
		There are two kinds of parent resources with "Core" support: • Gateway (Gateway conformance profile) • Service (Mesh conformance profile, ClusterIP Services only) This API may be extended in the future to support additional kinds of parent resources. ParentRefs must be *distinct*. This means either that: • They select different objects. If this is the case, then parentRef entries are distinct. In terms of fields, this means that the multi-part key defined by `group`, `kind`, `namespace`, and `name` must be unique across all parentRef entries in the Route. • They do not select different objects, but for each optional field used, each ParentRef that selects the same object must set the same set of optional fields to different values. If one ParentRef sets a combination of optional fields, all must set the same combination. Some examples: • If one ParentRef sets `sectionName`, all ParentRefs referencing the same object must also set `sectionName`. • If one ParentRef sets `port`, all ParentRefs referencing the same object must also set `port`. • If one ParentRef sets `sectionName` and `port`, all ParentRefs referencing the same object must also set `sectionName` and `port`. It is possible to separately reference multiple distinct objects that may be collapsed by an implementation. For example, some implementations may choose to merge compatible

Property	Type	Description
		Gateway Listeners together. If that is the case, the list of routes attached to those resources should also be merged. Note that for ParentRefs that cross namespace boundaries, there are specific rules. Cross-namespace references are only valid if they are explicitly allowed by something in the namespace they are referring to. For example, Gateway has the AllowedRoutes field, and ReferenceGrant provides a generic way to enable other kinds of cross-namespace reference.
`rules`	`array`	Rules are a list of HTTP matchers, filters and actions.

.spec.hostnames

Description

Hostnames defines a set of hostnames that should match against the HTTP Host header to select a HTTPRoute used to process the request. Implementations MUST ignore any port value specified in the HTTP Host header while performing a match and (absent of any applicable header modification configuration) MUST forward this header unmodified to the backend. Valid values for Hostnames are determined by RFC 1123 definition of a hostname with 2 notable exceptions: 1. IPs are not allowed. 2. A hostname may be prefixed with a wildcard label (`*.`). The wildcard label must appear by itself as the first label. If a hostname is specified by both the Listener and HTTPRoute, there must be at least one intersecting hostname for the HTTPRoute to be attached to the Listener. For example: * A Listener with `test.example.com` as the hostname matches HTTPRoutes that have either not specified any hostnames, or have specified at least one of `test.example.com` or `*.example.com`. * A Listener with `*.example.com` as the hostname matches HTTPRoutes that have either not specified any hostnames or have specified at least one hostname that matches the Listener hostname. For example, `*.example.com`, `test.example.com`, and `foo.test.example.com` would all match. On the other hand, `example.com` and `test.example.net` would not match. Hostnames that are prefixed with a wildcard label (`*.`) are interpreted as a suffix match. That means that a match for `*.example.com` would match both `test.example.com`, and `foo.test.example.com`, but not `example.com`. If both the Listener and HTTPRoute have specified hostnames, any HTTPRoute hostnames that do not match the Listener hostname MUST be ignored. For example, if a Listener specified `*.example.com`, and the HTTPRoute specified `test.example.com` and `test.example.net`, `test.example.net` must not be considered for a match. If both the Listener and HTTPRoute have specified hostnames, and none match with the criteria above, then the HTTPRoute is not accepted. The implementation must raise an 'Accepted' Condition with a status of `False` in the corresponding RouteParentStatus. In the event that multiple HTTPRoutes specify intersecting hostnames (e.g. overlapping wildcard matching and exact matching hostnames), precedence must be given to rules from the HTTPRoute with the largest number of: * Characters in a matching non-wildcard hostname. * Characters in a matching hostname. If ties exist across multiple Routes, the matching precedence rules for HTTPRouteMatches takes over. Support: Core

Type

`array`

.spec.hostnames[]

Description

Hostname is the fully qualified domain name of a network host. This matches the RFC 1123 definition of a hostname with 2 notable exceptions: 1. IPs are not allowed. 2. A hostname may be prefixed with a wildcard label (`*.`). The wildcard label must appear by itself as the first label. Hostname can be "precise" which is a domain name without the terminating dot of a network host (e.g. "foo.example.com") or "wildcard", which is a domain name prefixed with a single wildcard label (e.g. `*.example.com`). Note that as per RFC1035 and RFC1123, a *label* must consist of lower case alphanumeric characters or '-', and must start and end with an alphanumeric character. No other punctuation is allowed.

Type

`string`

.spec.parentRefs

Description

ParentRefs references the resources (usually Gateways) that a Route wants to be attached to. Note that the referenced parent resource needs to allow this for the attachment to be complete. For Gateways, that means the Gateway needs to allow attachment from Routes of this kind and namespace. For Services, that means the Service must either be in the same namespace for a "producer" route, or the mesh implementation must support and allow "consumer" routes for the referenced Service. ReferenceGrant is not applicable for governing ParentRefs to Services - it is not possible to create a "producer" route for a Service in a different namespace from the Route. There are two kinds of parent resources with "Core" support: * Gateway (Gateway conformance profile) * Service (Mesh conformance profile, ClusterIP Services only) This API may be extended in the future to support additional kinds of parent resources. ParentRefs must be _distinct_. This means either that: * They select different objects. If this is the case, then parentRef entries are distinct. In terms of fields, this means that the multi-part key defined by `group`, `kind`, `namespace`, and `name` must be unique across all parentRef entries in the Route. * They do not select different objects, but for each optional field used, each ParentRef that selects the same object must set the same set of optional fields to different values. If one ParentRef sets a combination of optional fields, all must set the same combination. Some examples: * If one ParentRef sets `sectionName`, all ParentRefs referencing the same object must also set `sectionName`. * If one ParentRef sets `port`, all ParentRefs referencing the same object must also set `port`. * If one ParentRef sets `sectionName` and `port`, all ParentRefs referencing the same object must also set `sectionName` and `port`. It is possible to separately reference multiple distinct objects that may be collapsed by an implementation. For example, some implementations may choose to merge compatible Gateway Listeners together. If that is the case, the list of routes attached to those resources should also be merged. Note that for ParentRefs that cross namespace boundaries, there are specific rules. Cross-namespace references are only valid if they are explicitly allowed by something in the namespace they are referring to. For example, Gateway has the AllowedRoutes field, and ReferenceGrant provides a generic way to enable other kinds of cross-namespace reference.

Type

`array`

.spec.parentRefs[]

Description

ParentReference identifies an API object (usually a Gateway) that can be considered a parent of this resource (usually a route). There are two kinds of parent resources with "Core" support: * Gateway (Gateway conformance profile) * Service (Mesh conformance profile, ClusterIP Services only) This API may be extended in the future to support additional kinds of parent resources. The API object must be valid in the cluster; the Group and Kind must be registered in the cluster for this reference to be valid.

Type

`object`

Required

`name`

Property	Type	Description
`group`	`string`	Group is the group of the referent. When unspecified, "gateway.networking.k8s.io" is inferred. To set the core API group (such as for a "Service" kind referent), Group must be explicitly set to "" (empty string). Support: Core
`kind`	`string`	Kind is kind of the referent. There are two kinds of parent resources with "Core" support: • Gateway (Gateway conformance profile) • Service (Mesh conformance profile, ClusterIP Services only) Support for other resources is Implementation-Specific.
`name`	`string`	Name is the name of the referent. Support: Core

Property	Type	Description
`namespace`	`string`	Namespace is the namespace of the referent. When unspecified, this refers to the local namespace of the Route. Note that there are specific rules for ParentRefs which cross namespace boundaries. Cross-namespace references are only valid if they are explicitly allowed by something in the namespace they are referring to. For example: Gateway has the AllowedRoutes field, and ReferenceGrant provides a generic way to enable any other kind of cross-namespace reference. Support: Core
`port`	`integer`	Port is the network port this Route targets. It can be interpreted differently based on the type of parent resource. When the parent resource is a Gateway, this targets all listeners listening on the specified port that also support this kind of Route(and select this Route). It's not recommended to set `Port` unless the networking behaviors specified in a Route must apply to a specific port as opposed to a listener(s) whose port(s) may be changed. When both Port and SectionName are specified, the name and port of the selected listener must match both specified values. Implementations MAY choose to support other parent resources. Implementations supporting other types of parent resources MUST clearly document how/if Port is interpreted. For the purpose of status, an attachment is considered successful as long as the parent resource accepts it partially. For example, Gateway listeners can restrict which Routes can attach to them by Route kind, namespace, or hostname. If 1 of 2 Gateway listeners accept attachment

Property	Type	Description
		from the referencing Route, the Route MUST be considered successfully attached. If no Gateway listeners accept attachment from this Route, the Route MUST be considered detached from the Gateway. Support: Extended
`sectionName`	`string`	SectionName is the name of a section within the target resource. In the following resources, SectionName is interpreted as the following: • Gateway: Listener name. When both Port (experimental) and SectionName are specified, the name and port of the selected listener must match both specified values. • Service: Port name. When both Port (experimental) and SectionName are specified, the name and port of the selected listener must match both specified values. Implementations MAY choose to support attaching Routes to other resources. If that is the case, they MUST clearly document how SectionName is interpreted. When unspecified (empty string), this will reference the entire resource. For the purpose of status, an attachment is considered successful if at least one section in the parent resource accepts it. For example, Gateway listeners can restrict which Routes can attach to them by Route kind, namespace, or hostname. If 1 of 2 Gateway listeners accept attachment from the referencing Route, the Route MUST be considered successfully attached. If no Gateway listeners accept attachment from this Route, the Route MUST be considered detached from the Gateway. Support: Core

.spec.rules

Description

Rules are a list of HTTP matchers, filters and actions.

Type

`array`

.spec.rules[]

Description

HTTPRouteRule defines semantics for matching an HTTP request based on conditions (matches), processing it (filters), and forwarding the request to an API object (backendRefs).

Type

`object`

Property	Type	Description
`backendRefs`	`array`	BackendRefs defines the backend(s) where matching requests should be sent. Failure behavior here depends on how many BackendRefs are specified and how many are invalid. If *all* entries in BackendRefs are invalid, and there are also no filters specified in this route rule, *all* traffic which matches this rule MUST receive a 500 status code. See the HTTPBackendRef definition for the rules about what makes a single HTTPBackendRef invalid. When a HTTPBackendRef is invalid, 500 status codes MUST be returned for requests that would have otherwise been routed to an invalid backend. If multiple backends are specified, and some are invalid, the proportion of requests

Property	Type	Description
		that would otherwise have been routed to an invalid backend MUST receive a 500 status code. For example, if two backends are specified with equal weights, and one is invalid, 50 percent of traffic must receive a 500. Implementations may choose how that 50 percent is determined. When a HTTPBackendRef refers to a Service that has no ready endpoints, implementations SHOULD return a 503 for requests to that backend instead. If an implementation chooses to do this, all of the above rules for 500 responses MUST also apply for responses that return a 503. Support: Core for Kubernetes Service Support: Extended for Kubernetes ServiceImport Support: Implementation-specific for any other resource Support for weight: Core
`filters`	`array`	Filters define the filters that are applied to requests that match this rule. Wherever possible, implementations SHOULD implement filters in the order they are specified. Implementations MAY choose to implement this ordering strictly, rejecting any combination or order of filters that cannot be supported. If implementations choose a strict interpretation of filter ordering, they MUST clearly document that behavior. To reject an invalid combination or order of filters, implementations SHOULD consider the Route Rules with this

Property	Type	Description
		configuration invalid. If all Route Rules in a Route are invalid, the entire Route would be considered invalid. If only a portion of Route Rules are invalid, implementations MUST set the "PartiallyInvalid" condition for the Route. Conformance-levels at this level are defined based on the type of filter: • ALL core filters MUST be supported by all implementations. • Implementers are encouraged to support extended filters. • Implementation-specific custom filters have no API guarantees across implementations. Specifying the same filter multiple times is not supported unless explicitly indicated in the filter. All filters are expected to be compatible with each other except for the URLRewrite and RequestRedirect filters, which may not be combined. If an implementation cannot support other combinations of filters, they must clearly document that limitation. In cases where incompatible or unsupported filters are specified and cause the `Accepted` condition to be set to status `False`, implementations may use the `IncompatibleFilters` reason to specify this configuration error. Support: Core
`matches`	`array`	Matches define conditions used for matching the rule against incoming HTTP requests. Each match is independent, i.e. this rule will be matched if **any** one of the matches is satisfied. For example, take the following matches configuration:

Property	Type	Description

		```
		matches:
		- path:
		    value: "/foo"
		  headers:
		  - name: "version"
		    value: "v2"
		- path:
		    value: "/v2/foo"
		```
		
		For a request to match against this rule, a request must satisfy EITHER of the two conditions:
		
		• path prefixed with `/foo` AND contains the header `version: v2`
		• path prefix of `/v2/foo`
		
		See the documentation for HTTPRouteMatch on how to specify multiple match conditions that should be ANDed together.
		
		If no matches are specified, the default is a prefix path match on "/", which has the effect of matching every HTTP request.
		
		Proxy or Load Balancer routing configuration generated from HTTPRoutes MUST prioritize matches based on the following criteria, continuing on ties. Across all rules specified on applicable Routes, precedence must be given to the match having:
		
		• "Exact" path match.
		• "Prefix" path match with largest number of characters.
		• Method match.
		• Largest number of header matches.
		• Largest number of query param matches.

Property	Type	Description
		Note: The precedence of RegularExpression path matches are implementation-specific. If ties still exist across multiple Routes, matching precedence MUST be determined in order of the following criteria, continuing on ties: • The oldest Route based on creation timestamp. • The Route appearing first in alphabetical order by "{namespace}/{name}". If ties still exist within an HTTPRoute, matching precedence MUST be granted to the FIRST matching rule (in list order) with a match meeting the above criteria. When no rules matching a request have been successfully attached to the parent a request is coming from, a HTTP 404 status code MUST be returned.
`name`	`string`	Name is the name of the route rule. This name MUST be unique within a Route if it is set. Support: Extended
`timeouts`	`object`	Timeouts defines the timeouts that can be configured for an HTTP request. Support: Extended

.spec.rules[].backendRefs

Description

BackendRefs defines the backend(s) where matching requests should be sent. Failure behavior here depends on how many BackendRefs are specified and how many are invalid. If *all* entries in BackendRefs are invalid, and there are also no filters specified in this route rule, *all* traffic which matches this rule MUST receive a 500 status code. See the HTTPBackendRef definition for the rules about what makes a single HTTPBackendRef invalid. When a HTTPBackendRef is invalid, 500 status codes MUST be returned for requests that would have otherwise been routed to an invalid backend. If multiple backends are specified, and some are invalid, the proportion of requests that would otherwise have been routed to an invalid backend MUST receive a 500 status code. For example, if two backends are specified with equal weights, and one is invalid, 50 percent of traffic must receive a 500. Implementations may choose how that 50 percent is determined. When a HTTPBackendRef refers to a Service that has no ready endpoints, implementations SHOULD return a 503 for requests to that backend instead. If an implementation chooses to do this, all of the above rules for 500 responses MUST also apply for responses that return a 503. Support: Core for Kubernetes Service Support: Extended for Kubernetes ServiceImport Support: Implementation-specific for any other resource Support for weight: Core

Type

`array`

.spec.rules[].backendRefs[]

Description

HTTPBackendRef defines how a HTTPRoute forwards a HTTP request. Note that when a namespace different than the local namespace is specified, a ReferenceGrant object is required in the referent namespace to allow that namespace's owner to accept the reference. See the ReferenceGrant documentation for details.

Type

`object`

Required

`name`

Property	Type	Description
`filters`	`array`	Filters defined at this level should be executed if and only if the request is being forwarded to the backend defined here. Support: Implementation-specific (For broader support of filters, use the Filters field in HTTPRouteRule.)
`group`	`string`	Group is the group of the referent. For example, "gateway.networking.k8s.io". When unspecified or empty string, core API group is inferred.
`kind`	`string`	Kind is the Kubernetes resource kind of the referent. For example "Service". Defaults to "Service" when not specified. ExternalName services can refer to CNAME DNS records that may live outside of the cluster and as such are difficult to reason about in terms of conformance. They also may not be safe to forward to (see CVE-2021-25740 for more information). Implementations SHOULD NOT support ExternalName Services. Support: Core (Services with a type other than ExternalName) Support: Implementation-specific (Services with type ExternalName)
`name`	`string`	Name is the name of the referent.

Property	Type	Description
`namespace`	`string`	Namespace is the namespace of the backend. When unspecified, the local namespace is inferred. Note that when a namespace different than the local namespace is specified, a ReferenceGrant object is required in the referent namespace to allow that namespace's owner to accept the reference. See the ReferenceGrant documentation for details. Support: Core
`port`	`integer`	Port specifies the destination port number to use for this resource. Port is required when the referent is a Kubernetes Service. In this case, the port number is the service port number, not the target port. For other resources, destination port might be derived from the referent resource or this field.
`weight`	`integer`	Weight specifies the proportion of requests forwarded to the referenced backend. This is computed as weight/(sum of all weights in this BackendRefs list). For non-zero values, there may be some epsilon from the exact proportion defined here depending on the precision an implementation supports. Weight is not a percentage and the sum of weights does not need to equal 100. If only one backend is specified and it has a weight greater than 0, 100% of the traffic is forwarded to that backend. If weight is set to 0, no traffic should be forwarded for this entry. If unspecified, weight defaults to 1. Support for this field varies based on the context where used.

.spec.rules[].backendRefs[].filters

Description

Filters defined at this level should be executed if and only if the request is being forwarded to the backend defined here. Support: Implementation-specific (For broader support of filters, use the Filters field in HTTPRouteRule.)

Type

`array`

.spec.rules[].backendRefs[].filters[]

Description

HTTPRouteFilter defines processing steps that must be completed during the request or response lifecycle. HTTPRouteFilters are meant as an extension point to express processing that may be done in Gateway implementations. Some examples include request or response modification, implementing authentication strategies, rate-limiting, and traffic shaping. API guarantee/conformance is defined based on the type of the filter.

Type

`object`

Required

`type`

Property	Type	Description
`extensionRef`	`object`	ExtensionRef is an optional, implementation-specific extension to the "filter" behavior. For example, resource "myroutefilter" in group "networking.example.net"). ExtensionRef MUST NOT be used for core and extended filters. This filter can be used multiple times within the same rule. Support: Implementation-specific
`requestHeaderModifier`	`object`	RequestHeaderModifier defines a schema for a filter that modifies request headers. Support: Core
`requestMirror`	`object`	RequestMirror defines a schema for a filter that mirrors requests. Requests are sent to the specified destination, but responses from that destination are ignored. This filter can be used multiple times within the same rule. Note that not all implementations will be able to support mirroring to multiple backends. Support: Extended
`requestRedirect`	`object`	RequestRedirect defines a schema for a filter that responds to the request with an HTTP redirection.

Property	Type	Description
		Support: Core
`responseHeaderModifier`	`object`	ResponseHeaderModifier defines a schema for a filter that modifies response headers. Support: Extended
`type`	`string`	Type identifies the type of filter to apply. As with other API fields, types are classified into three conformance levels: • Core: Filter types and their corresponding configuration defined by "Support: Core" in this package, e.g. "RequestHeaderModifier". All implementations must support core filters. • Extended: Filter types and their corresponding configuration defined by "Support: Extended" in this package, e.g. "RequestMirror". Implementers are encouraged to support extended filters. • Implementation-specific: Filters that are defined and supported by specific vendors. In the future, filters showing convergence in behavior across multiple implementations will be considered for inclusion in extended or core conformance levels. Filter-specific configuration for such filters is specified using the ExtensionRef field. `Type` should be set to "ExtensionRef" for custom filters. Implementers are encouraged to define custom implementation types to extend the core API with implementation-specific behavior.

Property	Type	Description
		If a reference to a custom filter type cannot be resolved, the filter MUST NOT be skipped. Instead, requests that would have been processed by that filter MUST receive a HTTP error response. Note that values may be added to this enum, implementations must ensure that unknown values will not cause a crash. Unknown values here must result in the implementation setting the Accepted Condition for the Route to `status: False`, with a Reason of `UnsupportedValue`.
`urlRewrite`	`object`	URLRewrite defines a schema for a filter that modifies a request during forwarding. Support: Extended

.spec.rules[].backendRefs[].filters[].extensionRef

Description

ExtensionRef is an optional, implementation-specific extension to the "filter" behavior. For example, resource "myroutefilter" in group "networking.example.net"). ExtensionRef MUST NOT be used for core and extended filters. This filter can be used multiple times within the same rule. Support: Implementation-specific

Type

`object`

Required

`group` `kind` `name`

Property	Type	Description
`group`	`string`	Group is the group of the referent. For example, "gateway.networking.k8s.io". When unspecified or empty string, core API group is inferred.
`kind`	`string`	Kind is kind of the referent. For example "HTTPRoute" or "Service".
`name`	`string`	Name is the name of the referent.

.spec.rules[].backendRefs[].filters[].requestHeaderModifier

Description

RequestHeaderModifier defines a schema for a filter that modifies request headers.
Support: Core

Type

`object`

Property	Type	Description
`add`	`array`	Add adds the given header(s) (name, value) to the request before the action. It appends to any existing values associated with the header name. Input: GET /foo HTTP/1.1 my-header: foo Config: add: • name: "my-header" value: "bar,baz"

Property	Type	Description
		Output: GET /foo HTTP/1.1 my-header: foo,bar,baz
`remove`	`array`	Remove the given header(s) from the HTTP request before the action. The value of Remove is a list of HTTP header names. Note that the header names are case-insensitive (see https://datatracker.ietf.org/doc/html/rfc2616#section-4.2 ↗). Input: GET /foo HTTP/1.1 my-header1: foo my-header2: bar my-header3: baz Config: remove: ["my-header1", "my-header3"] Output: GET /foo HTTP/1.1 my-header2: bar
`set`	`array`	Set overwrites the request with the given header (name, value) before the action. Input: GET /foo HTTP/1.1 my-header: foo Config: set: • name: "my-header" value: "bar" Output: GET /foo HTTP/1.1 my-header: bar

.spec.rules[].backendRefs[].filters[].requestHeaderModifier.add

Description

Add adds the given header(s) (name, value) to the request before the action. It appends to any existing values associated with the header name. Input: GET /foo HTTP/1.1 my-header:

foo Config: add: - name: "my-header" value: "bar,baz" Output: GET /foo HTTP/1.1 my-header: foo,bar,baz

Type

`array`

.spec.rules[].backendRefs[].filters[].requestHeaderModifier.add[]

Description

HTTPHeader represents an HTTP Header name and value as defined by RFC 7230.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name is the name of the HTTP Header to be matched. Name matching MUST be case-insensitive. (See https://tools.ietf.org/html/rfc7230#section-3.2 ↗). If multiple entries specify equivalent header names, the first entry with an equivalent name MUST be considered for a match. Subsequent entries with an equivalent header name MUST be ignored. Due to the case-insensitivity of header names, "foo" and "Foo" are considered equivalent.
`value`	`string`	Value is the value of HTTP Header to be matched.

.spec.rules[].backendRefs[].filters[].requestHeaderModifier.remove

Description

Remove the given header(s) from the HTTP request before the action. The value of Remove is a list of HTTP header names. Note that the header names are case-insensitive (see https://datatracker.ietf.org/doc/html/rfc2616#section-4.2). Input: GET /foo HTTP/1.1 my-header1: foo my-header2: bar my-header3: baz Config: remove: ["my-header1", "my-header3"] Output: GET /foo HTTP/1.1 my-header2: bar

Type

`array`

.spec.rules[].backendRefs[].filters[].requestHeaderModifier.remove[]

Type

`string`

.spec.rules[].backendRefs[].filters[].requestHeaderModifier.set

Description

Set overwrites the request with the given header (name, value) before the action. Input: GET /foo HTTP/1.1 my-header: foo Config: set: - name: "my-header" value: "bar" Output: GET /foo HTTP/1.1 my-header: bar

Type

`array`

.spec.rules[].backendRefs[].filters[].requestHeaderModifier.set[]

Description

HTTPHeader represents an HTTP Header name and value as defined by RFC 7230.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name is the name of the HTTP Header to be matched. Name matching MUST be case-insensitive. (See https://tools.ietf.org/html/rfc7230#section-3.2 ↗). If multiple entries specify equivalent header names, the first entry with an equivalent name MUST be considered for a match. Subsequent entries with an equivalent header name MUST be ignored. Due to the case-insensitivity of header names, "foo" and "Foo" are considered equivalent.
`value`	`string`	Value is the value of HTTP Header to be matched.

.spec.rules[].backendRefs[].filters[].requestMirror

Description

RequestMirror defines a schema for a filter that mirrors requests. Requests are sent to the specified destination, but responses from that destination are ignored. This filter can be used multiple times within the same rule. Note that not all implementations will be able to support mirroring to multiple backends. Support: Extended

Type

`object`

Required

`backendRef`

Property	Type	Description
`backendRef`	`object`	BackendRef references a resource where mirrored requests are sent. Mirrored requests must be sent only to a single destination endpoint within this BackendRef, irrespective of how many endpoints are present within this BackendRef. If the referent cannot be found, this BackendRef is invalid and must be dropped from the Gateway. The controller must ensure the "ResolvedRefs" condition on the Route status is set to `status: False` and not configure this backend in the underlying implementation. If there is a cross-namespace reference to an *existing* object that is not allowed by a ReferenceGrant, the controller must ensure the "ResolvedRefs" condition on the Route is set to `status: False`, with the "RefNotPermitted" reason and not configure this backend in the underlying implementation. In either error case, the Message of the `ResolvedRefs` Condition should be used to provide more detail about the problem. Support: Extended for Kubernetes Service Support: Implementation-specific for any other resource
`fraction`	`object`	Fraction represents the fraction of requests that should be mirrored to BackendRef. Only one of Fraction or Percent may be specified. If neither field is specified, 100% of requests will be mirrored.

Property	Type	Description
`percent`	`integer`	Percent represents the percentage of requests that should be mirrored to BackendRef. Its minimum value is 0 (indicating 0% of requests) and its maximum value is 100 (indicating 100% of requests). Only one of Fraction or Percent may be specified. If neither field is specified, 100% of requests will be mirrored.

.spec.rules[].backendRefs[].filters[].requestMirror.backendRef

Description

BackendRef references a resource where mirrored requests are sent. Mirrored requests must be sent only to a single destination endpoint within this BackendRef, irrespective of how many endpoints are present within this BackendRef. If the referent cannot be found, this BackendRef is invalid and must be dropped from the Gateway. The controller must ensure the "ResolvedRefs" condition on the Route status is set to `status: False` and not configure this backend in the underlying implementation. If there is a cross-namespace reference to an *existing* object that is not allowed by a ReferenceGrant, the controller must ensure the "ResolvedRefs" condition on the Route is set to `status: False`, with the "RefNotPermitted" reason and not configure this backend in the underlying implementation. In either error case, the Message of the `ResolvedRefs` Condition should be used to provide more detail about the problem. Support: Extended for Kubernetes Service Support: Implementation-specific for any other resource

Type

`object`

Required

`name`

Property	Type	Description
`group`	`string`	Group is the group of the referent. For example, "gateway.networking.k8s.io". When unspecified or empty string, core API group is inferred.
`kind`	`string`	Kind is the Kubernetes resource kind of the referent. For example "Service". Defaults to "Service" when not specified. ExternalName services can refer to CNAME DNS records that may live outside of the cluster and as such are difficult to reason about in terms of conformance. They also may not be safe to forward to (see CVE-2021-25740 for more information). Implementations SHOULD NOT support ExternalName Services. Support: Core (Services with a type other than ExternalName) Support: Implementation-specific (Services with type ExternalName)
`name`	`string`	Name is the name of the referent.
`namespace`	`string`	Namespace is the namespace of the backend. When unspecified, the local namespace is inferred. Note that when a namespace different than the local namespace is specified, a ReferenceGrant object is required in the referent namespace to allow that namespace's owner to accept the reference. See the ReferenceGrant documentation for details.

Property	Type	Description
		Support: Core
`port`	`integer`	Port specifies the destination port number to use for this resource. Port is required when the referent is a Kubernetes Service. In this case, the port number is the service port number, not the target port. For other resources, destination port might be derived from the referent resource or this field.

.spec.rules[].backendRefs[].filters[].requestMirror.fraction

Description

Fraction represents the fraction of requests that should be mirrored to BackendRef. Only one of Fraction or Percent may be specified. If neither field is specified, 100% of requests will be mirrored.

Type

`object`

Required

`numerator`

Property	Type	Description
`denominator`	`integer`	
`numerator`	`integer`	

.spec.rules[].backendRefs[].filters[].requestRedirect

Description

RequestRedirect defines a schema for a filter that responds to the request with an HTTP redirection. Support: Core

Type

`object`

Property	Type	Description
`hostname`	`string`	Hostname is the hostname to be used in the value of the `Location` header in the response. When empty, the hostname in the `Host` header of the request is used. Support: Core
`path`	`object`	Path defines parameters used to modify the path of the incoming request. The modified path is then used to construct the `Location` header. When empty, the request path is used as-is. Support: Extended
`port`	`integer`	Port is the port to be used in the value of the `Location` header in the response. If no port is specified, the redirect port MUST be derived using the following rules: • If redirect scheme is not-empty, the redirect port MUST be the well-known port associated with the redirect scheme. Specifically "http" to port 80 and "https" to port 443. If the redirect scheme does not have a well-known port, the listener port of the Gateway SHOULD be used. • If redirect scheme is empty, the redirect port MUST be the Gateway Listener port. Implementations SHOULD NOT add the port number in the 'Location' header in the following cases:

Property	Type	Description
		• A Location header that will use HTTP (whether that is determined via the Listener protocol or the Scheme field) *and* use port 80. • A Location header that will use HTTPS (whether that is determined via the Listener protocol or the Scheme field) *and* use port 443. Support: Extended
`scheme`	`string`	Scheme is the scheme to be used in the value of the `Location` header in the response. When empty, the scheme of the request is used. Scheme redirects can affect the port of the redirect, for more information, refer to the documentation for the port field of this filter. Note that values may be added to this enum, implementations must ensure that unknown values will not cause a crash. Unknown values here must result in the implementation setting the Accepted Condition for the Route to `status: False`, with a Reason of `UnsupportedValue`. Support: Extended
`statusCode`	`integer`	StatusCode is the HTTP status code to be used in response. Note that values may be added to this enum, implementations must ensure that unknown values will not cause a crash.

Property	Type	Description
		Unknown values here must result in the implementation setting the Accepted Condition for the Route to `status: False`, with a Reason of `UnsupportedValue`. Support: Core

.spec.rules[].backendRefs[].filters[].requestRedirect.path

Description

Path defines parameters used to modify the path of the incoming request. The modified path is then used to construct the `Location` header. When empty, the request path is used as-is. Support: Extended

Type

`object`

Required

`type`

Property	Type	Description
`replaceFullPath`	`string`	ReplaceFullPath specifies the value with which to replace the full path of a request during a rewrite or redirect.
`replacePrefixMatch`	`string`	ReplacePrefixMatch specifies the value with which to replace the prefix match of a request during a rewrite or redirect. For example, a request to "/foo/bar" with a prefix match of "/foo" and a ReplacePrefixMatch of "/xyz" would be modified to "/xyz/bar". Note that this matches the behavior of the PathPrefix match type. This matches full path elements. A path

Property	Type	Description
		element refers to the list of labels in the path split by the `/` separator. When specified, a trailing `/` is ignored. For example, the paths `/abc`, `/abc/`, and `/abc/def` would all match the prefix `/abc`, but the path `/abcd` would not. ReplacePrefixMatch is only compatible with a `PathPrefix` HTTPRouteMatch. Using any other HTTPRouteMatch type on the same HTTPRouteRule will result in the implementation setting the Accepted Condition for the Route to `status: False`. Request Path | Prefix Match | Replace Prefix | Modified Path
`type`	`string`	Type defines the type of path modifier. Additional types may be added in a future release of the API. Note that values may be added to this enum, implementations must ensure that unknown values will not cause a crash. Unknown values here must result in the implementation setting the Accepted Condition for the Route to `status: False`, with a Reason of `UnsupportedValue`.

.spec.rules[].backendRefs[].filters[].responseHeaderModifier

Description

ResponseHeaderModifier defines a schema for a filter that modifies response headers.
Support: Extended

Type

`object`

Property	Type	Description
`add`	`array`	Add adds the given header(s) (name, value) to the request before the action. It appends to any existing values associated with the header name. Input: GET /foo HTTP/1.1 my-header: foo Config: add: • name: "my-header" value: "bar,baz" Output: GET /foo HTTP/1.1 my-header: foo,bar,baz
`remove`	`array`	Remove the given header(s) from the HTTP request before the action. The value of Remove is a list of HTTP header names. Note that the header names are case-insensitive (see https://datatracker.ietf.org/doc/html/rfc2616#section-4.2). Input: GET /foo HTTP/1.1 my-header1: foo my-header2: bar my-header3: baz Config: remove: ["my-header1", "my-header3"] Output: GET /foo HTTP/1.1 my-header2: bar
`set`	`array`	Set overwrites the request with the given header (name, value) before the action. Input: GET /foo HTTP/1.1 my-header: foo

Property	Type	Description
		Config: set: • name: "my-header" value: "bar" Output: GET /foo HTTP/1.1 my-header: bar

.spec.rules[].backendRefs[].filters[].responseHeaderModifier.add

Description

Add adds the given header(s) (name, value) to the request before the action. It appends to any existing values associated with the header name. Input: GET /foo HTTP/1.1 my-header: foo Config: add: - name: "my-header" value: "bar,baz" Output: GET /foo HTTP/1.1 my-header: foo,bar,baz

Type

`array`

.spec.rules[].backendRefs[].filters[].responseHeaderModifier.add[]

Description

HTTPHeader represents an HTTP Header name and value as defined by RFC 7230.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name is the name of the HTTP Header to be matched. Name matching MUST be case-insensitive. (See https://tools.ietf.org/html/rfc7230#section-3.2). If multiple entries specify equivalent header names, the first entry with an equivalent name MUST be considered for a match. Subsequent entries with an equivalent header name MUST be ignored. Due to the case-insensitivity of header names, "foo" and "Foo" are considered equivalent.
`value`	`string`	Value is the value of HTTP Header to be matched.

.spec.rules[].backendRefs[].filters[].responseHeaderModifier.remove

Description

Remove the given header(s) from the HTTP request before the action. The value of Remove is a list of HTTP header names. Note that the header names are case-insensitive (see https://datatracker.ietf.org/doc/html/rfc2616#section-4.2). Input: GET /foo HTTP/1.1 my-header1: foo my-header2: bar my-header3: baz Config: remove: ["my-header1", "my-header3"] Output: GET /foo HTTP/1.1 my-header2: bar

Type

`array`

.spec.rules[].backendRefs[].filters[].responseHeaderModifier.remove[]

Type

`string`

.spec.rules[].backendRefs[].filters[].responseHeaderModifier.set

Description

Set overwrites the request with the given header (name, value) before the action. Input: GET /foo HTTP/1.1 my-header: foo Config: set: - name: "my-header" value: "bar" Output: GET /foo HTTP/1.1 my-header: bar

Type

`array`

.spec.rules[].backendRefs[].filters[].responseHeaderModifier.set[]

Description

HTTPHeader represents an HTTP Header name and value as defined by RFC 7230.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name is the name of the HTTP Header to be matched. Name matching MUST be case-insensitive. (See https://tools.ietf.org/html/rfc7230#section-3.2). If multiple entries specify equivalent header names, the first entry with an equivalent name MUST be considered for a match. Subsequent entries with an equivalent header name MUST be ignored. Due to the case-insensitivity of header names, "foo" and "Foo" are considered equivalent.

Property	Type	Description
`value`	`string`	Value is the value of HTTP Header to be matched.

.spec.rules[].backendRefs[].filters[].urlRewrite

Description

URLRewrite defines a schema for a filter that modifies a request during forwarding. Support: Extended

Type

`object`

Property	Type	Description
`hostname`	`string`	Hostname is the value to be used to replace the Host header value during forwarding. Support: Extended
`path`	`object`	Path defines a path rewrite. Support: Extended

.spec.rules[].backendRefs[].filters[].urlRewrite.path

Description

Path defines a path rewrite. Support: Extended

Type

`object`

Required

`type`

Property	Type	Description
`replaceFullPath`	`string`	ReplaceFullPath specifies the value with which to replace the full path of a request during a rewrite or redirect.
`replacePrefixMatch`	`string`	ReplacePrefixMatch specifies the value with which to replace the prefix match of a request during a rewrite or redirect. For example, a request to "/foo/bar" with a prefix match of "/foo" and a ReplacePrefixMatch of "/xyz" would be modified to "/xyz/bar". Note that this matches the behavior of the PathPrefix match type. This matches full path elements. A path element refers to the list of labels in the path split by the `/` separator. When specified, a trailing `/` is ignored. For example, the paths `/abc`, `/abc/`, and `/abc/def` would all match the prefix `/abc`, but the path `/abcd` would not. ReplacePrefixMatch is only compatible with a `PathPrefix` HTTPRouteMatch. Using any other HTTPRouteMatch type on the same HTTPRouteRule will result in the implementation setting the Accepted Condition for the Route to `status: False`. Request Path | Prefix Match | Replace Prefix | Modified Path
`type`	`string`	Type defines the type of path modifier. Additional types may be added in a future release of the API.

Property	Type	Description
		Note that values may be added to this enum, implementations must ensure that unknown values will not cause a crash. Unknown values here must result in the implementation setting the Accepted Condition for the Route to `status: False`, with a Reason of `UnsupportedValue`.

.spec.rules[].filters

Description

Filters define the filters that are applied to requests that match this rule. Wherever possible, implementations SHOULD implement filters in the order they are specified. Implementations MAY choose to implement this ordering strictly, rejecting any combination or order of filters that cannot be supported. If implementations choose a strict interpretation of filter ordering, they MUST clearly document that behavior. To reject an invalid combination or order of filters, implementations SHOULD consider the Route Rules with this configuration invalid. If all Route Rules in a Route are invalid, the entire Route would be considered invalid. If only a portion of Route Rules are invalid, implementations MUST set the "PartiallyInvalid" condition for the Route. Conformance-levels at this level are defined based on the type of filter: - ALL core filters MUST be supported by all implementations. - Implementers are encouraged to support extended filters. - Implementation-specific custom filters have no API guarantees across implementations. Specifying the same filter multiple times is not supported unless explicitly indicated in the filter. All filters are expected to be compatible with each other except for the URLRewrite and RequestRedirect filters, which may not be combined. If an implementation cannot support other combinations of filters, they must clearly document that limitation. In cases where incompatible or unsupported filters are specified and cause the \`Accepted\` condition to be set to status \`False\`, implementations may use the \`IncompatibleFilters\` reason to specify this configuration error. Support: Core

Type

`array`

.spec.rules[].filters[]

Description

HTTPRouteFilter defines processing steps that must be completed during the request or response lifecycle. HTTPRouteFilters are meant as an extension point to express processing that may be done in Gateway implementations. Some examples include request or response modification, implementing authentication strategies, rate-limiting, and traffic shaping. API guarantee/conformance is defined based on the type of the filter.

Type

`object`

Required

`type`

Property	Type	Description
`extensionRef`	`object`	ExtensionRef is an optional, implementation-specific extension to the "filter" behavior. For example, resource "myroutefilter" in group "networking.example.net"). ExtensionRef MUST NOT be used for core and extended filters. This filter can be used multiple times within the same rule. Support: Implementation-specific
`requestHeaderModifier`	`object`	RequestHeaderModifier defines a schema for a filter that modifies request headers. Support: Core
`requestMirror`	`object`	RequestMirror defines a schema for a filter that mirrors requests. Requests are sent to the

Property	Type	Description
		specified destination, but responses from that destination are ignored. This filter can be used multiple times within the same rule. Note that not all implementations will be able to support mirroring to multiple backends. Support: Extended
`requestRedirect`	`object`	RequestRedirect defines a schema for a filter that responds to the request with an HTTP redirection. Support: Core
`responseHeaderModifier`	`object`	ResponseHeaderModifier defines a schema for a filter that modifies response headers. Support: Extended
`type`	`string`	Type identifies the type of filter to apply. As with other API fields, types are classified into three conformance levels: • Core: Filter types and their corresponding configuration defined by "Support: Core" in this package, e.g. "RequestHeaderModifier". All implementations must support core filters. • Extended: Filter types and their corresponding configuration defined by "Support: Extended" in this package, e.g.

Property	Type	Description
		"RequestMirror". Implementers are encouraged to support extended filters. • Implementation-specific: Filters that are defined and supported by specific vendors. In the future, filters showing convergence in behavior across multiple implementations will be considered for inclusion in extended or core conformance levels. Filter-specific configuration for such filters is specified using the ExtensionRef field. `Type` should be set to "ExtensionRef" for custom filters. Implementers are encouraged to define custom implementation types to extend the core API with implementation-specific behavior. If a reference to a custom filter type cannot be resolved, the filter MUST NOT be skipped. Instead, requests that would have been processed by that filter MUST receive a HTTP error response. Note that values may be added to this enum, implementations must ensure that unknown values will not cause a crash. Unknown values here must result in the implementation setting the Accepted Condition for the Route to `status: False`, with a Reason of `UnsupportedValue`.

Property	Type	Description
`urlRewrite`	`object`	URLRewrite defines a schema for a filter that modifies a request during forwarding. Support: Extended

.spec.rules[].filters[].extensionRef

Description

ExtensionRef is an optional, implementation-specific extension to the "filter" behavior. For example, resource "myroutefilter" in group "networking.example.net"). ExtensionRef MUST NOT be used for core and extended filters. This filter can be used multiple times within the same rule. Support: Implementation-specific

Type

`object`

Required

`group` `kind` `name`

Property	Type	Description
`group`	`string`	Group is the group of the referent. For example, "gateway.networking.k8s.io". When unspecified or empty string, core API group is inferred.
`kind`	`string`	Kind is kind of the referent. For example "HTTPRoute" or "Service".
`name`	`string`	Name is the name of the referent.

.spec.rules[].filters[].requestHeaderModifier

Description

RequestHeaderModifier defines a schema for a filter that modifies request headers.
Support: Core

Type

`object`

Property	Type	Description
`add`	`array`	Add adds the given header(s) (name, value) to the request before the action. It appends to any existing values associated with the header name. Input: GET /foo HTTP/1.1 my-header: foo Config: add: • name: "my-header" value: "bar,baz" Output: GET /foo HTTP/1.1 my-header: foo,bar,baz
`remove`	`array`	Remove the given header(s) from the HTTP request before the action. The value of Remove is a list of HTTP header names. Note that the header names are case-insensitive (see https://datatracker.ietf.org/doc/html/rfc2616#section-4.2). Input: GET /foo HTTP/1.1 my-header1: foo my-header2: bar my-header3: baz Config: remove: ["my-header1", "my-header3"] Output: GET /foo HTTP/1.1 my-header2: bar

Property	Type	Description
`set`	`array`	Set overwrites the request with the given header (name, value) before the action. Input: GET /foo HTTP/1.1 my-header: foo Config: set: • name: "my-header" value: "bar" Output: GET /foo HTTP/1.1 my-header: bar

.spec.rules[].filters[].requestHeaderModifier.add

Description

Add adds the given header(s) (name, value) to the request before the action. It appends to any existing values associated with the header name. Input: GET /foo HTTP/1.1 my-header: foo Config: add: - name: "my-header" value: "bar,baz" Output: GET /foo HTTP/1.1 my-header: foo,bar,baz

Type

`array`

.spec.rules[].filters[].requestHeaderModifier.add[]

Description

HTTPHeader represents an HTTP Header name and value as defined by RFC 7230.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name is the name of the HTTP Header to be matched. Name matching MUST be case-insensitive. (See https://tools.ietf.org/html/rfc7230#section-3.2). If multiple entries specify equivalent header names, the first entry with an equivalent name MUST be considered for a match. Subsequent entries with an equivalent header name MUST be ignored. Due to the case-insensitivity of header names, "foo" and "Foo" are considered equivalent.
`value`	`string`	Value is the value of HTTP Header to be matched.

.spec.rules[].filters[].requestHeaderModifier.remove

Description

Remove the given header(s) from the HTTP request before the action. The value of Remove is a list of HTTP header names. Note that the header names are case-insensitive (see https://datatracker.ietf.org/doc/html/rfc2616#section-4.2). Input: GET /foo HTTP/1.1 my-header1: foo my-header2: bar my-header3: baz Config: remove: ["my-header1", "my-header3"] Output: GET /foo HTTP/1.1 my-header2: bar

Type

`array`

.spec.rules[].filters[].requestHeaderModifier.remove[]

Type

`string`

.spec.rules[].filters[].requestHeaderModifier.set

Description

Set overwrites the request with the given header (name, value) before the action. Input: GET /foo HTTP/1.1 my-header: foo Config: set: - name: "my-header" value: "bar" Output: GET /foo HTTP/1.1 my-header: bar

Type

`array`

.spec.rules[].filters[].requestHeaderModifier.set[]

Description

HTTPHeader represents an HTTP Header name and value as defined by RFC 7230.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name is the name of the HTTP Header to be matched. Name matching MUST be case-insensitive. (See https://tools.ietf.org/html/rfc7230#section-3.2). If multiple entries specify equivalent header names, the first entry with an equivalent name MUST be considered for a match. Subsequent entries with an equivalent header name MUST be ignored. Due to the case-insensitivity of header names, "foo" and "Foo" are considered equivalent.
`value`	`string`	Value is the value of HTTP Header to be matched.

.spec.rules[].filters[].requestMirror

Description

RequestMirror defines a schema for a filter that mirrors requests. Requests are sent to the specified destination, but responses from that destination are ignored. This filter can be used multiple times within the same rule. Note that not all implementations will be able to support mirroring to multiple backends. Support: Extended

Type

`object`

Required

`backendRef`

Property	Type	Description
`backendRef`	`object`	BackendRef references a resource where mirrored requests are sent. Mirrored requests must be sent only to a single destination endpoint within this BackendRef, irrespective of how many endpoints are present within this BackendRef. If the referent cannot be found, this BackendRef is invalid and must be dropped from the Gateway. The controller must ensure the "ResolvedRefs" condition on the Route status is set to `status: False` and not configure this backend in the underlying implementation. If there is a cross-namespace reference to an *existing* object that is not allowed by a ReferenceGrant, the controller must ensure the "ResolvedRefs" condition on the Route is set to `status: False`, with the "RefNotPermitted" reason and not configure this backend in the underlying implementation. In either error case, the Message of the `ResolvedRefs` Condition should be used to provide more detail about the problem. Support: Extended for Kubernetes Service

Property	Type	Description
		Support: Implementation-specific for any other resource
`fraction`	`object`	Fraction represents the fraction of requests that should be mirrored to BackendRef. Only one of Fraction or Percent may be specified. If neither field is specified, 100% of requests will be mirrored.
`percent`	`integer`	Percent represents the percentage of requests that should be mirrored to BackendRef. Its minimum value is 0 (indicating 0% of requests) and its maximum value is 100 (indicating 100% of requests). Only one of Fraction or Percent may be specified. If neither field is specified, 100% of requests will be mirrored.

.spec.rules[].filters[].requestMirror.backendRef

Description

BackendRef references a resource where mirrored requests are sent. Mirrored requests must be sent only to a single destination endpoint within this BackendRef, irrespective of how many endpoints are present within this BackendRef. If the referent cannot be found, this BackendRef is invalid and must be dropped from the Gateway. The controller must ensure the "ResolvedRefs" condition on the Route status is set to `status: False` and not configure this backend in the underlying implementation. If there is a cross-namespace reference to an *existing* object that is not allowed by a ReferenceGrant, the controller must ensure the "ResolvedRefs" condition on the Route is set to `status: False`, with the "RefNotPermitted" reason and not configure this backend in the underlying implementation. In either error case, the Message of the `ResolvedRefs` Condition should be used to provide more detail about the problem. Support: Extended for Kubernetes Service Support: Implementation-specific for any other resource

Type

`object`

Required

`name`

Property	Type	Description
`group`	`string`	Group is the group of the referent. For example, "gateway.networking.k8s.io". When unspecified or empty string, core API group is inferred.
`kind`	`string`	Kind is the Kubernetes resource kind of the referent. For example "Service". Defaults to "Service" when not specified. ExternalName services can refer to CNAME DNS records that may live outside of the cluster and as such are difficult to reason about in terms of conformance. They also may not be safe to forward to (see CVE-2021-25740 for more information). Implementations SHOULD NOT support ExternalName Services. Support: Core (Services with a type other than ExternalName) Support: Implementation-specific (Services with type ExternalName)
`name`	`string`	Name is the name of the referent.
`namespace`	`string`	Namespace is the namespace of the backend. When unspecified, the local namespace is inferred.

Property	Type	Description
		Note that when a namespace different than the local namespace is specified, a ReferenceGrant object is required in the referent namespace to allow that namespace's owner to accept the reference. See the ReferenceGrant documentation for details. Support: Core
`port`	`integer`	Port specifies the destination port number to use for this resource. Port is required when the referent is a Kubernetes Service. In this case, the port number is the service port number, not the target port. For other resources, destination port might be derived from the referent resource or this field.

.spec.rules[].filters[].requestMirror.fraction

Description

Fraction represents the fraction of requests that should be mirrored to BackendRef. Only one of Fraction or Percent may be specified. If neither field is specified, 100% of requests will be mirrored.

Type

`object`

Required

`numerator`

Property	Type	Description
`denominator`	`integer`	
`numerator`	`integer`	

.spec.rules[].filters[].requestRedirect

Description

RequestRedirect defines a schema for a filter that responds to the request with an HTTP redirection. Support: Core

Type

`object`

Property	Type	Description
`hostname`	`string`	Hostname is the hostname to be used in the value of the `Location` header in the response. When empty, the hostname in the `Host` header of the request is used. Support: Core
`path`	`object`	Path defines parameters used to modify the path of the incoming request. The modified path is then used to construct the `Location` header. When empty, the request path is used as-is. Support: Extended
`port`	`integer`	Port is the port to be used in the value of the `Location` header in the response. If no port is specified, the redirect port MUST be derived using the following rules: • If redirect scheme is not-empty, the redirect port MUST be the well-known port associated with the redirect scheme. Specifically "http" to port 80 and "https" to port 443. If the

Property	Type	Description
		• redirect scheme does not have a well-known port, the listener port of the Gateway SHOULD be used. • If redirect scheme is empty, the redirect port MUST be the Gateway Listener port. Implementations SHOULD NOT add the port number in the 'Location' header in the following cases: • A Location header that will use HTTP (whether that is determined via the Listener protocol or the Scheme field) *and* use port 80. • A Location header that will use HTTPS (whether that is determined via the Listener protocol or the Scheme field) *and* use port 443. Support: Extended
`scheme`	`string`	Scheme is the scheme to be used in the value of the `Location` header in the response. When empty, the scheme of the request is used. Scheme redirects can affect the port of the redirect, for more information, refer to the documentation for the port field of this filter. Note that values may be added to this enum, implementations must ensure that unknown values will not cause a crash. Unknown values here must result in the implementation setting the Accepted Condition for the Route to `status: False`, with a Reason of `UnsupportedValue`. Support: Extended

Property	Type	Description
`statusCode`	`integer`	StatusCode is the HTTP status code to be used in response. Note that values may be added to this enum, implementations must ensure that unknown values will not cause a crash. Unknown values here must result in the implementation setting the Accepted Condition for the Route to `status: False`, with a Reason of `UnsupportedValue`. Support: Core

.spec.rules[].filters[].requestRedirect.path

Description

Path defines parameters used to modify the path of the incoming request. The modified path is then used to construct the `Location` header. When empty, the request path is used as-is. Support: Extended

Type

`object`

Required

`type`

Property	Type	Description
`replaceFullPath`	`string`	ReplaceFullPath specifies the value with which to replace the full path of a request during a rewrite or redirect.

Property	Type	Description
`replacePrefixMatch`	`string`	ReplacePrefixMatch specifies the value with which to replace the prefix match of a request during a rewrite or redirect. For example, a request to "/foo/bar" with a prefix match of "/foo" and a ReplacePrefixMatch of "/xyz" would be modified to "/xyz/bar". Note that this matches the behavior of the PathPrefix match type. This matches full path elements. A path element refers to the list of labels in the path split by the `/` separator. When specified, a trailing `/` is ignored. For example, the paths `/abc`, `/abc/`, and `/abc/def` would all match the prefix `/abc`, but the path `/abcd` would not. ReplacePrefixMatch is only compatible with a `PathPrefix` HTTPRouteMatch. Using any other HTTPRouteMatch type on the same HTTPRouteRule will result in the implementation setting the Accepted Condition for the Route to `status: False`. Request Path | Prefix Match | Replace Prefix | Modified Path
`type`	`string`	Type defines the type of path modifier. Additional types may be added in a future release of the API. Note that values may be added to this enum, implementations must ensure that unknown values will not cause a crash. Unknown values here must result in the implementation setting the Accepted Condition for

Property	Type	Description
		the Route to `status: False`, with a Reason of `UnsupportedValue`.

.spec.rules[].filters[].responseHeaderModifier

Description

ResponseHeaderModifier defines a schema for a filter that modifies response headers. Support: Extended

Type

`object`

Property	Type	Description
`add`	`array`	Add adds the given header(s) (name, value) to the request before the action. It appends to any existing values associated with the header name. Input: GET /foo HTTP/1.1 my-header: foo Config: add: • name: "my-header" value: "bar,baz" Output: GET /foo HTTP/1.1 my-header: foo,bar,baz
`remove`	`array`	Remove the given header(s) from the HTTP request before the action. The value of Remove is a list of HTTP header names. Note that the header names are case-insensitive (see https://datatracker.ietf.org/doc/html/rfc2616#section-4.2 ↗). Input: GET /foo HTTP/1.1 my-header1: foo my-header2: bar my-header3: baz

Property	Type	Description
		Config: remove: ["my-header1", "my-header3"] Output: GET /foo HTTP/1.1 my-header2: bar
`set`	`array`	Set overwrites the request with the given header (name, value) before the action. Input: GET /foo HTTP/1.1 my-header: foo Config: set: • name: "my-header" value: "bar" Output: GET /foo HTTP/1.1 my-header: bar

.spec.rules[].filters[].responseHeaderModifier.add

Description

Add adds the given header(s) (name, value) to the request before the action. It appends to any existing values associated with the header name. Input: GET /foo HTTP/1.1 my-header: foo Config: add: - name: "my-header" value: "bar,baz" Output: GET /foo HTTP/1.1 my-header: foo,bar,baz

Type

`array`

.spec.rules[].filters[].responseHeaderModifier.add[]

Description

HTTPHeader represents an HTTP Header name and value as defined by RFC 7230.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name is the name of the HTTP Header to be matched. Name matching MUST be case-insensitive. (See https://tools.ietf.org/html/rfc7230#section-3.2). If multiple entries specify equivalent header names, the first entry with an equivalent name MUST be considered for a match. Subsequent entries with an equivalent header name MUST be ignored. Due to the case-insensitivity of header names, "foo" and "Foo" are considered equivalent.
`value`	`string`	Value is the value of HTTP Header to be matched.

.spec.rules[].filters[].responseHeaderModifier.remove

Description

Remove the given header(s) from the HTTP request before the action. The value of Remove is a list of HTTP header names. Note that the header names are case-insensitive (see https://datatracker.ietf.org/doc/html/rfc2616#section-4.2). Input: GET /foo HTTP/1.1 my-header1: foo my-header2: bar my-header3: baz Config: remove: ["my-header1", "my-header3"] Output: GET /foo HTTP/1.1 my-header2: bar

Type

`array`

.spec.rules[].filters[].responseHeaderModifier.remove[]

Type

`string`

.spec.rules[].filters[].responseHeaderModifier.set

Description

Set overwrites the request with the given header (name, value) before the action. Input: GET /foo HTTP/1.1 my-header: foo Config: set: - name: "my-header" value: "bar" Output: GET /foo HTTP/1.1 my-header: bar

Type

`array`

.spec.rules[].filters[].responseHeaderModifier.set[]

Description

HTTPHeader represents an HTTP Header name and value as defined by RFC 7230.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name is the name of the HTTP Header to be matched. Name matching MUST be case-insensitive. (See https://tools.ietf.org/html/rfc7230#section-3.2 ↗). If multiple entries specify equivalent header names, the first entry with an equivalent name MUST be considered for a match. Subsequent entries with an equivalent header name MUST be ignored. Due to the case-insensitivity of header names, "foo" and "Foo" are considered equivalent.
`value`	`string`	Value is the value of HTTP Header to be matched.

.spec.rules[].filters[].urlRewrite

Description

URLRewrite defines a schema for a filter that modifies a request during forwarding. Support: Extended

Type

`object`

Property	Type	Description
`hostname`	`string`	Hostname is the value to be used to replace the Host header value during forwarding. Support: Extended
`path`	`object`	Path defines a path rewrite. Support: Extended

.spec.rules[].filters[].urlRewrite.path

Description

Path defines a path rewrite. Support: Extended

Type

`object`

Required

`type`

Property	Type	Description
`replaceFullPath`	`string`	ReplaceFullPath specifies the value with which to replace the full path of a request during a rewrite or redirect.
`replacePrefixMatch`	`string`	ReplacePrefixMatch specifies the value with which to replace the prefix match of a request during a rewrite or redirect. For example, a request to "/foo/bar" with a prefix match of "/foo" and a ReplacePrefixMatch of "/xyz" would be modified to "/xyz/bar". Note that this matches the behavior of the PathPrefix match type. This matches full path elements. A path element refers to the list of labels in the path split by the `/` separator. When specified, a trailing `/` is ignored. For example, the paths `/abc`, `/abc/`, and `/abc/def` would all match the prefix `/abc`, but the path `/abcd` would not. ReplacePrefixMatch is only compatible with a `PathPrefix` HTTPRouteMatch. Using any other HTTPRouteMatch type on the same HTTPRouteRule will result in the implementation setting the Accepted Condition for the Route to `status: False`. Request Path | Prefix Match | Replace Prefix | Modified Path
`type`	`string`	Type defines the type of path modifier. Additional types may be added in a future release of the API. Note that values may be added to this enum, implementations must ensure that unknown values

Property	Type	Description
		will not cause a crash. Unknown values here must result in the implementation setting the Accepted Condition for the Route to `status: False`, with a Reason of `UnsupportedValue`.

.spec.rules[].matches

Description

Matches define conditions used for matching the rule against incoming HTTP requests. Each match is independent, i.e. this rule will be matched if **any** one of the matches is satisfied. For example, take the following matches configuration: ``` matches: - path: value: "/foo" headers: - name: "version" value: "v2" - path: value: "/v2/foo" ``` For a request to match against this rule, a request must satisfy EITHER of the two conditions: - path prefixed with `/foo` AND contains the header `version: v2` - path prefix of `/v2/foo` See the documentation for HTTPRouteMatch on how to specify multiple match conditions that should be ANDed together. If no matches are specified, the default is a prefix path match on "/", which has the effect of matching every HTTP request. Proxy or Load Balancer routing configuration generated from HTTPRoutes MUST prioritize matches based on the following criteria, continuing on ties. Across all rules specified on applicable Routes, precedence must be given to the match having: * "Exact" path match. * "Prefix" path match with largest number of characters. * Method match. * Largest number of header matches. * Largest number of query param matches. Note: The precedence of RegularExpression path matches are implementation-specific. If ties still exist across multiple Routes, matching precedence MUST be determined in order of the following criteria, continuing on ties: * The oldest Route based on creation timestamp. * The Route appearing first in alphabetical order by "{namespace}/{name}". If ties still exist within an HTTPRoute, matching precedence MUST be granted to the FIRST matching rule (in list order) with a match meeting the above criteria. When no rules matching a request have been successfully attached to the parent a request is coming from, a HTTP 404 status code MUST be returned.

Type

`array`

.spec.rules[].matches[]

Description

HTTPRouteMatch defines the predicate used to match requests to a given action. Multiple match types are ANDed together, i.e. the match will evaluate to true only if all conditions are satisfied. For example, the match below will match a HTTP request only if its path starts with `/foo` AND it contains the `version: v1` header: ``` match: path: value: "/foo" headers: - name: "version" value "v1" ```

Type

`object`

Property	Type	Description
`headers`	`array`	Headers specifies HTTP request header matchers. Multiple match values are ANDed together, meaning, a request must match all the specified headers to select the route.
`method`	`string`	Method specifies HTTP method matcher. When specified, this route will be matched only if the request has the specified method. Support: Extended
`path`	`object`	Path specifies a HTTP request path matcher. If this field is not specified, a default prefix match on the "/" path is provided.
`queryParams`	`array`	QueryParams specifies HTTP query parameter matchers. Multiple match values are ANDed together, meaning, a request must match all the specified query parameters to select the route.

Property	Type	Description
		Support: Extended

.spec.rules[].matches[].headers

Description

Headers specifies HTTP request header matchers. Multiple match values are ANDed together, meaning, a request must match all the specified headers to select the route.

Type

`array`

.spec.rules[].matches[].headers[]

Description

HTTPHeaderMatch describes how to select a HTTP route by matching HTTP request headers.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name is the name of the HTTP Header to be matched. Name matching MUST be case-insensitive. (See https://tools.ietf.org/html/rfc7230#section-3.2 ↗). If multiple entries specify equivalent header names, only the first entry with an equivalent name MUST be considered for a match. Subsequent entries with an equivalent header name MUST be ignored. Due to the case-insensitivity of header names, "foo" and "Foo" are considered equivalent.

Property	Type	Description
		When a header is repeated in an HTTP request, it is implementation-specific behavior as to how this is represented. Generally, proxies should follow the guidance from the RFC: https://www.rfc-editor.org/rfc/rfc7230.html#section-3.2.2 regarding processing a repeated header, with special handling for "Set-Cookie".
`type`	`string`	Type specifies how to match against the value of the header. Support: Core (Exact) Support: Implementation-specific (RegularExpression) Since RegularExpression HeaderMatchType has implementation-specific conformance, implementations can support POSIX, PCRE or any other dialects of regular expressions. Please read the implementation's documentation to determine the supported dialect.
`value`	`string`	Value is the value of HTTP Header to be matched.

.spec.rules[].matches[].path

Description

Path specifies a HTTP request path matcher. If this field is not specified, a default prefix match on the "/" path is provided.

Type

`object`

Property	Type	Description
`type`	`string`	Type specifies how to match against the path Value. Support: Core (Exact, PathPrefix) Support: Implementation-specific (RegularExpression)
`value`	`string`	Value of the HTTP path to match against.

.spec.rules[].matches[].queryParams

Description

QueryParams specifies HTTP query parameter matchers. Multiple match values are ANDed together, meaning, a request must match all the specified query parameters to select the route. Support: Extended

Type

`array`

.spec.rules[].matches[].queryParams[]

Description

HTTPQueryParamMatch describes how to select a HTTP route by matching HTTP query parameters.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name is the name of the HTTP query param to be matched. This must be an exact string match. (See https://tools.ietf.org/html/rfc7230#section-2.7.3 ↗). If multiple entries specify equivalent query param names, only the first entry with an equivalent name MUST be considered for a match. Subsequent entries with an equivalent query param name MUST be ignored. If a query param is repeated in an HTTP request, the behavior is purposely left undefined, since different data planes have different capabilities. However, it is *recommended* that implementations should match against the first value of the param if the data plane supports it, as this behavior is expected in other load balancing contexts outside of the Gateway API. Users SHOULD NOT route traffic based on repeated query params to guard themselves against potential differences in the implementations.
`type`	`string`	Type specifies how to match against the value of the query parameter. Support: Extended (Exact) Support: Implementation-specific (RegularExpression) Since RegularExpression QueryParamMatchType has Implementation-specific conformance, implementations can support POSIX, PCRE or any other dialects of regular expressions. Please read the implementation's documentation to determine the supported dialect.

Property	Type	Description
`value`	`string`	Value is the value of HTTP query param to be matched.

.spec.rules[].timeouts

Description

Timeouts defines the timeouts that can be configured for an HTTP request. Support: Extended

Type

`object`

Property	Type	Description
`backendRequest`	`string`	BackendRequest specifies a timeout for an individual request from the gateway to a backend. This covers the time from when the request first starts being sent from the gateway to when the full response has been received from the backend. Setting a timeout to the zero duration (e.g. "0s") SHOULD disable the timeout completely. Implementations that cannot completely disable the timeout MUST instead interpret the zero duration as the longest possible value to which the timeout can be set. An entire client HTTP transaction with a gateway, covered by the Request timeout, may result in more than one call from the gateway to the destination backend, for example, if automatic retries are supported. The value of BackendRequest must be a Gateway API Duration string as defined by GEP-2257. When this field is unspecified, its behavior is implementation-specific;

Property	Type	Description
		when specified, the value of BackendRequest must be no more than the value of the Request timeout (since the Request timeout encompasses the BackendRequest timeout). Support: Extended
`request`	`string`	Request specifies the maximum duration for a gateway to respond to an HTTP request. If the gateway has not been able to respond before this deadline is met, the gateway MUST return a timeout error. For example, setting the `rules.timeouts.request` field to the value `10s` in an `HTTPRoute` will cause a timeout if a client request is taking longer than 10 seconds to complete. Setting a timeout to the zero duration (e.g. "0s") SHOULD disable the timeout completely. Implementations that cannot completely disable the timeout MUST instead interpret the zero duration as the longest possible value to which the timeout can be set. This timeout is intended to cover as close to the whole request-response transaction as possible although an implementation MAY choose to start the timeout after the entire request stream has been received instead of immediately after the transaction is initiated by the client. The value of Request is a Gateway API Duration string as defined by GEP-2257. When this field is unspecified, request timeout behavior is implementation-specific. Support: Extended

.status

Description

Status defines the current state of HTTPRoute.

Type

`object`

Required

`parents`

Property	Type	Description
`parents`	`array`	Parents is a list of parent resources (usually Gateways) that are associated with the route, and the status of the route with respect to each parent. When this route attaches to a parent, the controller that manages the parent must add an entry to this list when the controller first sees the route and should update the entry as appropriate when the route or gateway is modified. Note that parent references that cannot be resolved by an implementation of this API will not be added to this list. Implementations of this API can only populate Route status for the Gateways/parent resources they are responsible for. A maximum of 32 Gateways will be represented in this list. An empty list means the route has not been attached to any Gateway.

.status.parents

Description

Parents is a list of parent resources (usually Gateways) that are associated with the route, and the status of the route with respect to each parent. When this route attaches to a parent, the controller that manages the parent must add an entry to this list when the controller first sees the route and should update the entry as appropriate when the route or gateway is modified. Note that parent references that cannot be resolved by an implementation of this API will not be added to this list. Implementations of this API can only populate Route status for the Gateways/parent resources they are responsible for. A maximum of 32 Gateways will be represented in this list. An empty list means the route has not been attached to any Gateway.

Type

`array`

.status.parents[]

Description

RouteParentStatus describes the status of a route with respect to an associated Parent.

Type

`object`

Required

`conditions` `controllerName` `parentRef`

Property	Type	Description
`conditions`	`array`	Conditions describes the status of the route with respect to the Gateway. Note that the route's availability is also subject to the Gateway's own status conditions and listener status. If the Route's ParentRef specifies an existing Gateway that supports Routes of this kind AND that Gateway's controller has sufficient access, then that Gateway's controller MUST set the "Accepted" condition on the Route, to indicate whether the route has been accepted or rejected by the Gateway, and why. A Route MUST be considered "Accepted" if at least one of the Route's rules is implemented by the Gateway. There are a number of cases where the "Accepted" condition may not be set due to lack of controller visibility,

Property	Type	Description
		that includes when: • The Route refers to a nonexistent parent. • The Route is of a type that the controller does not support. • The Route is in a namespace the controller does not have access to.
`controllerName`	`string`	ControllerName is a domain/path string that indicates the name of the controller that wrote this status. This corresponds with the controllerName field on GatewayClass. Example: "example.net/gateway-controller". The format of this field is DOMAIN "/" PATH, where DOMAIN and PATH are valid Kubernetes names (https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗). Controllers MUST populate this field when writing status. Controllers should ensure that entries to status populated with their ControllerName are cleaned up when they are no longer necessary.
`parentRef`	`object`	ParentRef corresponds with a ParentRef in the spec that this RouteParentStatus struct describes the status of.

.status.parents[].conditions

Description

Conditions describes the status of the route with respect to the Gateway. Note that the route's availability is also subject to the Gateway's own status conditions and listener status. If the Route's ParentRef specifies an existing Gateway that supports Routes of this kind AND that Gateway's controller has sufficient access, then that Gateway's controller MUST set the "Accepted" condition on the Route, to indicate whether the route has been accepted or rejected by the Gateway, and why. A Route MUST be considered "Accepted" if at least one of the Route's rules is implemented by the Gateway. There are a number of cases where the "Accepted" condition may not be set due to lack of controller visibility, that includes when: * The Route refers to a nonexistent parent. * The Route is of a type that the controller does not support. * The Route is in a namespace the controller does not have access to.

Type

`array`

.status.parents[].conditions[]

Description

Condition contains details for one aspect of the current state of this API Resource.

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

Property	Type	Description
		string.
`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.parents[].parentRef

Description

ParentRef corresponds with a ParentRef in the spec that this RouteParentStatus struct describes the status of.

Type

`object`

Required

`name`

Property	Type	Description
`group`	`string`	Group is the group of the referent. When unspecified, "gateway.networking.k8s.io" is inferred. To set the core API group (such as for a "Service" kind referent), Group must be explicitly set to "" (empty string). Support: Core
`kind`	`string`	Kind is kind of the referent. There are two kinds of parent resources with "Core" support: • Gateway (Gateway conformance profile) • Service (Mesh conformance profile, ClusterIP Services only) Support for other resources is Implementation-Specific.
`name`	`string`	Name is the name of the referent. Support: Core

Property	Type	Description
`namespace`	`string`	Namespace is the namespace of the referent. When unspecified, this refers to the local namespace of the Route. Note that there are specific rules for ParentRefs which cross namespace boundaries. Cross-namespace references are only valid if they are explicitly allowed by something in the namespace they are referring to. For example: Gateway has the AllowedRoutes field, and ReferenceGrant provides a generic way to enable any other kind of cross-namespace reference. Support: Core
`port`	`integer`	Port is the network port this Route targets. It can be interpreted differently based on the type of parent resource. When the parent resource is a Gateway, this targets all listeners listening on the specified port that also support this kind of Route(and select this Route). It's not recommended to set `Port` unless the networking behaviors specified in a Route must apply to a specific port as opposed to a listener(s) whose port(s) may be changed. When both Port and SectionName are specified, the name and port of the selected listener must match both specified values. Implementations MAY choose to support other parent resources. Implementations supporting other types of parent resources MUST clearly document how/if Port is interpreted. For the purpose of status, an attachment is considered successful as long as the parent resource accepts it partially. For example, Gateway listeners can restrict which Routes can attach to them by Route kind, namespace, or hostname. If 1 of 2 Gateway listeners accept attachment

Property	Type	Description
		from the referencing Route, the Route MUST be considered successfully attached. If no Gateway listeners accept attachment from this Route, the Route MUST be considered detached from the Gateway. Support: Extended
`sectionName`	`string`	SectionName is the name of a section within the target resource. In the following resources, SectionName is interpreted as the following: • Gateway: Listener name. When both Port (experimental) and SectionName are specified, the name and port of the selected listener must match both specified values. • Service: Port name. When both Port (experimental) and SectionName are specified, the name and port of the selected listener must match both specified values. Implementations MAY choose to support attaching Routes to other resources. If that is the case, they MUST clearly document how SectionName is interpreted. When unspecified (empty string), this will reference the entire resource. For the purpose of status, an attachment is considered successful if at least one section in the parent resource accepts it. For example, Gateway listeners can restrict which Routes can attach to them by Route kind, namespace, or hostname. If 1 of 2 Gateway listeners accept attachment from the referencing Route, the Route MUST be considered successfully attached. If no Gateway listeners accept attachment from this Route, the Route MUST be considered detached from the Gateway. Support: Core

API Endpoints

The following API endpoints are available:

- `/apis/gateway.networking.k8s.io/v1/namespaces/{namespace}/httproutes`
  - `DELETE`: delete collection of HTTPRoute
  - `GET`: list objects of kind HTTPRoute
  - `POST`: create a new HTTPRoute
- `/apis/gateway.networking.k8s.io/v1/namespaces/{namespace}/httproutes/{name}`
  - `DELETE`: delete the specified HTTPRoute
  - `GET`: read the specified HTTPRoute
  - `PATCH`: partially update the specified HTTPRoute
  - `PUT`: replace the specified HTTPRoute
- `/apis/gateway.networking.k8s.io/v1/namespaces/{namespace}/httproutes/{name}/status`
  - `GET`: read status of the specified HTTPRoute
  - `PATCH`: partially update status of the specified HTTPRoute
  - `PUT`: replace status of the specified HTTPRoute

/apis/gateway.networking.k8s.io/v1/namespaces/{namespace}/httproutes

HTTP method

`DELETE`

Description

delete collection of HTTPRoute

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind HTTPRoute

HTTP responses

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

HTTP method

`POST`

Description

create a new HTTPRoute

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

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

Body parameters

Parameter	Type	Description
`body`	`HTTPRoute` schema	`application/json` formatted

HTTP responses

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

/apis/gateway.networking.k8s.io/v1/namespaces/{namespace}/httproutes/{name}

HTTP method

`DELETE`

Description

delete the specified HTTPRoute

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 HTTPRoute

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified HTTPRoute

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	`HTTPRoute` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified HTTPRoute

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`	`HTTPRoute` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`HTTPRoute` schema

HTTP code	Response body
201 - Created	`HTTPRoute` schema
401 - Unauthorized	Empty

/apis/gateway.networking.k8s.io/v1/namespaces/{namespace}/httproutes/{name}/status

HTTP method

`GET`

Description

read status of the specified HTTPRoute

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified HTTPRoute

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	`HTTPRoute` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified HTTPRoute

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

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

HTTP responses

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

Service [v1]

Description

Service is a named abstraction of software service (for example, mysql) consisting of local port (for example 3306) that the proxy listens on, and the selector that determines which pods will answer requests sent through the proxy.

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`	ServiceSpec describes the attributes that a user creates on a service.
`status`	`object`	ServiceStatus represents the current status of a service.

.spec

Description

ServiceSpec describes the attributes that a user creates on a service.

Type

`object`

Property	Type	Description
`allocateLoadBalancerNodePorts`	`boolean`	allocateLoadBalancerNodePorts defines if NodePorts will be automatically allocated for services with type LoadBalancer. Default is

Property	Type	Description
		"true". It may be set to "false" if the cluster load-balancer does not rely on NodePorts. If the caller requests specific NodePorts (by specifying a value), those requests will be respected, regardless of this field. This field may only be set for services with type LoadBalancer and will be cleared if the type is changed to any other type.
`clusterIP`	`string`	clusterIP is the IP address of the service and is usually assigned randomly. If an address is specified manually, is in-range (as per system configuration), and is not in use, it will be allocated to the service; otherwise creation of the service will fail. This field may not be changed through updates unless the type field is also being changed to ExternalName (which requires this field to be blank) or the type field is being changed from ExternalName (in which case this field may optionally be specified, as describe above). Valid values are "None", empty string (""), or a valid IP address. Setting this to "None" makes a "headless service" (no virtual IP), which is useful when direct endpoint connections are preferred and proxying is not required. Only applies to types ClusterIP, NodePort, and LoadBalancer. If this field is specified when creating a Service of type ExternalName, creation will fail. This field will be wiped when updating a Service to type ExternalName. More info: https://kubernetes.io/docs/concepts/services

Property	Type	Description
		networking/service/#virtual-ips-and-service-proxies ↗
`clusterIPs`	`array`	ClusterIPs is a list of IP addresses assigned to this service, and are usually assigned randomly. If an address is specified manually, is in-range (as per system configuration), and is not in use, it will be allocated to the service; otherwise creation of the service will fail. This field may not be changed through updates unless the type field is also being changed to ExternalName (which requires this field to be empty) or the type field is being changed from ExternalName (in which case this field may optionally be specified, as describe above). Valid values are "None", empty string (""), or a valid IP address. Setting this to "None" makes a "headless service" (no virtual IP), which is useful when direct endpoint connections are preferred and proxying is not required. Only applies to types ClusterIP NodePort, and LoadBalancer. If this field is specified when creating a Service of type ExternalName, creation will fail. This field wil be wiped when updating a Service to type ExternalName. If this field is not specified, it will be initialized from the clusterIP field. If this field is specified, clients must ensure that clusterIPs[0] and clusterIP have the same value. This field may hold a maximum of two entries (dual-stack IPs, in either order).

Property	Type	Description
		These IPs must correspond to the values of the ipFamilies field. Both clusterIPs and ipFamilies are governed by the ipFamilyPolicy field. More info: https://kubernetes.io/docs/concepts/services networking/service/#virtual-ips-and-service-proxies ↗
`externalIPs`	`array`	externalIPs is a list of IP addresses for which nodes in the cluster will also accept traffic fo this service. These IPs are not managed by Kubernetes. The user is responsible for ensuring that traffic arrives at a node with this IP. A common example is external load-balancers that are not part of the Kubernetes system.
`externalName`	`string`	externalName is the external reference that discovery mechanisms will return as an alias for this service (e.g. a DNS CNAME record). No proxying will be involved. Must be a lowercase RFC-1123 hostname (https://tools.ietf.org/html/rfc1123 ↗) and requires `type` to be "ExternalName".
`externalTrafficPolicy`	`string`	externalTrafficPolicy describes how nodes distribute service traffic they receive on one of the Service's "externally-facing" addresses (NodePorts, ExternalIPs, and LoadBalancer IPs). If set to "Local", the proxy will configure the service in a way that

Property	Type	Description
		assumes that external load balancers will take care of balancing the service traffic between nodes, and so each node will deliver traffic only to the node-local endpoints of the service, without masquerading the client source IP. (Traffic mistakenly sent to a node with no endpoints will be dropped.) The default value, "Cluster", uses the standard behavior of routing to all endpoints evenly (possibly modified by topology and other features). Note that traffic sent to an External IP or LoadBalancer IP from within the cluster will always get "Cluster" semantics, but clients sending to a NodePort from within the cluste may need to take traffic policy into account when picking a node. Possible enum values: • `"Cluster"` routes traffic to all endpoints. • `"Local"` preserves the source IP of the traffic by routing only to endpoints on the same node as the traffic was received on (dropping the traffic if there are no local endpoints).
`healthCheckNodePort`	`integer`	healthCheckNodePort specifies the healthcheck nodePort for the service. This only applies when type is set to LoadBalancer and externalTrafficPolicy is se to Local. If a value is specified, is in-range, and is not in use, it will be used. If not

Property	Type	Description
		specified, a value will be automatically allocated. External systems (e.g. load-balancers) can use this port to determine if a given node holds endpoints for this service or not. If this field is specified when creating a Service which does not need it, creation will fail. This field will be wiped when updating a Service to no longer need it (e.g. changing type). This field cannot be updated once set.
`internalTrafficPolicy`	`string`	InternalTrafficPolicy describes how nodes distribute service traffic they receive on the ClusterIP. If set to "Local", the proxy will assume that pods only want to talk to endpoints of the service on the same node as the pod, dropping the traffic if there are no local endpoints. The default value, "Cluster", uses the standard behavior of routing to all endpoints evenly (possibly modified by topology and other features). Possible enum values: • `"Cluster"` routes traffic to all endpoints. • `"Local"` routes traffic only to endpoints on the same node as the client pod (dropping the traffic if there are no local endpoints).
`ipFamilies`	`array`	IPFamilies is a list of IP families (e.g. IPv4, IPv6) assigned to this service. This field is

Property	Type	Description
		usually assigned automatically based on cluster configuration and the ipFamilyPolicy field. If this field is specified manually, the requested family is available in the cluster, and ipFamilyPolicy allows it, it will be used; otherwise creation of the service will fail. This field is conditionally mutable: it allows for adding or removing a secondary IP family, but it does not allow changing the primary IP family of the Service. Valid values are "IPv4" and "IPv6". This field only applies to Services of types ClusterIP, NodePort, and LoadBalancer, and does apply to "headless" services. This field will be wiped when updating a Service to type ExternalName. This field may hold a maximum of two entries (dual-stack families, in either order). These families must correspond to the values of the clusterIPs field, if specified. Both clusterIPs and ipFamilies are governed by the ipFamilyPolicy field.
`ipFamilyPolicy`	`string`	IPFamilyPolicy represents the dual-stack-ness requested or required by this Service. I there is no value provided, then this field will be set to SingleStack. Services can be "SingleStack" (a single IP family), "PreferDualStack" (two IP families on dual-stack configured clusters or a single IP family on single-stack clusters), or "RequireDualStack" (two IP families on dual-stack configured clusters, otherwise fail).

Property	Type	Description
		The ipFamilies and clusterIPs fields depend on the value of this field. This field will be wiped when updating a service to type ExternalName. Possible enum values: • `"PreferDualStack"` indicates that this service prefers dual-stack when the cluster is configured for dual-stack. If the cluster is not configured for dual-stack th service will be assigned a single IPFamily. If the IPFamily is not set in service.spec.ipFamilies then the service will be assigned the default IPFamily configured on the cluster • `"RequireDualStack"` indicates that thi service requires dual-stack. Using IPFamilyPolicyRequireDualStack on a single stack cluster will result in validatio errors. The IPFamilies (and their order) assigned to this service is based on service.spec.ipFamilies. If service.spec.ipFamilies was not provided then it will be assigned according to how they are configured on the cluster. If service.spec.ipFamilies has only one entry then the alternative IPFamily will be added by apiserver • `"SingleStack"` indicates that this service is required to have a single IPFamily. The IPFamily assigned is base on the default IPFamily used by the cluster or as identified by service.spec.ipFamilies field

Property	Type	Description
`loadBalancerClass`	`string`	loadBalancerClass is the class of the load balancer implementation this Service belongs to. If specified, the value of this field must be a label-style identifier, with an optional prefix, e.g. "internal-vip" or "example.com/internal-vip". Unprefixed names are reserved for end-users. This field can only be set when the Service type is 'LoadBalancer'. If not set, the default load balancer implementation is used, today this is typically done through the cloud provider integration, but should apply for any default implementation. If set, it is assumed that a load balancer implementation is watching fo Services with a matching class. Any default load balancer implementation (e.g. cloud providers) should ignore Services that set this field. This field can only be set when creating or updating a Service to type 'LoadBalancer'. Once set, it can not be changed. This field will be wiped when a service is updated to a non 'LoadBalancer' type.
`loadBalancerIP`	`string`	Only applies to Service Type: LoadBalancer. This feature depends on whether the underlying cloud-provider supports specifying the loadBalancerIP when a load balancer is created. This field will be ignored if the cloud-provider does not support the feature. Deprecated: This field was under-specified and its meaning varies across

Property	Type	Description
		implementations. Using it is non-portable and it may not support dual-stack. Users are encouraged to use implementation-specific annotations when available.
`loadBalancerSourceRanges`	`array`	If specified and supported by the platform, this will restrict traffic through the cloud-provider load-balancer will be restricted to the specified client IPs. This field will be ignored if the cloud-provider does not support the feature." More info: https://kubernetes.io/docs/tasks/access-application-cluster/create-external-load-balancer/ ↗
`ports`	`array`	The list of ports that are exposed by this service. More info: https://kubernetes.io/docs/concepts/services-networking/service/#virtual-ips-and-service-proxies ↗
`publishNotReadyAddresses`	`boolean`	publishNotReadyAddresses indicates that any agent which deals with endpoints for this Service should disregard any indications of ready/not-ready. The primary use case for setting this field is for a StatefulSet's Headless Service to propagate SRV DNS records for its Pods for the purpose of peer discovery. The Kubernetes controllers that generate Endpoints and EndpointSlice resources for Services interpret this to mean

Property	Type	Description
		that all endpoints are considered "ready" even if the Pods themselves are not. Agents which consume only Kubernetes generated endpoints through the Endpoints or EndpointSlice resources can safely assume this behavior.
`selector`	`object`	Route service traffic to pods with label keys and values matching this selector. If empty or not present, the service is assumed to have an external process managing its endpoints, which Kubernetes will not modify. Only applies to types ClusterIP, NodePort, and LoadBalancer. Ignored if type is ExternalName. More info: https://kubernetes.io/docs/concepts/services networking/service/ ↗
`sessionAffinity`	`string`	Supports "ClientIP" and "None". Used to maintain session affinity. Enable client IP based session affinity. Must be ClientIP or None. Defaults to None. More info: https://kubernetes.io/docs/concepts/services networking/service/#virtual-ips-and-service-proxies ↗ Possible enum values: • `"ClientIP"` is the Client IP based. • `"None"` - no session affinity.

Property	Type	Description
`sessionAffinityConfig`	`object`	SessionAffinityConfig represents the configurations of session affinity.
`trafficDistribution`	`string`	TrafficDistribution offers a way to express preferences for how traffic is distributed to Service endpoints. Implementations can use this field as a hint, but are not required to guarantee strict adherence. If the field is not set, the implementation will apply its default routing strategy. If set to "PreferClose", implementations should prioritize endpoints that are topologically close (e.g., same zone). This is a beta field and requires enabling ServiceTrafficDistribution feature.
`type`	`string`	type determines how the Service is exposed Defaults to ClusterIP. Valid options are ExternalName, ClusterIP, NodePort, and LoadBalancer. "ClusterIP" allocates a cluster-internal IP address for load-balancin to endpoints. Endpoints are determined by the selector or if that is not specified, by manual construction of an Endpoints object or EndpointSlice objects. If clusterIP is "None", no virtual IP is allocated and the endpoints are published as a set of endpoints rather than a virtual IP. "NodePort builds on ClusterIP and allocates a port on every node which routes to the same endpoints as the clusterIP. "LoadBalancer" builds on NodePort and creates an external

Property	Type	Description
		load-balancer (if supported in the current cloud) which routes to the same endpoints as the clusterIP. "ExternalName" aliases this service to the specified externalName. Several other fields do not apply to ExternalName services. More info: https://kubernetes.io/docs/concepts/services networking/service/#publishing-services-service-types Possible enum values: • `"ClusterIP"` means a service will only be accessible inside the cluster, via the cluster IP. • `"ExternalName"` means a service consists of only a reference to an externa name that kubedns or equivalent will return as a CNAME record, with no exposing or proxying of any pods involved. • `"LoadBalancer"` means a service will be exposed via an external load balancer (if the cloud provider supports it), in addition to 'NodePort' type. • `"NodePort"` means a service will be exposed on one port of every node, in addition to 'ClusterIP' type.

.spec.clusterIPs

Description

ClusterIPs is a list of IP addresses assigned to this service, and are usually assigned randomly. If an address is specified manually, is in-range (as per system configuration), and is not in use, it will be allocated to the service; otherwise creation of the service will fail. This field may not be changed through updates unless the type field is also being changed to ExternalName (which requires this field to be empty) or the type field is being changed from ExternalName (in which case this field may optionally be specified, as describe above). Valid values are "None", empty string (""), or a valid IP address. Setting this to "None" makes a "headless service" (no virtual IP), which is useful when direct endpoint connections are preferred and proxying is not required. Only applies to types ClusterIP, NodePort, and LoadBalancer. If this field is specified when creating a Service of type ExternalName, creation will fail. This field will be wiped when updating a Service to type ExternalName. If this field is not specified, it will be initialized from the clusterIP field. If this field is specified, clients must ensure that clusterIPs[0] and clusterIP have the same value. This field may hold a maximum of two entries (dual-stack IPs, in either order). These IPs must correspond to the values of the ipFamilies field. Both clusterIPs and ipFamilies are governed by the ipFamilyPolicy field. More info: https://kubernetes.io/docs/concepts/services-networking/service/#virtual-ips-and-service-proxies

Type

`array`

.spec.clusterIPs[]

Type

`string`

.spec.externalIPs

Description

externalIPs is a list of IP addresses for which nodes in the cluster will also accept traffic for this service. These IPs are not managed by Kubernetes. The user is responsible for ensuring that traffic arrives at a node with this IP. A common example is external load-balancers that are not part of the Kubernetes system.

Type

`array`

.spec.externalIPs[]

Type

`string`

.spec.ipFamilies

Description

IPFamilies is a list of IP families (e.g. IPv4, IPv6) assigned to this service. This field is usually assigned automatically based on cluster configuration and the ipFamilyPolicy field. If this field is specified manually, the requested family is available in the cluster, and ipFamilyPolicy allows it, it will be used; otherwise creation of the service will fail. This field is conditionally mutable: it allows for adding or removing a secondary IP family, but it does not allow changing the primary IP family of the Service. Valid values are "IPv4" and "IPv6". This field only applies to Services of types ClusterIP, NodePort, and LoadBalancer, and does apply to "headless" services. This field will be wiped when updating a Service to type ExternalName. This field may hold a maximum of two entries (dual-stack families, in either order). These families must correspond to the values of the clusterIPs field, if specified. Both clusterIPs and ipFamilies are governed by the ipFamilyPolicy field.

Type

`array`

.spec.ipFamilies[]

Type

`string`

.spec.loadBalancerSourceRanges

Description

If specified and supported by the platform, this will restrict traffic through the cloud-provider load-balancer will be restricted to the specified client IPs. This field will be ignored if the cloud-provider does not support the feature." More info: https://kubernetes.io/docs/tasks/access-application-cluster/create-external-load-balancer/

Type

`array`

.spec.loadBalancerSourceRanges[]

Type

`string`

.spec.ports

Description

The list of ports that are exposed by this service. More info: https://kubernetes.io/docs/concepts/services-networking/service/#virtual-ips-and-service-proxies

Type

`array`

.spec.ports[]

Description

ServicePort contains information on service's port.

Type

`object`

Required

`port`

Property	Type	Description
`appProtocol`	`string`	The application protocol for this port. This is used as a hint for implementations to offer richer behavior for protocols that they understand. This field follows

Property	Type	Description
		standard Kubernetes label syntax. Valid values are either: • Un-prefixed protocol names - reserved for IANA standard service names (as per RFC-6335 and https://www.iana.org/assignments/service-names ↗). • Kubernetes-defined prefixed names: • 'kubernetes.io/h2c' - HTTP/2 prior knowledge over cleartext as described in https://www.rfc-editor.org/rfc/rfc9113.html#name-starting-http-2-with-prior- ↗ • 'kubernetes.io/ws' - WebSocket over cleartext as described in https://www.rfc-editor.org/rfc/rfc6455 ↗ • 'kubernetes.io/wss' - WebSocket over TLS as described in https://www.rfc-editor.org/rfc/rfc6455 ↗ • Other protocols should use implementation-defined prefixed names such as mycompany.com/my-custom-protocol.
`name`	`string`	The name of this port within the service. This must be a DNS_LABEL. All ports within a ServiceSpec must have unique names. When considering the endpoints for a Service, this must match the 'name' field in the EndpointPort. Optional if only one ServicePort is defined on this service.
`nodePort`	`integer`	The port on each node on which this service is exposed when type is NodePort or LoadBalancer.

Property	Type	Description
		Usually assigned by the system. If a value is specified, in-range, and not in use it will be used, otherwise the operation will fail. If not specified, a port will be allocated if this Service requires one. If this field is specified when creating a Service which does not need it, creation will fail. This field will be wiped when updating a Service to no longer need it (e.g. changing type from NodePort to ClusterIP). More info: https://kubernetes.io/docs/concepts/services-networking/service/#type-nodeport
`port`	`integer`	The port that will be exposed by this service.
`protocol`	`string`	The IP protocol for this port. Supports "TCP", "UDP", and "SCTP". Default is TCP. Possible enum values: • `"SCTP"` is the SCTP protocol. • `"TCP"` is the TCP protocol. • `"UDP"` is the UDP protocol.
`targetPort`	`integer|string`	IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

.spec.selector

Description

Route service traffic to pods with label keys and values matching this selector. If empty or not present, the service is assumed to have an external process managing its endpoints, which Kubernetes will not modify. Only applies to types ClusterIP, NodePort, and LoadBalancer. Ignored if type is ExternalName. More info: https://kubernetes.io/docs/concepts/services-networking/service/

Type

`object`

.spec.sessionAffinityConfig

Description

SessionAffinityConfig represents the configurations of session affinity.

Type

`object`

Property	Type	Description
`clientIP`	`object`	ClientIPConfig represents the configurations of Client IP based session affinity.

.spec.sessionAffinityConfig.clientIP

Description

ClientIPConfig represents the configurations of Client IP based session affinity.

Type

`object`

Property	Type	Description
`timeoutSeconds`	`integer`	timeoutSeconds specifies the seconds of ClientIP type session sticky time. The value must be >0 && <=86400(for 1 day) if ServiceAffinity == "ClientIP". Default value is 10800(for 3 hours).

.status

Description

ServiceStatus represents the current status of a service.

Type

`object`

Property	Type	Description
`conditions`	`array`	Current service state
`loadBalancer`	`object`	LoadBalancerStatus represents the status of a load-balancer.

.status.conditions

Description

Current service state

Type

`array`

.status.conditions[]

Description

Condition contains details for one aspect of the current state of this API Resource.

Type

`object`

Required

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

Property	Type	Description
`lastTransitionTime`	`string`	Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.
`message`	`string`	message is a human readable message indicating details about the transition. This may be an empty string.
`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

Property	Type	Description
		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.loadBalancer

Description

LoadBalancerStatus represents the status of a load-balancer.

Type

`object`

Property	Type	Description
`ingress`	`array`	Ingress is a list containing ingress points for the load-balancer. Traffic intended for the service should be sent to these ingress points.

.status.loadBalancer.ingress

Description

Ingress is a list containing ingress points for the load-balancer. Traffic intended for the service should be sent to these ingress points.

Type

`array`

.status.loadBalancer.ingress[]

Description

LoadBalancerIngress represents the status of a load-balancer ingress point: traffic intended for the service should be sent to an ingress point.

Type

`object`

Property	Type	Description
`hostname`	`string`	Hostname is set for load-balancer ingress points that are DNS based (typically AWS load-balancers)
`ip`	`string`	IP is set for load-balancer ingress points that are IP based (typically GCE or OpenStack load-balancers)
`ipMode`	`string`	IPMode specifies how the load-balancer IP behaves, and may only be specified when the ip field is specified. Setting this to "VIP" indicates that traffic is delivered to the node with the destination set to the load-balancer's IP and port. Setting this to "Proxy" indicates that traffic is delivered to the node or pod with the destination set to the node's IP and node port or the pod's IP and port. Service implementations may use this information to adjust traffic routing.
`ports`	`array`	Ports is a list of records of service ports If used, every port defined in the service should have an entry in it

.status.loadBalancer.ingress[].ports

Description

Ports is a list of records of service ports If used, every port defined in the service should have an entry in it

Type

`array`

.status.loadBalancer.ingress[].ports[]

Description

PortStatus represents the error condition of a service port

Type

`object`

Required

`port` `protocol`

Property	Type	Description
`error`	`string`	Error is to record the problem with the service port The format of the error shall comply with the following rules: - built-in error values shall be specified in this file and those shall use CamelCase names • cloud provider specific error values must have names that comply with the format foo.example.com/CamelCase.
`port`	`integer`	Port is the port number of the service port of which status is recorded here

Property	Type	Description
`protocol`	`string`	Protocol is the protocol of the service port of which status is recorded here The supported values are: "TCP", "UDP", "SCTP" Possible enum values: • `"SCTP"` is the SCTP protocol. • `"TCP"` is the TCP protocol. • `"UDP"` is the UDP protocol.

API Endpoints

The following API endpoints are available:

- `/kubernetes/{cluster}/api/v1/namespaces/{namespace}/services`
  - `DELETE`: delete collection of Service
  - `GET`: list objects of kind Service
  - `POST`: create a new Service
- `/kubernetes/{cluster}/api/v1/namespaces/{namespace}/services/{name}`
  - `DELETE`: delete the specified Service
  - `GET`: read the specified Service
  - `PATCH`: partially update the specified Service
  - `PUT`: replace the specified Service
- `/kubernetes/{cluster}/api/v1/namespaces/{namespace}/services/{name}/status`
  - `GET`: read status of the specified Service
  - `PATCH`: partially update status of the specified Service

- `PUT`: replace status of the specified Service

/kubernetes/{cluster}/api/v1/namespaces/{namespace}/services

HTTP method

`DELETE`

Description

delete collection of Service

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind Service

HTTP responses

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

HTTP method

`POST`

Description

create a new Service

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`	`Service` schema	`application/json` formatted

HTTP responses

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

HTTP code	Response body
202 - Accepted	`Service` schema
401 - Unauthorized	Empty

/kubernetes/{cluster}/api/v1/namespaces/{namespace}/services/{name}

HTTP method

`DELETE`

Description

delete the specified Service

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 Service

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified Service

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

Parameter	Type	Description
		present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

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

HTTP method

`PUT`

Description

replace the specified Service

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

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

Body parameters

Parameter	Type	Description
`body`	`Service` schema	`application/json` formatted

HTTP responses

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

/kubernetes/{cluster}/api/v1/namespaces/{namespace}/services/{name}/status

HTTP method

`GET`

Description

read status of the specified Service

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified Service

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	`Service` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified Service

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`	`Service` schema	`application/json` formatted

HTTP responses

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

VpcEgressGateway [vpc-egress-gateways.kubeovn.io/v1]

Type

`object`

Specification

Property	Type	Description
`status`	`object`	
`spec`	`object`	

.status

Type

`object`

Required

`conditions` `phase`

Property	Type	Description
`replicas`	`integer`	
`labelSelector`	`string`	

Property	Type	Description
`conditions`	`array`	
`internalIPs`	`array`	
`externalIPs`	`array`	
`phase`	`string`	
`ready`	`boolean`	
`workload`	`object`	

.status.conditions

Type

`array`

.status.conditions[]

Type

`object`

Required

`lastTransitionTime` `lastUpdateTime` `observedGeneration` `reason` `status` `type`

Property	Type	Description
`lastTransitionTime`	`string`	
`lastUpdateTime`	`string`	
`message`	`string`	
`observedGeneration`	`integer`	

Property	Type	Description
`reason`	`string`	
`status`	`string`	
`type`	`string`	

.status.internalIPs

Type

`array`

.status.internalIPs[]

Type

`string`

.status.externalIPs

Type

`array`

.status.externalIPs[]

Type

`string`

.status.workload

Type

`object`

Property	Type	Description
`apiVersion`	`string`	
`kind`	`string`	
`name`	`string`	
`nodes`	`array`	

.status.workload.nodes

Type

`array`

.status.workload.nodes[]

Type

`string`

.spec

Type

`object`

Required

`externalSubnet`

Property	Type	Description
`replicas`	`integer`	
`prefix`	`string`	
`vpc`	`string`	

Property	Type	Description
`internalSubnet`	`string`	
`externalSubnet`	`string`	
`internalIPs`	`array`	
`externalIPs`	`array`	
`image`	`string`	
`bfd`	`object`	
`selectors`	`array`	
`policies`	`array`	
`trafficPolicy`	`string`	
`nodeSelector`	`array`	

.spec.internalIPs

Type

`array`

.spec.internalIPs[]

Type

`string`

.spec.externalIPs

Type

`array`

.spec.externalIPs[]

Type

`string`

.spec.bfd

Type

`object`

Property	Type	Description
`enabled`	`boolean`	
`minRX`	`integer`	
`minTX`	`integer`	
`multiplier`	`integer`	

.spec.selectors

Type

`array`

.spec.selectors[]

Type

`object`

Property	Type	Description
`namespaceSelector`	`object`	
`podSelector`	`object`	

.spec.selectors[].namespaceSelector

Type

`object`

Property	Type	Description
`matchLabels`	`object`	
`matchExpressions`	`array`	

.spec.selectors[].namespaceSelector.matchLabels

Type

`object`

.spec.selectors[].namespaceSelector.matchExpressions

Type

`array`

.spec.selectors[].namespaceSelector.matchExpressions[]

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	
`operator`	`string`	

Property	Type	Description
`values`	`array`	

.spec.selectors[].namespaceSelector.matchExpressions[].values

Type

`array`

.spec.selectors[].namespaceSelector.matchExpressions[].values[]

Type

`string`

.spec.selectors[].podSelector

Type

`object`

Property	Type	Description
`matchLabels`	`object`	
`matchExpressions`	`array`	

.spec.selectors[].podSelector.matchLabels

Type

`object`

.spec.selectors[].podSelector.matchExpressions

Type

`array`

.spec.selectors[].podSelector.matchExpressions[]

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	
`operator`	`string`	
`values`	`array`	

.spec.selectors[].podSelector.matchExpressions[].values

Type

`array`

.spec.selectors[].podSelector.matchExpressions[].values[]

Type

`string`

.spec.policies

Type

`array`

.spec.policies[]

Type

`object`

Property	Type	Description
`snat`	`boolean`	
`ipBlocks`	`array`	
`subnets`	`array`	

.spec.policies[].ipBlocks

Type

`array`

.spec.policies[].ipBlocks[]

Type

`string`

.spec.policies[].subnets

Type

`array`

.spec.policies[].subnets[]

Type

`string`

.spec.nodeSelector

Type

`array`

.spec.nodeSelector[]

Type

`object`

Property	Type	Description
`matchLabels`	`object`	
`matchExpressions`	`array`	
`matchFields`	`array`	

.spec.nodeSelector[].matchLabels

Type

`object`

.spec.nodeSelector[].matchExpressions

Type

`array`

.spec.nodeSelector[].matchExpressions[]

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	
`operator`	`string`	
`values`	`array`	

.spec.nodeSelector[].matchExpressions[].values

Type

`array`

.spec.nodeSelector[].matchExpressions[].values[]

Type

`string`

.spec.nodeSelector[].matchFields

Type

`array`

.spec.nodeSelector[].matchFields[]

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	
`operator`	`string`	
`values`	`array`	

.spec.nodeSelector[].matchFields[].values

Type

`array`

.spec.nodeSelector[].matchFields[].values[]

Type

`string`

API Endpoints

The following API endpoints are available:

- `/apis/kubeovn.io/v1/namespaces/{namespace}/vpcegressgateways`
  - `DELETE`: delete collection of VpcEgressGateway
  - `GET`: list objects of kind VpcEgressGateway
  - `POST`: create a new VpcEgressGateway
- `/apis/kubeovn.io/v1/namespaces/{namespace}/vpcegressgateways/{name}`
  - `DELETE`: delete the specified VpcEgressGateway
  - `GET`: read the specified VpcEgressGateway
  - `PATCH`: partially update the specified VpcEgressGateway
  - `PUT`: replace the specified VpcEgressGateway

- `/apis/kubeovn.io/v1/namespaces/{namespace}/vpcegressgateways/{name}/status`
  - `GET`: read status of the specified VpcEgressGateway
  - `PATCH`: partially update status of the specified VpcEgressGateway
  - `PUT`: replace status of the specified VpcEgressGateway

/apis/kubeovn.io/v1/namespaces/{namespace}/vpcegressgateways

HTTP method

`DELETE`

Description

delete collection of VpcEgressGateway

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind VpcEgressGateway

HTTP responses

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

HTTP method

`POST`

Description

create a new VpcEgressGateway

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`	`VpcEgressGateway` schema	`application/json` formatted

HTTP responses

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

/apis/kubeovn.io/v1/namespaces/{namespace}/vpcegressgateways/{name}

HTTP method

`DELETE`

Description

delete the specified VpcEgressGateway

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 VpcEgressGateway

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified VpcEgressGateway

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

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

HTTP responses

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

HTTP method

`PUT`

Description

replace the specified VpcEgressGateway

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

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

HTTP responses

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

/apis/kubeovn.io/v1/namespaces/{namespace}/vpcegressgateways/{name}/status

HTTP method

`GET`

Description

read status of the specified VpcEgressGateway

HTTP responses

HTTP code	Response body
200 - OK	`VpcEgressGateway` schema

HTTP code	Response body
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified VpcEgressGateway

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	`VpcEgressGateway` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified VpcEgressGateway

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`	`VpcEgressGateway` schema	`application/json` formatted

HTTP responses

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

Vpc [vpcs.kubeovn.io/v1]

Type

`object`

Specification

Property	Type	Description
`spec`	`object`	
`status`	`object`	

.spec

Type

`object`

Property	Type	Description
`defaultSubnet`	`string`	
`enableExternal`	`boolean`	
`enableBfd`	`boolean`	
`namespaces`	`array`	

Property	Type	Description
`extraExternalSubnets`	`array`	
`staticRoutes`	`array`	
`policyRoutes`	`array`	
`vpcPeerings`	`array`	
`bfdPort`	`object`	

.spec.namespaces

Type

`array`

.spec.namespaces[]

Type

`string`

.spec.extraExternalSubnets

Type

`array`

.spec.extraExternalSubnets[]

Type

`string`

.spec.staticRoutes

Type

`array`

.spec.staticRoutes[]

Type

`object`

Property	Type	Description
`policy`	`string`	
`cidr`	`string`	
`nextHopIP`	`string`	
`ecmpMode`	`string`	
`bfdId`	`string`	
`routeTable`	`string`	

.spec.policyRoutes

Type

`array`

.spec.policyRoutes[]

Type

`object`

Property	Type	Description
`priority`	`integer`	

Property	Type	Description
`action`	`string`	
`match`	`string`	
`nextHopIP`	`string`	

.spec.vpcPeerings

Type

`array`

.spec.vpcPeerings[]

Type

`object`

Property	Type	Description
`remoteVpc`	`string`	
`localConnectIP`	`string`	

.spec.bfdPort

Type

`object`

Property	Type	Description
`enabled`	`boolean`	
`ip`	`string`	

Property	Type	Description
`nodeSelector`	`object`	

.spec.bfdPort.nodeSelector

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	
`matchLabels`	`object`	

.spec.bfdPort.nodeSelector.matchExpressions

Type

`array`

.spec.bfdPort.nodeSelector.matchExpressions[]

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	
`operator`	`string`	
`values`	`array`	

.spec.bfdPort.nodeSelector.matchExpressions[].values

Type

`array`

.spec.bfdPort.nodeSelector.matchExpressions[].values[]

Type

`string`

.spec.bfdPort.nodeSelector.matchLabels

Type

`object`

.status

Type

`object`

Property	Type	Description
`conditions`	`array`	
`default`	`boolean`	
`defaultLogicalSwitch`	`string`	
`router`	`string`	
`standby`	`boolean`	
`enableExternal`	`boolean`	
`enableBfd`	`boolean`	

Property	Type	Description
`subnets`	`array`	
`extraExternalSubnets`	`array`	
`vpcPeerings`	`array`	
`tcpLoadBalancer`	`string`	
`tcpSessionLoadBalancer`	`string`	
`udpLoadBalancer`	`string`	
`udpSessionLoadBalancer`	`string`	
`sctpLoadBalancer`	`string`	
`sctpSessionLoadBalancer`	`string`	
`bfdPort`	`object`	

.status.conditions

Type

`array`

.status.conditions[]

Type

`object`

Property	Type	Description
`lastTransitionTime`	`string`	
`lastUpdateTime`	`string`	

Property	Type	Description
`message`	`string`	
`reason`	`string`	
`status`	`string`	
`type`	`string`	

.status.subnets

Type

`array`

.status.subnets[]

Type

`string`

.status.extraExternalSubnets

Type

`array`

.status.extraExternalSubnets[]

Type

`string`

.status.vpcPeerings

Type

`array`

.status.vpcPeerings[]

Type

`string`

.status.bfdPort

Type

`object`

Property	Type	Description
`ip`	`string`	
`name`	`string`	
`nodes`	`array`	

.status.bfdPort.nodes

Type

`array`

.status.bfdPort.nodes[]

Type

`string`

API Endpoints

The following API endpoints are available:

- `/apis/kubeovn.io/v1/namespaces/{namespace}/vpcs`

  - `DELETE`: delete collection of Vpc
  - `GET`: list objects of kind Vpc
  - `POST`: create a new Vpc
- `/apis/kubeovn.io/v1/namespaces/{namespace}/vpcs/{name}`
  - `DELETE`: delete the specified Vpc
  - `GET`: read the specified Vpc
  - `PATCH`: partially update the specified Vpc
  - `PUT`: replace the specified Vpc
- `/apis/kubeovn.io/v1/namespaces/{namespace}/vpcs/{name}/status`
  - `GET`: read status of the specified Vpc
  - `PATCH`: partially update status of the specified Vpc
  - `PUT`: replace status of the specified Vpc

/apis/kubeovn.io/v1/namespaces/{namespace}/vpcs

HTTP method

`DELETE`

Description

delete collection of Vpc

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind Vpc

HTTP responses

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

HTTP method

`POST`

Description

create a new Vpc

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`	`Vpc` schema	`application/json` formatted

HTTP responses

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

/apis/kubeovn.io/v1/namespaces/{namespace}/vpcs/{name}

HTTP method

`DELETE`

Description

delete the specified Vpc

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 Vpc

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified Vpc

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:

Parameter	Type	Description
		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	`Vpc` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified Vpc

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`	`Vpc` schema	`application/json` formatted

HTTP responses

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

/apis/kubeovn.io/v1/namespaces/{namespace}/vpcs/{name}/status

HTTP method

`GET`

Description

read status of the specified Vpc

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified Vpc

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

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

HTTP responses

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

HTTP method

`PUT`

Description

replace status of the specified Vpc

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

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

HTTP responses

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