Red Hat build of MicroShift 4.12

API reference

MicroShift API reference

Last Updated: 2023-10-10

Red Hat build of MicroShift 4.12 API reference

MicroShift API reference

Legal Notice

Abstract

This document provides details about the REST API functions in MicroShift.

Table of Contents

CHAPTER 1. UNDERSTANDING API TIERS

IMPORTANT

This guidance does not cover layered Red Hat build of MicroShift offerings.

Red Hat requests that application developers validate that any behavior they depend on is explicitly defined in the formal API documentation to prevent introducing dependencies on unspecified implementation-specific behavior or dependencies on bugs in a particular implementation of an API. For example, new releases of an ingress router may not be compatible with older releases if an application uses an undocumented API or relies on undefined behavior.

1.1. API TIERS

All commercially supported APIs, components, and features are associated under one of the following support levels:

API tier 1

APIs and application operating environments (AOEs) are stable within a major release. They may be deprecated within a major release, but they will not be removed until a subsequent major release.

API tier 2

APIs and AOEs are stable within a major release for a minimum of 9 months or 3 minor releases from the announcement of deprecation, whichever is longer.

API tier 3

This level applies to languages, tools, applications, and optional Operators included with Red Hat build of MicroShift through Operator Hub. Each component will specify a lifetime during which the API and AOE will be supported. Newer versions of language runtime specific components will attempt to be as API and AOE compatible from minor version to minor version as possible. Minor version to minor version compatibility is not guaranteed, however.

Components and developer tools that receive continuous updates through the Operator Hub, referred to as Operators and operands, should be considered API tier 3. Developers should use caution and understand how these components may change with each minor release. Users are encouraged to consult the compatibility guidelines documented by the component.

API tier 4

No compatibility is provided. API and AOE can change at any point. These capabilities should not be used by applications needing long-term support.

It is common practice for Operators to use custom resource definitions (CRDs) internally to accomplish a task. These objects are not meant for use by actors external to the Operator and are intended to be hidden. If any CRD is not meant for use by actors external to the Operator, the **operators.operatorframework.io/internal-objects** annotation in the Operators **ClusterServiceVersion** (CSV) should be specified to signal that the corresponding resource is internal use only and the CRD may be explicitly labeled as tier 4.

1.2. MAPPING API TIERS TO API GROUPS

For each API tier defined by Red Hat, we provide a mapping table for specific API groups where the upstream communities are committed to maintain forward compatibility. Any API group that does not specify an explicit compatibility level and is not specifically discussed below is assigned API tier 3 by default except for **v1alpha1** APIs which are assigned tier 4 by default.

1.2.1. Support for Kubernetes API groups

API groups that end with the suffix ***.k8s.io** or have the form **version.<name>** with no suffix are governed by the Kubernetes deprecation policy and follow a general mapping between API version exposed and corresponding support tier unless otherwise specified.

API version example	API tier
v1	Tier 1
v1beta1	Tier 2
v1alpha1	Tier 4

1.2.2. Support for OpenShift API groups

API groups that end with the suffix ***.openshift.io** are governed by the Red Hat build of MicroShift deprecation policy and follow a general mapping between API version exposed and corresponding compatibility level unless otherwise specified.

API version example	API tier
route.openshift.io/v1	Tier 1
security.openshift.io/v1	Tier 1 except for **RangeAllocation** (tier 4) and ***Reviews** (tier 2)

1.3. API DEPRECATION POLICY

Red Hat build of MicroShift is composed of many components sourced from many upstream communities. It is anticipated that the set of components, the associated API interfaces, and correlated features will evolve over time and might require formal deprecation in order to remove the capability.

1.3.1. Deprecating parts of the API

Red Hat build of MicroShift is a distributed system where multiple components interact with a shared state managed by the cluster control plane through a set of structured APIs. Per Kubernetes conventions, each API presented by Red Hat build of MicroShift is associated with a group identifier and each API group is independently versioned. Each API group is managed in a distinct upstream community including Kubernetes, Metal3, Multus, Operator Framework, Open Cluster Management, OpenShift itself, and more.

While each upstream community might define their own unique deprecation policy for a given API group and version, Red Hat normalizes the community specific policy to one of the compatibility levels defined prior based on our integration in and awareness of each upstream community to simplify end-user consumption and support.

The deprecation policy and schedule for APIs vary by compatibility level.

The deprecation policy covers all elements of the API including:

- REST resources, also known as API objects
- Fields of REST resources
- Annotations on REST resources, excluding version-specific qualifiers
- Enumerated or constant values

Other than the most recent API version in each group, older API versions must be supported after their announced deprecation for a duration of no less than:

API tier	Duration
Tier 1	Stable within a major release. They may be deprecated within a major release, but they will not be removed until a subsequent major release.
Tier 2	9 months or 3 releases from the announcement of deprecation, whichever is longer.
Tier 3	See the component-specific schedule.
Tier 4	None. No compatibility is guaranteed.

The following rules apply to all tier 1 APIs:

- API elements can only be removed by incrementing the version of the group.
- API objects must be able to round-trip between API versions without information loss, with the exception of whole REST resources that do not exist in some versions. In cases where equivalent fields do not exist between versions, data will be preserved in the form of annotations during conversion.
- API versions in a given group can not deprecate until a new API version at least as stable is released, except in cases where the entire API object is being removed.

1.3.2. Deprecating CLI elements

Client-facing CLI commands are not versioned in the same way as the API, but are user-facing component systems. The two major ways a user interacts with a CLI are through a command or flag, which is referred to in this context as CLI elements.

All CLI elements default to API tier 1 unless otherwise noted or the CLI depends on a lower tier API.

	Element	API tier
Generally available (GA)	Flags and commands	Tier 1
Technology Preview	Flags and commands	Tier 3
Developer Preview	Flags and commands	Tier 4

1.3.3. Deprecating an entire component

The duration and schedule for deprecating an entire component maps directly to the duration associated with the highest API tier of an API exposed by that component. For example, a component that surfaced APIs with tier 1 and 2 could not be removed until the tier 1 deprecation schedule was met.

API tier	Duration
Tier 1	Stable within a major release. They may be deprecated within a major release, but they will not be removed until a subsequent major release.
Tier 2	9 months or 3 releases from the announcement of deprecation, whichever is longer.
Tier 3	See the component-specific schedule.
Tier 4	None. No compatibility is guaranteed.

CHAPTER 2. UNDERSTANDING API COMPATIBILITY GUIDELINES

IMPORTANT

This guidance does not cover layered Red Hat build of MicroShift offerings.

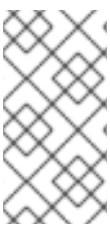
2.1. API COMPATIBILITY GUIDELINES

Red Hat recommends that application developers adopt the following principles in order to improve compatibility with Red Hat build of MicroShift:

- Use APIs and components with support tiers that match the application's need.
- Build applications using the published client libraries where possible.
- Applications are only guaranteed to run correctly if they execute in an environment that is as new as the environment it was built to execute against. An application that was built for Red Hat build of MicroShift 4.7 is not guaranteed to function properly on Red Hat build of MicroShift 4.6.
- Do not design applications that rely on configuration files provided by system packages or other components. These files can change between versions unless the upstream community is explicitly committed to preserving them. Where appropriate, depend on any Red Hat provided interface abstraction over those configuration files in order to maintain forward compatibility. Direct file system modification of configuration files is discouraged, and users are strongly encouraged to integrate with an Operator provided API where available to avoid dual-writer conflicts.
- Do not depend on API fields prefixed with **unsupported<FieldName>** or annotations that are not explicitly mentioned in product documentation.
- Do not depend on components with shorter compatibility guarantees than your application.
- Do not perform direct storage operations on the etcd server. All etcd access must be performed via the api-server or through documented backup and restore procedures.

Red Hat recommends that application developers follow the compatibility guidelines defined by Red Hat Enterprise Linux (RHEL). Red Hat build of MicroShift strongly recommends the following guidelines when building an application or hosting an application on the platform:

- Do not depend on a specific Linux kernel or Red Hat build of MicroShift version.
- Avoid reading from **proc**, **sys**, and **debug** file systems, or any other pseudo file system.
- Avoid using **ioctls** to directly interact with hardware.
- Avoid direct interaction with **cgroups** in order to not conflict with Red Hat build of MicroShift host-agents that provide the container execution environment.



NOTE

During the lifecycle of a release, Red Hat makes commercially reasonable efforts to maintain API and application operating environment (AOE) compatibility across all minor releases and z-stream releases. If necessary, Red Hat might make exceptions to this compatibility goal for critical impact security or other significant issues.

2.2. API COMPATIBILITY EXCEPTIONS

The following are exceptions to compatibility in Red Hat build of MicroShift:

Functional defaults between an upgraded cluster and a new installation
No assurances are made at this time that a new installation of a product minor release will have the same functional defaults as a version of the product that was installed with a prior minor release and upgraded to the equivalent version. For example, future versions of the product may provision cloud infrastructure with different defaults than prior minor versions. In addition, different default security choices may be made in future versions of the product than those made in past versions of the product. Past versions of the product will forward upgrade, but preserve legacy choices where appropriate specifically to maintain backwards compatibility.

Usage of API fields that have the prefix "unsupported” or undocumented annotations
Select APIs in the product expose fields with the prefix **unsupported<FieldName>**. No assurances are made at this time that usage of this field is supported across releases or within a release. Product support can request a customer to specify a value in this field when debugging specific problems, but its usage is not supported outside of that interaction. Usage of annotations on objects that are not explicitly documented are not assured support across minor releases.

API availability per product installation topology
The OpenShift distribution will continue to evolve its supported installation topology, and not all APIs in one install topology will necessarily be included in another. For example, certain topologies may restrict read/write access to particular APIs if they are in conflict with the product installation topology or not include a particular API at all if not pertinent to that topology. APIs that exist in a given topology will be supported in accordance with the compatibility tiers defined above.

2.3. API COMPATIBILITY COMMON TERMINOLOGY

2.3.1. Application Programming Interface (API)

An API is a public interface implemented by a software program that enables it to interact with other software. In Red Hat build of MicroShift, the API is served from a centralized API server and is used as the hub for all system interaction.

2.3.2. Application Operating Environment (AOE)

An AOE is the integrated environment that executes the end-user application program. The AOE is a containerized environment that provides isolation from the host operating system (OS). At a minimum, AOE allows the application to run in an isolated manner from the host OS libraries and binaries, but still share the same OS kernel as all other containers on the host. The AOE is enforced at runtime and it describes the interface between an application and its operating environment. It includes intersection points between the platform, operating system and environment, with the user application including projection of downward API, DNS, resource accounting, device access, platform workload identity, isolation among containers, isolation between containers and host OS.

The AOE does not include components that might vary by installation, such as Container Network Interface (CNI) plugin selection or extensions to the product such as admission hooks. Components that integrate with the cluster at a level below the container environment might be subjected to additional variation between versions.

2.3.3. Compatibility in a virtualized environment

Virtual environments emulate bare-metal environments such that unprivileged applications that run on

bare-metal environments will run, unmodified, in corresponding virtual environments. Virtual environments present simplified abstracted views of physical resources, so some differences might exist.

2.3.4. Compatibility in a cloud environment

Red Hat build of MicroShift might choose to offer integration points with a hosting cloud environment via cloud provider specific integrations. The compatibility of these integration points are specific to the guarantee provided by the native cloud vendor and its intersection with the Red Hat build of MicroShift compatibility window. Where Red Hat build of MicroShift provides an integration with a cloud environment natively as part of the default installation, Red Hat develops against stable cloud API endpoints to provide commercially reasonable support with forward looking compatibility that includes stable deprecation policies. Example areas of integration between the cloud provider and Red Hat build of MicroShift include, but are not limited to, dynamic volume provisioning, service load balancer integration, pod workload identity, dynamic management of compute, and infrastructure provisioned as part of initial installation.

2.3.5. Major, minor, and z-stream releases

A Red Hat major release represents a significant step in the development of a product. Minor releases appear more frequently within the scope of a major release and represent deprecation boundaries that might impact future application compatibility. A z-stream release is an update to a minor release which provides a stream of continuous fixes to an associated minor release. API and AOE compatibility is never broken in a z-stream release except when this policy is explicitly overridden in order to respond to an unforeseen security impact.

For example, in the release 4.3.2:

- 4 is the major release version
- 3 is the minor release version
- 2 is the z-stream release version

2.3.6. Extended user support (EUS)

A minor release in an Red Hat build of MicroShift major release that has an extended support window for critical bug fixes. Users are able to migrate between EUS releases by incrementally adopting minor versions between EUS releases. It is important to note that the deprecation policy is defined across minor releases and not EUS releases. As a result, an EUS user might have to respond to a deprecation when migrating to a future EUS while sequentially upgrading through each minor release.

2.3.7. Developer Preview

An optional product capability that is not officially supported by Red Hat, but is intended to provide a mechanism to explore early phase technology. By default, Developer Preview functionality is opt-in, and subject to removal at any time. Enabling a Developer Preview feature might render a cluster unsupportable dependent upon the scope of the feature.

2.3.8. Technology Preview

An optional product capability that provides early access to upcoming product innovations to test functionality and provide feedback during the development process. The feature is not fully supported, might not be functionally complete, and is not intended for production use. Usage of a Technology

Preview function requires explicit opt-in. Learn more about the Technology Preview Features Support Scope.

CHAPTER 3. NETWORK APIS

3.1. ROUTE [ROUTE.OPENSHIFT.IO/V1]

Description

A route allows developers to expose services through an HTTP(S) aware load balancing and proxy layer via a public DNS entry. The route may further specify TLS options and a certificate, or specify a public CNAME that the router should also accept for HTTP and HTTPS traffic. An administrator typically configures their router to be visible outside the cluster firewall, and may also add additional security, caching, or traffic controls on the service content. Routers usually talk directly to the service endpoints.
Once a route is created, the **host** field may not be changed. Generally, routers use the oldest route with a given host when resolving conflicts.

Routers are subject to additional customization and may support additional controls via the annotations field.

Because administrators may configure multiple routers, the route status field is used to return information to clients about the names and states of the route under each router. If a client chooses a duplicate name, for instance, the route status conditions are used to indicate the route cannot be chosen.

To enable HTTP/2 ALPN on a route it requires a custom (non-wildcard) certificate. This prevents connection coalescing by clients, notably web browsers. We do not support HTTP/2 ALPN on routes that use the default certificate because of the risk of connection re-use/coalescing. Routes that do not have their own custom certificate will not be HTTP/2 ALPN-enabled on either the frontend or the backend.

Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **spec**

3.1.1. 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_v2**	
spec	**object**	RouteSpec describes the hostname or path the route exposes, any security information, and one to four backends (services) the route points to. Requests are distributed among the backends depending on the weights assigned to each backend. When using roundrobin scheduling the portion of requests that go to each backend is the backend weight divided by the sum of all of the backend weights. When the backend has more than one endpoint the requests that end up on the backend are roundrobin distributed among the endpoints. Weights are between 0 and 256 with default 100. Weight 0 causes no requests to the backend. If all weights are zero the route will be considered to have no backends and return a standard 503 response. The **tls** field is optional and allows specific certificates or behavior for the route. Routers typically configure a default certificate on a wildcard domain to terminate routes without explicit certificates, but custom hostnames usually must choose passthrough (send traffic directly to the backend via the TLS Server-Name- Indication field) or provide a certificate.

Property	Type	Description
status	**object**	RouteStatus provides relevant info about the status of a route, including which routers acknowledge it.

3.1.1.1. .spec

Description

RouteSpec describes the hostname or path the route exposes, any security information, and one to four backends (services) the route points to. Requests are distributed among the backends depending on the weights assigned to each backend. When using roundrobin scheduling the portion of requests that go to each backend is the backend weight divided by the sum of all of the backend weights. When the backend has more than one endpoint the requests that end up on the backend are roundrobin distributed among the endpoints. Weights are between 0 and 256 with default 100. Weight 0 causes no requests to the backend. If all weights are zero the route will be considered to have no backends and return a standard 503 response.
The **tls** field is optional and allows specific certificates or behavior for the route. Routers typically configure a default certificate on a wildcard domain to terminate routes without explicit certificates, but custom hostnames usually must choose passthrough (send traffic directly to the backend via the TLS Server-Name- Indication field) or provide a certificate.

Type

object

Required

- **to**

Property	Type	Description
alternateBackends	**array**	alternateBackends allows up to 3 additional backends to be assigned to the route. Only the Service kind is allowed, and it will be defaulted to Service. Use the weight field in RouteTargetReference object to specify relative preference.
alternateBackends[]	**object**	RouteTargetReference specifies the target that resolve into endpoints. Only the 'Service' kind is allowed. Use 'weight' field to emphasize one over others.

Property	Type	Description
host	**string**	host is an alias/DNS that points to the service. Optional. If not specified a route name will typically be automatically chosen. Must follow DNS952 subdomain conventions.
path	**string**	path that the router watches for, to route traffic for to the service. Optional
port	**object**	RoutePort defines a port mapping from a router to an endpoint in the service endpoints.
subdomain	**string**	subdomain is a DNS subdomain that is requested within the ingress controller's domain (as a subdomain). If host is set this field is ignored. An ingress controller may choose to ignore this suggested name, in which case the controller will report the assigned name in the status.ingress array or refuse to admit the route. If this value is set and the server does not support this field host will be populated automatically. Otherwise host is left empty. The field may have multiple parts separated by a dot, but not all ingress controllers may honor the request. This field may not be changed after creation except by a user with the update routes/custom-host permission. Example: subdomain **frontend** automatically receives the router subdomain **apps.mycluster.com** to have a full hostname **frontend.apps.mycluster.com**.
tls	**object**	TLSConfig defines config used to secure a route and provide termination

Property	Type	Description
to	**object**	RouteTargetReference specifies the target that resolve into endpoints. Only the 'Service' kind is allowed. Use 'weight' field to emphasize one over others.
wildcardPolicy	**string**	Wildcard policy if any for the route. Currently only 'Subdomain' or 'None' is allowed.

3.1.1.2. .spec.alternateBackends

Description

alternateBackends allows up to 3 additional backends to be assigned to the route. Only the Service kind is allowed, and it will be defaulted to Service. Use the weight field in RouteTargetReference object to specify relative preference.

Type

array

3.1.1.3. .spec.alternateBackends[]

Description

RouteTargetReference specifies the target that resolve into endpoints. Only the 'Service' kind is allowed. Use 'weight' field to emphasize one over others.

Type

object

Required

- **kind**
- **name**

Property	Type	Description
kind	**string**	The kind of target that the route is referring to. Currently, only 'Service' is allowed
name	**string**	name of the service/target that is being referred to. e.g. name of the service

Property	Type	Description
weight	**integer**	weight as an integer between 0 and 256, default 100, that specifies the target's relative weight against other target reference objects. 0 suppresses requests to this backend.

3.1.1.4. .spec.port

Description

RoutePort defines a port mapping from a router to an endpoint in the service endpoints.

Type

object

Required

- **targetPort**

Property	Type	Description
targetPort	**IntOrString**	The target port on pods selected by the service this route points to. If this is a string, it will be looked up as a named port in the target endpoints port list. Required

3.1.1.5. .spec.tls

Description

TLSConfig defines config used to secure a route and provide termination

Type

object

Required

- **termination**

Property	Type	Description
caCertificate	**string**	caCertificate provides the cert authority certificate contents

Property	Type	Description
certificate	**string**	certificate provides certificate contents. This should be a single serving certificate, not a certificate chain. Do not include a CA certificate.
destinationCACertificate	**string**	destinationCACertificate provides the contents of the ca certificate of the final destination. When using reencrypt termination this file should be provided in order to have routers use it for health checks on the secure connection. If this field is not specified, the router may provide its own destination CA and perform hostname validation using the short service name (service.namespace.svc), which allows infrastructure generated certificates to automatically verify.
insecureEdgeTerminationPolicy	**string**	insecureEdgeTerminationPolicy indicates the desired behavior for insecure connections to a route. While each router may make its own decisions on which ports to expose, this is normally port 80. * Allow - traffic is sent to the server on the insecure port (default) * Disable - no traffic is allowed on the insecure port. * Redirect - clients are redirected to the secure port.
key	**string**	key provides key file contents

Property	Type	Description
termination	**string**	termination indicates termination type. * edge - TLS termination is done by the router and http is used to communicate with the backend (default) * passthrough - Traffic is sent straight to the destination without the router providing TLS termination * reencrypt - TLS termination is done by the router and https is used to communicate with the backend

3.1.1.6. .spec.to

Description

RouteTargetReference specifies the target that resolve into endpoints. Only the 'Service' kind is allowed. Use 'weight' field to emphasize one over others.

Type

object

Required

- **kind**
- **name**

Property	Type	Description
kind	**string**	The kind of target that the route is referring to. Currently, only 'Service' is allowed
name	**string**	name of the service/target that is being referred to. e.g. name of the service
weight	**integer**	weight as an integer between 0 and 256, default 100, that specifies the target's relative weight against other target reference objects. 0 suppresses requests to this backend.

3.1.1.7. .status

Description

RouteStatus provides relevant info about the status of a route, including which routers acknowledge it.

Type

object

Property	Type	Description
ingress	**array**	ingress describes the places where the route may be exposed. The list of ingress points may contain duplicate Host or RouterName values. Routes are considered live once they are **Ready**
ingress[]	**object**	RouteIngress holds information about the places where a route is exposed.

3.1.1.8. .status.ingress

Description

ingress describes the places where the route may be exposed. The list of ingress points may contain duplicate Host or RouterName values. Routes are considered live once they are **Ready**

Type

array

3.1.1.9. .status.ingress[]

Description

RouteIngress holds information about the places where a route is exposed.

Type

object

Property	Type	Description
conditions	**array**	Conditions is the state of the route, may be empty.
conditions[]	**object**	RouteIngressCondition contains details for the current condition of this route on a particular router.
host	**string**	Host is the host string under which the route is exposed; this value is required

Property	Type	Description
routerCanonicalHostname	**string**	CanonicalHostname is the external host name for the router that can be used as a CNAME for the host requested for this route. This value is optional and may not be set in all cases.
routerName	**string**	Name is a name chosen by the router to identify itself; this value is required
wildcardPolicy	**string**	Wildcard policy is the wildcard policy that was allowed where this route is exposed.

3.1.1.10. .status.ingress[].conditions

Description

Conditions is the state of the route, may be empty.

Type

array

3.1.1.11. .status.ingress[].conditions[]

Description

RouteIngressCondition contains details for the current condition of this route on a particular router.

Type

object

Required

- **type**
- **status**

Property	Type	Description
lastTransitionTime	**Time**	RFC 3339 date and time when this condition last transitioned
message	**string**	Human readable message indicating details about last transition.

Property	Type	Description
reason	**string**	(brief) reason for the condition's last transition, and is usually a machine and human readable constant
status	**string**	Status is the status of the condition. Can be True, False, Unknown.
type	**string**	Type is the type of the condition. Currently only Admitted.

3.1.2. API endpoints

The following API endpoints are available:

- **/apis/route.openshift.io/v1/routes**
  - **GET**: list or watch objects of kind Route
- **/apis/route.openshift.io/v1/watch/routes**
  - **GET**: watch individual changes to a list of Route. deprecated: use the 'watch' parameter with a list operation instead.
- **/apis/route.openshift.io/v1/namespaces/{namespace}/routes**
  - **DELETE**: delete collection of Route
  - **GET**: list or watch objects of kind Route
  - **POST**: create a Route
- **/apis/route.openshift.io/v1/watch/namespaces/{namespace}/routes**
  - **GET**: watch individual changes to a list of Route. deprecated: use the 'watch' parameter with a list operation instead.
- **/apis/route.openshift.io/v1/namespaces/{namespace}/routes/{name}**
  - **DELETE**: delete a Route
  - **GET**: read the specified Route
  - **PATCH**: partially update the specified Route
  - **PUT**: replace the specified Route
- **/apis/route.openshift.io/v1/watch/namespaces/{namespace}/routes/{name}**
  - **GET**: watch changes to an object of kind Route. deprecated: use the 'watch' parameter with a list operation instead, filtered to a single item with the 'fieldSelector' parameter.

- **/apis/route.openshift.io/v1/namespaces/{namespace}/routes/{name}/status**
  - **GET**: read status of the specified Route
  - **PATCH**: partially update status of the specified Route
  - **PUT**: replace status of the specified Route

3.1.2.1. /apis/route.openshift.io/v1/routes

Table 3.1. Global query parameters

Parameter	Type	Description
allowWatchBookmarks	**boolean**	allowWatchBookmarks requests watch events with type "BOOKMARK". Servers that do not implement bookmarks may ignore this flag and bookmarks are sent at the server's discretion. Clients should not assume bookmarks are returned at any specific interval, nor may they assume the server will send any BOOKMARK event during a session. If this is not a watch, this field is ignored.
continue	**string**	The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key". This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.
fieldSelector	**string**	A selector to restrict the list of returned objects by their fields. Defaults to everything.

Parameter	Type	Description
labelSelector	**string**	A selector to restrict the list of returned objects by their labels. Defaults to everything.
limit	**integer**	limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true. The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
pretty	**string**	If 'true', then the output is pretty printed.
resourceVersion	**string**	resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
resourceVersionMatch	**string**	resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset

Parameter	Type	Description
timeoutSeconds	**integer**	Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.
watch	**boolean**	Watch for changes to the described resources and return them as a stream of add, update, and remove notifications. Specify resourceVersion.

HTTP method

GET

Description

list or watch objects of kind Route

Table 3.2. HTTP responses

HTTP code	Reponse body
200 - OK	**RouteList** schema
401 - Unauthorized	Empty

3.1.2.2. /apis/route.openshift.io/v1/watch/routes

Table 3.3. Global query parameters

Parameter	Type	Description
allowWatchBookmarks	**boolean**	allowWatchBookmarks requests watch events with type "BOOKMARK". Servers that do not implement bookmarks may ignore this flag and bookmarks are sent at the server's discretion. Clients should not assume bookmarks are returned at any specific interval, nor may they assume the server will send any BOOKMARK event during a session. If this is not a watch, this field is ignored.

Parameter	Type	Description
continue	**string**	The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key". This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.
fieldSelector	**string**	A selector to restrict the list of returned objects by their fields. Defaults to everything.
labelSelector	**string**	A selector to restrict the list of returned objects by their labels. Defaults to everything.

Parameter	Type	Description
limit	**integer**	limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true. The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
pretty	**string**	If 'true', then the output is pretty printed.
resourceVersion	**string**	resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
resourceVersionMatch	**string**	resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
timeoutSeconds	**integer**	Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.

Parameter	Type	Description
watch	**boolean**	Watch for changes to the described resources and return them as a stream of add, update, and remove notifications. Specify resourceVersion.

HTTP method

GET

Description

watch individual changes to a list of Route. deprecated: use the 'watch' parameter with a list operation instead.

Table 3.4. HTTP responses

HTTP code	Reponse body
200 - OK	**WatchEvent** schema
401 - Unauthorized	Empty

3.1.2.3. /apis/route.openshift.io/v1/namespaces/{namespace}/routes

Table 3.5. Global path parameters

Parameter	Type	Description
namespace	**string**	object name and auth scope, such as for teams and projects

Table 3.6. Global query parameters

Parameter	Type	Description
pretty	**string**	If 'true', then the output is pretty printed.

HTTP method

DELETE

Description

delete collection of Route

Table 3.7. Query parameters

Parameter	Type	Description

Parameter	Type	Description
continue	**string**	The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key". This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.
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
fieldSelector	**string**	A selector to restrict the list of returned objects by their fields. Defaults to everything.
gracePeriodSeconds	**integer**	The duration in seconds before the object should be deleted. Value must be non-negative integer. The value zero indicates delete immediately. If this value is nil, the default grace period for the specified type will be used. Defaults to a per object value if not specified. zero means delete immediately.
labelSelector	**string**	A selector to restrict the list of returned objects by their labels. Defaults to everything.

Parameter	Type	Description
limit	**integer**	limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true. The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
orphanDependents	**boolean**	Deprecated: please use the PropagationPolicy, this field will be deprecated in 1.7. Should the dependent objects be orphaned. If true/false, the "orphan" finalizer will be added to/removed from the object's finalizers list. Either this field or PropagationPolicy may be set, but not both.
propagationPolicy	**string**	Whether and how garbage collection will be performed. Either this field or OrphanDependents may be set, but not both. The default policy is decided by the existing finalizer set in the metadata.finalizers and the resource-specific default policy. Acceptable values are: 'Orphan' - orphan the dependents; 'Background' - allow the garbage collector to delete the dependents in the background; 'Foreground' - a cascading policy that deletes all dependents in the foreground.

Parameter	Type	Description
resourceVersion	**string**	resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
resourceVersionMatch	**string**	resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
timeoutSeconds	**integer**	Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.

Table 3.8. Body parameters

Parameter	Type	Description
body	**DeleteOptions** schema	

Table 3.9. HTTP responses

HTTP code	Reponse body
200 - OK	**Status** schema
401 - Unauthorized	Empty

HTTP method

GET

Description

list or watch objects of kind Route

Table 3.10. Query parameters

Parameter	Type	Description

Parameter	Type	Description
allowWatchBookmarks	**boolean**	allowWatchBookmarks requests watch events with type "BOOKMARK". Servers that do not implement bookmarks may ignore this flag and bookmarks are sent at the server's discretion. Clients should not assume bookmarks are returned at any specific interval, nor may they assume the server will send any BOOKMARK event during a session. If this is not a watch, this field is ignored.
continue	**string**	The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key". This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.
fieldSelector	**string**	A selector to restrict the list of returned objects by their fields. Defaults to everything.
labelSelector	**string**	A selector to restrict the list of returned objects by their labels. Defaults to everything.

Parameter	Type	Description
limit	**integer**	limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true. The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
resourceVersion	**string**	resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
resourceVersionMatch	**string**	resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
timeoutSeconds	**integer**	Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.

Parameter	Type	Description
watch	**boolean**	Watch for changes to the described resources and return them as a stream of add, update, and remove notifications. Specify resourceVersion.

Table 3.11. HTTP responses

HTTP code	Reponse body
200 - OK	**RouteList** schema
401 - Unauthorized	Empty

HTTP method

POST

Description

create a Route

Table 3.12. 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
fieldManager	**string**	fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by https://golang.org/pkg/unicode/#IsPrint.

Table 3.13. Body parameters

Parameter	Type	Description
body	**Route** schema	

Table 3.14. HTTP responses

HTTP code	Reponse body
200 - OK	**Route** schema

HTTP code	Reponse body
201 - Created	**Route** schema
202 - Accepted	**Route** schema
401 - Unauthorized	Empty

3.1.2.4. /apis/route.openshift.io/v1/watch/namespaces/{namespace}/routes

Table 3.15. Global path parameters

Parameter	Type	Description
namespace	**string**	object name and auth scope, such as for teams and projects

Table 3.16. Global query parameters

Parameter	Type	Description
allowWatchBookmarks	**boolean**	allowWatchBookmarks requests watch events with type "BOOKMARK". Servers that do not implement bookmarks may ignore this flag and bookmarks are sent at the server's discretion. Clients should not assume bookmarks are returned at any specific interval, nor may they assume the server will send any BOOKMARK event during a session. If this is not a watch, this field is ignored.

Parameter	Type	Description
continue	**string**	The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key". This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.
fieldSelector	**string**	A selector to restrict the list of returned objects by their fields. Defaults to everything.
labelSelector	**string**	A selector to restrict the list of returned objects by their labels. Defaults to everything.

Parameter	Type	Description
limit	**integer**	limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true. The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
pretty	**string**	If 'true', then the output is pretty printed.
resourceVersion	**string**	resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
resourceVersionMatch	**string**	resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset

Parameter	Type	Description
timeoutSeconds	**integer**	Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.
watch	**boolean**	Watch for changes to the described resources and return them as a stream of add, update, and remove notifications. Specify resourceVersion.

HTTP method

GET

Description

watch individual changes to a list of Route. deprecated: use the 'watch' parameter with a list operation instead.

Table 3.17. HTTP responses

HTTP code	Reponse body
200 - OK	**WatchEvent** schema
401 - Unauthorized	Empty

3.1.2.5. /apis/route.openshift.io/v1/namespaces/{namespace}/routes/{name}

Table 3.18. Global path parameters

Parameter	Type	Description
name	**string**	name of the Route
namespace	**string**	object name and auth scope, such as for teams and projects

Table 3.19. Global query parameters

Parameter	Type	Description
pretty	**string**	If 'true', then the output is pretty printed.

HTTP method

DELETE

Description

delete a Route

Table 3.20. 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
gracePeriodSeconds	**integer**	The duration in seconds before the object should be deleted. Value must be non-negative integer. The value zero indicates delete immediately. If this value is nil, the default grace period for the specified type will be used. Defaults to a per object value if not specified. zero means delete immediately.
orphanDependents	**boolean**	Deprecated: please use the PropagationPolicy, this field will be deprecated in 1.7. Should the dependent objects be orphaned. If true/false, the "orphan" finalizer will be added to/removed from the object's finalizers list. Either this field or PropagationPolicy may be set, but not both.
propagationPolicy	**string**	Whether and how garbage collection will be performed. Either this field or OrphanDependents may be set, but not both. The default policy is decided by the existing finalizer set in the metadata.finalizers and the resource-specific default policy. Acceptable values are: 'Orphan' - orphan the dependents; 'Background' - allow the garbage collector to delete the dependents in the background; 'Foreground' - a cascading policy that deletes all dependents in the foreground.

Table 3.21. Body parameters

Parameter	Type	Description
body	**DeleteOptions** schema	

Table 3.22. HTTP responses

HTTP code	Reponse body
200 - OK	**Status** schema
202 - Accepted	**Status** schema

HTTP code	Reponse body
401 - Unauthorized	Empty

HTTP method

GET

Description

read the specified Route

Table 3.23. HTTP responses

HTTP code	Reponse body
200 - OK	**Route** schema
401 - Unauthorized	Empty

HTTP method

PATCH

Description

partially update the specified Route

Table 3.24. 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
fieldManager	**string**	fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by https://golang.org/pkg/unicode/#IsPrint. This field is required for apply requests (application/apply-patch) but optional for non-apply patch types (JsonPatch, MergePatch, StrategicMergePatch).
force	**boolean**	Force is going to "force" Apply requests. It means user will re-acquire conflicting fields owned by other people. Force flag must be unset for non-apply patch requests.

Table 3.25. Body parameters

Parameter	Type	Description
body	**Patch** schema	

Table 3.26. HTTP responses

HTTP code	Reponse body
200 - OK	**Route** schema
201 - Created	**Route** schema
401 - Unauthorized	Empty

HTTP method

PUT

Description

replace the specified Route

Table 3.27. 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
fieldManager	**string**	fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by https://golang.org/pkg/unicode/#IsPrint.

Table 3.28. Body parameters

Parameter	Type	Description
body	**Route** schema	

Table 3.29. HTTP responses

HTTP code	Reponse body
200 - OK	**Route** schema

HTTP code	Reponse body
201 - Created	**Route** schema
401 - Unauthorized	Empty

3.1.2.6. /apis/route.openshift.io/v1/watch/namespaces/{namespace}/routes/{name}

Table 3.30. Global path parameters

Parameter	Type	Description
name	**string**	name of the Route
namespace	**string**	object name and auth scope, such as for teams and projects

Table 3.31. Global query parameters

Parameter	Type	Description
allowWatchBookmarks	**boolean**	allowWatchBookmarks requests watch events with type "BOOKMARK". Servers that do not implement bookmarks may ignore this flag and bookmarks are sent at the server's discretion. Clients should not assume bookmarks are returned at any specific interval, nor may they assume the server will send any BOOKMARK event during a session. If this is not a watch, this field is ignored.

Parameter	Type	Description
continue	**string**	The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key". This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.
fieldSelector	**string**	A selector to restrict the list of returned objects by their fields. Defaults to everything.
labelSelector	**string**	A selector to restrict the list of returned objects by their labels. Defaults to everything.

Parameter	Type	Description
limit	**integer**	limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true. The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
pretty	**string**	If 'true', then the output is pretty printed.
resourceVersion	**string**	resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
resourceVersionMatch	**string**	resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset

Parameter	Type	Description
timeoutSeconds	**integer**	Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.
watch	**boolean**	Watch for changes to the described resources and return them as a stream of add, update, and remove notifications. Specify resourceVersion.

HTTP method

GET

Description

watch changes to an object of kind Route. deprecated: use the 'watch' parameter with a list operation instead, filtered to a single item with the 'fieldSelector' parameter.

Table 3.32. HTTP responses

HTTP code	Reponse body
200 - OK	**WatchEvent** schema
401 - Unauthorized	Empty

3.1.2.7. /apis/route.openshift.io/v1/namespaces/{namespace}/routes/{name}/status

Table 3.33. Global path parameters

Parameter	Type	Description
name	**string**	name of the Route
namespace	**string**	object name and auth scope, such as for teams and projects

Table 3.34. Global query parameters

Parameter	Type	Description
pretty	**string**	If 'true', then the output is pretty printed.

HTTP method

GET

Description

read status of the specified Route

Table 3.35. HTTP responses

HTTP code	Reponse body
200 - OK	**Route** schema
401 - Unauthorized	Empty

HTTP method

PATCH

Description

partially update status of the specified Route

Table 3.36. 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
fieldManager	**string**	fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by https://golang.org/pkg/unicode/#IsPrint. This field is required for apply requests (application/apply-patch) but optional for non-apply patch types (JsonPatch, MergePatch, StrategicMergePatch).
force	**boolean**	Force is going to "force" Apply requests. It means user will re-acquire conflicting fields owned by other people. Force flag must be unset for non-apply patch requests.

Table 3.37. Body parameters

Parameter	Type	Description
body	**Patch** schema	

Table 3.38. HTTP responses

HTTP code	Reponse body
200 - OK	**Route** schema

HTTP code	Reponse body
201 - Created	**Route** schema
401 - Unauthorized	Empty

HTTP method

PUT

Description

replace status of the specified Route

Table 3.39. 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
fieldManager	**string**	fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by https://golang.org/pkg/unicode/#IsPrint.

Table 3.40. Body parameters

Parameter	Type	Description
body	**Route** schema	

Table 3.41. HTTP responses

HTTP code	Reponse body
200 - OK	**Route** schema
201 - Created	**Route** schema
401 - Unauthorized	Empty

CHAPTER 4. SECURITY APIS

4.1. SECURITYCONTEXTCONSTRAINTS [SECURITY.OPENSHIFT.IO/V1]

Description

SecurityContextConstraints (SCC) governs the ability to make requests that affect the SecurityContext that applies to a container. Use the security.openshift.io group to manage SecurityContextConstraints. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **allowHostDirVolumePlugin**
- **allowHostIPC**
- **allowHostNetwork**
- **allowHostPID**
- **allowHostPorts**
- **allowPrivilegedContainer**
- **readOnlyRootFilesystem**

4.1.1. Specification

Property	Type	Description
allowHostDirVolumePlugin	**boolean**	AllowHostDirVolumePlugin determines if the policy allow containers to use the HostDir volume plugin
allowHostIPC	**boolean**	AllowHostIPC determines if the policy allows host ipc in the containers.
allowHostNetwork	**boolean**	AllowHostNetwork determines if the policy allows the use of HostNetwork in the pod spec.
allowHostPID	**boolean**	AllowHostPID determines if the policy allows host pid in the containers.

Property	Type	Description
allowHostPorts	**boolean**	AllowHostPorts determines if the policy allows host ports in the containers.
allowPrivilegeEscalation	``	AllowPrivilegeEscalation determines if a pod can request to allow privilege escalation. If unspecified, defaults to true.
allowPrivilegedContainer	**boolean**	AllowPrivilegedContainer determines if a container can request to be run as privileged.
allowedCapabilities	``	AllowedCapabilities is a list of capabilities that can be requested to add to the container. Capabilities in this field maybe added at the pod author's discretion. You must not list a capability in both AllowedCapabilities and RequiredDropCapabilities. To allow all capabilities you may use '*'.
allowedFlexVolumes	``	AllowedFlexVolumes is a whitelist of allowed Flexvolumes. Empty or nil indicates that all Flexvolumes may be used. This parameter is effective only when the usage of the Flexvolumes is allowed in the "Volumes" field.
allowedUnsafeSysctls	``	AllowedUnsafeSysctls is a list of explicitly allowed unsafe sysctls, defaults to none. Each entry is either a plain sysctl name or ends in "**" in which case it is considered as a prefix of allowed sysctls. Single * means all unsafe sysctls are allowed. Kubelet has to whitelist all allowed unsafe sysctls explicitly to avoid rejection. Examples: e.g. "foo/**" allows "foo/bar", "foo/baz", etc. e.g. "foo.*" allows "foo.bar", "foo.baz", etc.

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
defaultAddCapabilities	``	DefaultAddCapabilities is the default set of capabilities that will be added to the container unless the pod spec specifically drops the capability. You may not list a capabiility in both DefaultAddCapabilities and RequiredDropCapabilities.
defaultAllowPrivilegeEscalation	``	DefaultAllowPrivilegeEscalation controls the default setting for whether a process can gain more privileges than its parent process.
forbiddenSysctls	``	ForbiddenSysctls is a list of explicitly forbidden sysctls, defaults to none. Each entry is either a plain sysctl name or ends in "**" in which case it is considered as a prefix of forbidden sysctls. Single * means all sysctls are forbidden. Examples: e.g. "foo/**" forbids "foo/bar", "foo/baz", etc. e.g. "foo.*" forbids "foo.bar", "foo.baz", etc.
fsGroup	``	FSGroup is the strategy that will dictate what fs group is used by the SecurityContext.
groups	``	The groups that have permission to use this security context constraints

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**	Standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
priority	``	Priority influences the sort order of SCCs when evaluating which SCCs to try first for a given pod request based on access in the Users and Groups fields. The higher the int, the higher priority. An unset value is considered a 0 priority. If scores for multiple SCCs are equal they will be sorted from most restrictive to least restrictive. If both priorities and restrictions are equal the SCCs will be sorted by name.
readOnlyRootFilesystem	**boolean**	ReadOnlyRootFilesystem when set to true will force containers to run with a read only root file system. If the container specifically requests to run with a non-read only root file system the SCC should deny the pod. If set to false the container may run with a read only root file system if it wishes but it will not be forced to.
requiredDropCapabilities	``	RequiredDropCapabilities are the capabilities that will be dropped from the container. These are required to be dropped and cannot be added.

Property	Type	Description
runAsUser	``	RunAsUser is the strategy that will dictate what RunAsUser is used in the SecurityContext.
seLinuxContext	``	SELinuxContext is the strategy that will dictate what labels will be set in the SecurityContext.
seccompProfiles	``	SeccompProfiles lists the allowed profiles that may be set for the pod or container's seccomp annotations. An unset (nil) or empty value means that no profiles may be specifid by the pod or container. The wildcard '*' may be used to allow all profiles. When used to generate a value for a pod the first non-wildcard profile will be used as the default.
supplementalGroups	``	SupplementalGroups is the strategy that will dictate what supplemental groups are used by the SecurityContext.
users	``	The users who have permissions to use this security context constraints
volumes	``	Volumes is a white list of allowed volume plugins. FSType corresponds directly with the field names of a VolumeSource (azureFile, configMap, emptyDir). To allow all volumes you may use "*". To allow no volumes, set to ["none"].

4.1.2. API endpoints

The following API endpoints are available:

- **/apis/security.openshift.io/v1/securitycontextconstraints**
  - **DELETE**: delete collection of SecurityContextConstraints
  - **GET**: list objects of kind SecurityContextConstraints
  - **POST**: create SecurityContextConstraints

- **/apis/security.openshift.io/v1/watch/securitycontextconstraints**
  - **GET**: watch individual changes to a list of SecurityContextConstraints. deprecated: use the 'watch' parameter with a list operation instead.
- **/apis/security.openshift.io/v1/securitycontextconstraints/{name}**
  - **DELETE**: delete SecurityContextConstraints
  - **GET**: read the specified SecurityContextConstraints
  - **PATCH**: partially update the specified SecurityContextConstraints
  - **PUT**: replace the specified SecurityContextConstraints
- **/apis/security.openshift.io/v1/watch/securitycontextconstraints/{name}**
  - **GET**: watch changes to an object of kind SecurityContextConstraints. deprecated: use the 'watch' parameter with a list operation instead, filtered to a single item with the 'fieldSelector' parameter.

4.1.2.1. /apis/security.openshift.io/v1/securitycontextconstraints

Table 4.1. Global query parameters

Parameter	Type	Description
pretty	**string**	If 'true', then the output is pretty printed.

HTTP method

DELETE

Description

delete collection of SecurityContextConstraints

Table 4.2. Query parameters

Parameter	Type	Description
allowWatchBookmarks	**boolean**	allowWatchBookmarks requests watch events with type "BOOKMARK". Servers that do not implement bookmarks may ignore this flag and bookmarks are sent at the server's discretion. Clients should not assume bookmarks are returned at any specific interval, nor may they assume the server will send any BOOKMARK event during a session. If this is not a watch, this field is ignored.

Parameter	Type	Description
continue	**string**	The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key". This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.
fieldSelector	**string**	A selector to restrict the list of returned objects by their fields. Defaults to everything.
labelSelector	**string**	A selector to restrict the list of returned objects by their labels. Defaults to everything.

Parameter	Type	Description
limit	**integer**	limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true. The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
resourceVersion	**string**	resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
resourceVersionMatch	**string**	resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
timeoutSeconds	**integer**	Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.

Parameter	Type	Description
watch	**boolean**	Watch for changes to the described resources and return them as a stream of add, update, and remove notifications. Specify resourceVersion.

Table 4.3. HTTP responses

HTTP code	Reponse body
200 - OK	**Status** schema
401 - Unauthorized	Empty

HTTP method

GET

Description

list objects of kind SecurityContextConstraints

Table 4.4. Query parameters

Parameter	Type	Description
allowWatchBookmarks	**boolean**	allowWatchBookmarks requests watch events with type "BOOKMARK". Servers that do not implement bookmarks may ignore this flag and bookmarks are sent at the server's discretion. Clients should not assume bookmarks are returned at any specific interval, nor may they assume the server will send any BOOKMARK event during a session. If this is not a watch, this field is ignored.

Parameter	Type	Description
continue	**string**	The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key". This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.
fieldSelector	**string**	A selector to restrict the list of returned objects by their fields. Defaults to everything.
labelSelector	**string**	A selector to restrict the list of returned objects by their labels. Defaults to everything.

Parameter	Type	Description
limit	**integer**	limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true. The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
resourceVersion	**string**	resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
resourceVersionMatch	**string**	resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
timeoutSeconds	**integer**	Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.

Parameter	Type	Description
watch	**boolean**	Watch for changes to the described resources and return them as a stream of add, update, and remove notifications. Specify resourceVersion.

Table 4.5. HTTP responses

HTTP code	Reponse body
200 - OK	**SecurityContextConstraintsList** schema
401 - Unauthorized	Empty

HTTP method

POST

Description

create SecurityContextConstraints

Table 4.6. 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
fieldManager	**string**	fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by https://golang.org/pkg/unicode/#IsPrint.

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, provided that the `ServerSideFieldValidation` feature gate is also enabled. 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 and is the default behavior when the `ServerSideFieldValidation` feature gate is disabled. - 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 when the `ServerSideFieldValidation` feature gate is enabled. - 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.

Table 4.7. Body parameters

Parameter	Type	Description
body	**SecurityContextConstraints** schema	

Table 4.8. HTTP responses

HTTP code	Reponse body
200 - OK	**SecurityContextConstraints** schema
201 - Created	**SecurityContextConstraints** schema
202 - Accepted	**SecurityContextConstraints** schema
401 - Unauthorized	Empty

4.1.2.2. /apis/security.openshift.io/v1/watch/securitycontextconstraints

Table 4.9. Global query parameters

Parameter	Type	Description
allowWatchBookmarks	**boolean**	allowWatchBookmarks requests watch events with type "BOOKMARK". Servers that do not implement bookmarks may ignore this flag and bookmarks are sent at the server's discretion. Clients should not assume bookmarks are returned at any specific interval, nor may they assume the server will send any BOOKMARK event during a session. If this is not a watch, this field is ignored.
continue	**string**	The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key". This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.
fieldSelector	**string**	A selector to restrict the list of returned objects by their fields. Defaults to everything.
labelSelector	**string**	A selector to restrict the list of returned objects by their labels. Defaults to everything.

Parameter	Type	Description
limit	**integer**	limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true. The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
pretty	**string**	If 'true', then the output is pretty printed.
resourceVersion	**string**	resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
resourceVersionMatch	**string**	resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
timeoutSeconds	**integer**	Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.

Parameter	Type	Description
watch	**boolean**	Watch for changes to the described resources and return them as a stream of add, update, and remove notifications. Specify resourceVersion.

HTTP method

GET

Description

watch individual changes to a list of SecurityContextConstraints. deprecated: use the 'watch' parameter with a list operation instead.

Table 4.10. HTTP responses

HTTP code	Reponse body
200 - OK	**WatchEvent** schema
401 - Unauthorized	Empty

4.1.2.3. /apis/security.openshift.io/v1/securitycontextconstraints/{name}

Table 4.11. Global path parameters

Parameter	Type	Description
name	**string**	name of the SecurityContextConstraints

Table 4.12. Global query parameters

Parameter	Type	Description
pretty	**string**	If 'true', then the output is pretty printed.

HTTP method

DELETE

Description

delete SecurityContextConstraints

Table 4.13. Query parameters

Parameter	Type	Description

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
gracePeriodSeconds	**integer**	The duration in seconds before the object should be deleted. Value must be non-negative integer. The value zero indicates delete immediately. If this value is nil, the default grace period for the specified type will be used. Defaults to a per object value if not specified. zero means delete immediately.
orphanDependents	**boolean**	Deprecated: please use the PropagationPolicy, this field will be deprecated in 1.7. Should the dependent objects be orphaned. If true/false, the "orphan" finalizer will be added to/removed from the object's finalizers list. Either this field or PropagationPolicy may be set, but not both.
propagationPolicy	**string**	Whether and how garbage collection will be performed. Either this field or OrphanDependents may be set, but not both. The default policy is decided by the existing finalizer set in the metadata.finalizers and the resource-specific default policy. Acceptable values are: 'Orphan' - orphan the dependents; 'Background' - allow the garbage collector to delete the dependents in the background; 'Foreground' - a cascading policy that deletes all dependents in the foreground.

Table 4.14. Body parameters

Parameter	Type	Description
body	**DeleteOptions** schema	

Table 4.15. HTTP responses

HTTP code	Reponse body
200 - OK	**Status** schema
202 - Accepted	**Status** schema

HTTP code	Reponse body
401 - Unauthorized	Empty

HTTP method

GET

Description

read the specified SecurityContextConstraints

Table 4.16. Query parameters

Parameter	Type	Description
resourceVersion	**string**	resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset

Table 4.17. HTTP responses

HTTP code	Reponse body
200 - OK	**SecurityContextConstraints** schema
401 - Unauthorized	Empty

HTTP method

PATCH

Description

partially update the specified SecurityContextConstraints

Table 4.18. 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
fieldManager	**string**	fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by https://golang.org/pkg/unicode/#IsPrint.
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields, provided that the `ServerSideFieldValidation` feature gate is also enabled. 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 and is the default behavior when the `ServerSideFieldValidation` feature gate is disabled. - 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 when the `ServerSideFieldValidation` feature gate is enabled. - 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.

Table 4.19. Body parameters

Parameter	Type	Description
body	**Patch** schema	

Table 4.20. HTTP responses

HTTP code	Reponse body
200 - OK	**SecurityContextConstraints** schema
401 - Unauthorized	Empty

HTTP method

PUT

Description

replace the specified SecurityContextConstraints

Table 4.21. 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
fieldManager	**string**	fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by https://golang.org/pkg/unicode/#IsPrint.
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields, provided that the `ServerSideFieldValidation` feature gate is also enabled. 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 and is the default behavior when the `ServerSideFieldValidation` feature gate is disabled. - 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 when the `ServerSideFieldValidation` feature gate is enabled. - 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.

Table 4.22. Body parameters

Parameter	Type	Description
body	**SecurityContextConstraints** schema	

Table 4.23. HTTP responses

HTTP code	Reponse body
200 - OK	**SecurityContextConstraints** schema
201 - Created	**SecurityContextConstraints** schema
401 - Unauthorized	Empty

4.1.2.4. /apis/security.openshift.io/v1/watch/securitycontextconstraints/{name}

Table 4.24. Global path parameters

Parameter	Type	Description
name	**string**	name of the SecurityContextConstraints

Table 4.25. Global query parameters

Parameter	Type	Description
allowWatchBookmarks	**boolean**	allowWatchBookmarks requests watch events with type "BOOKMARK". Servers that do not implement bookmarks may ignore this flag and bookmarks are sent at the server's discretion. Clients should not assume bookmarks are returned at any specific interval, nor may they assume the server will send any BOOKMARK event during a session. If this is not a watch, this field is ignored.

Parameter	Type	Description
continue	**string**	The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key". This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.
fieldSelector	**string**	A selector to restrict the list of returned objects by their fields. Defaults to everything.
labelSelector	**string**	A selector to restrict the list of returned objects by their labels. Defaults to everything.

Parameter	Type	Description
limit	**integer**	limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true. The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
pretty	**string**	If 'true', then the output is pretty printed.
resourceVersion	**string**	resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
resourceVersionMatch	**string**	resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details. Defaults to unset
timeoutSeconds	**integer**	Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.

Parameter	Type	Description
watch	**boolean**	Watch for changes to the described resources and return them as a stream of add, update, and remove notifications. Specify resourceVersion.

HTTP method

GET

Description

watch changes to an object of kind SecurityContextConstraints. deprecated: use the 'watch' parameter with a list operation instead, filtered to a single item with the 'fieldSelector' parameter.

Table 4.26. HTTP responses

HTTP code	Reponse body
200 - OK	**WatchEvent** schema
401 - Unauthorized	Empty