OpenShift Container Platform 4.17

Image APIs

Reference guide for image APIs

Last Updated: 2024-11-05

OpenShift Container Platform 4.17 Image APIs

Reference guide for image APIs

Legal Notice

Abstract

This document describes the OpenShift Container Platform image API objects and their detailed specifications.

Table of Contents

CHAPTER 1. IMAGE APIS

1.1. IMAGE [IMAGE.OPENSHIFT.IO/V1]

Description

Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

1.2. IMAGESIGNATURE [IMAGE.OPENSHIFT.IO/V1]

Description

ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

1.3. IMAGESTREAMIMAGE [IMAGE.OPENSHIFT.IO/V1]

Description

ImageStreamImage represents an Image that is retrieved by image name from an ImageStream. User interfaces and regular users can use this resource to access the metadata details of a tagged image in the image stream history for viewing, since Image resources are not directly accessible to end users. A not found error will be returned if no such image is referenced by a tag within the ImageStream. Images are created when spec tags are set on an image stream that represent an image in an external registry, when pushing to the integrated registry, or when tagging an existing image from one image stream to another. The name of an image stream image is in the form "<STREAM>@<DIGEST>", where the digest is the content addressible identifier for the image (sha256:xxxxx...). You can use ImageStreamImages as the from.kind of an image stream spec tag to reference an image exactly. The only operations supported on the imagestreamimage endpoint are retrieving the image.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

1.4. IMAGESTREAMIMPORT [IMAGE.OPENSHIFT.IO/V1]

Description

The image stream import resource provides an easy way for a user to find and import container images from other container image registries into the server. Individual images or an entire image repository may be imported, and users may choose to see the results of the import prior to tagging the resulting images into the specified image stream.
This API is intended for end-user tools that need to see the metadata of the image prior to import (for instance, to generate an application from it). Clients that know the desired image can continue to create spec.tags directly into their image streams.

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

Type

object

1.5. IMAGESTREAMLAYERS [IMAGE.OPENSHIFT.IO/V1]

Description

ImageStreamLayers describes information about the layers referenced by images in this image stream.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

1.6. IMAGESTREAMMAPPING [IMAGE.OPENSHIFT.IO/V1]

Description

ImageStreamMapping represents a mapping from a single image stream tag to a container image as well as the reference to the container image stream the image came from. This resource is used by privileged integrators to create an image resource and to associate it with an image stream in the status tags field. Creating an ImageStreamMapping will allow any user who can view the image stream to tag or pull that image, so only create mappings where the user has proven they have access to the image contents directly. The only operation supported for this resource is create and the metadata name and namespace should be set to the image stream containing the tag that should be updated.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

1.7. IMAGESTREAM [IMAGE.OPENSHIFT.IO/V1]

Description

An ImageStream stores a mapping of tags to images, metadata overrides that are applied when images are tagged in a stream, and an optional reference to a container image repository on a registry. Users typically update the spec.tags field to point to external images which are imported

from container registries using credentials in your namespace with the pull secret type, or to existing image stream tags and images which are immediately accessible for tagging or pulling. The history of images applied to a tag is visible in the status.tags field and any user who can view an image stream is allowed to tag that image into their own image streams. Access to pull images from the integrated registry is granted by having the "get imagestreams/layers" permission on a given image stream. Users may remove a tag by deleting the imagestreamtag resource, which causes both spec and status for that tag to be removed. Image stream history is retained until an administrator runs the prune operation, which removes references that are no longer in use. To preserve a historical image, ensure there is a tag in spec pointing to that image by its digest.

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

Type

object

1.8. IMAGESTREAMTAG [IMAGE.OPENSHIFT.IO/V1]

Description

ImageStreamTag represents an Image that is retrieved by tag name from an ImageStream. Use this resource to interact with the tags and images in an image stream by tag, or to see the image details for a particular tag. The image associated with this resource is the most recently successfully tagged, imported, or pushed image (as described in the image stream status.tags.items list for this tag). If an import is in progress or has failed the previous image will be shown. Deleting an image stream tag clears both the status and spec fields of an image stream. If no image can be retrieved for a given tag, a not found error will be returned.

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

Type

object

1.9. IMAGETAG [IMAGE.OPENSHIFT.IO/V1]

Description

ImageTag represents a single tag within an image stream and includes the spec, the status history, and the currently referenced image (if any) of the provided tag. This type replaces the ImageStreamTag by providing a full view of the tag. ImageTags are returned for every spec or status tag present on the image stream. If no tag exists in either form a not found error will be returned by the API. A create operation will succeed if no spec tag has already been defined and the spec field is set. Delete will remove both spec and status elements from the image stream.

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

Type

object

1.10. SECRETLIST [IMAGE.OPENSHIFT.IO/V1]

Description

SecretList is a list of Secret.

Type

object

CHAPTER 2. IMAGE [IMAGE.OPENSHIFT.IO/V1]

Description

Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

2.1. 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
dockerImageConfig	**string**	DockerImageConfig is a JSON blob that the runtime uses to set up the container. This is a part of manifest schema v2. Will not be set when the image represents a manifest list.
dockerImageLayers	**array**	DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.
dockerImageLayers[]	**object**	ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.
dockerImageManifest	**string**	DockerImageManifest is the raw JSON of the manifest

Property	Type	Description
dockerImageManifestMediaType	**string**	DockerImageManifestMediaType specifies the mediaType of manifest. This is a part of manifest schema v2.
dockerImageManifests	**array**	DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.
dockerImageManifests[]	**object**	ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.
dockerImageMetadata	**RawExtension**	DockerImageMetadata contains metadata about this image
dockerImageMetadataVersion	**string**	DockerImageMetadataVersion conveys the version of the object, which if empty defaults to "1.0"
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image.
dockerImageSignatures	**array (string)**	DockerImageSignatures provides the signatures as opaque blobs. This is a part of manifest schema v1.
kind	**string**	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

Property	Type	Description
metadata	**ObjectMeta_v2**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signatures	**array**	Signatures holds all signatures of the image.
signatures[]	**object**	ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

2.1.1. .dockerImageLayers

Description

DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.

Type

array

2.1.2. .dockerImageLayers[]

Description

ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.

Type

object

Required

- **name**

- **size**
- **mediaType**

Property	Type	Description
mediaType	**string**	MediaType of the referenced object.
name	**string**	Name of the layer as defined by the underlying store.
size	**integer**	Size of the layer in bytes as defined by the underlying store.

2.1.3. .dockerImageManifests

Description

DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.

Type

array

2.1.4. .dockerImageManifests[]

Description

ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Type

object

Required

- **digest**
- **mediaType**
- **manifestSize**
- **architecture**
- **os**

Property	Type	Description
architecture	**string**	Architecture specifies the supported CPU architecture, for example **amd64** or **ppc64le**.

Property	Type	Description
digest	**string**	Digest is the unique identifier for the manifest. It refers to an Image object.
manifestSize	**integer**	ManifestSize represents the size of the raw object contents, in bytes.
mediaType	**string**	MediaType defines the type of the manifest, possible values are application/vnd.oci.image.manifest.v1+json, application/vnd.docker.distribution.manifest.v2+json or application/vnd.docker.distribution.manifest.v1+json.
os	**string**	OS specifies the operating system, for example **linux**.
variant	**string**	Variant is an optional field repreenting a variant of the CPU, for example v6 to specify a particular CPU variant of the ARM CPU.

2.1.5. .signatures

Description

Signatures holds all signatures of the image.

Type

array

2.1.6. .signatures[]

Description

ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose.

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

Type

object

Required

- **type**
- **content**

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
conditions	**array**	Conditions represent the latest available observations of a signature's current state.
conditions[]	**object**	SignatureCondition describes an image signature condition of particular kind at particular probe time.
content	**string**	Required: An opaque binary string which is an image's signature.
created	**Time**	If specified, it is the time of signature's creation.
imageIdentity	**string**	A human readable string representing image's identity. It could be a product name and version, or an image pull spec (e.g. "registry.access.redhat.com/rhel7/rhel:7.2").
issuedBy	**object**	SignatureIssuer holds information about an issuer of signing certificate or key.
issuedTo	**object**	SignatureSubject holds information about a person or entity who created the signature.

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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signedClaims	**object (string)**	Contains claims from the signature.
type	**string**	Required: Describes a type of stored blob.

2.1.7. .signatures[].conditions

Description

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

Type

array

2.1.8. .signatures[].conditions[]

Description

SignatureCondition describes an image signature condition of particular kind at particular probe time.

Type

object

Required

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

Property	Type	Description
lastProbeTime	**Time**	Last time the condition was checked.
lastTransitionTime	**Time**	Last time the condition transit from one status to another.
message	**string**	Human readable message indicating details about last transition.
reason	**string**	(brief) reason for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of signature condition, Complete or Failed.

2.1.9. .signatures[].issuedBy

Description

SignatureIssuer holds information about an issuer of signing certificate or key.

Type

object

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.

2.1.10. .signatures[].issuedTo

Description

SignatureSubject holds information about a person or entity who created the signature.

Type

object

Required

- **publicKeyID**

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.
publicKeyID	**string**	If present, it is a human readable key id of public key belonging to the subject used to verify image signature. It should contain at least 64 lowest bits of public key's fingerprint (e.g. 0x685ebe62bf278440).

2.2. API ENDPOINTS

The following API endpoints are available:

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

2.2.1. /apis/image.openshift.io/v1/images

HTTP method

DELETE

Description

delete collection of Image

Table 2.1. 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

Table 2.2. HTTP responses

HTTP code	Reponse body
200 - OK	**Status_v5** schema
401 - Unauthorized	Empty

HTTP method

GET

Description

list or watch objects of kind Image

Table 2.3. HTTP responses

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

HTTP method

POST

Description

create an Image

Table 2.4. Query parameters

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

Parameter	Type	Description
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Table 2.5. Body parameters

Parameter	Type	Description
body	**Image** schema	

Table 2.6. HTTP responses

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

2.2.2. /apis/image.openshift.io/v1/watch/images

HTTP method

GET

Description

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

Table 2.7. HTTP responses

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

2.2.3. /apis/image.openshift.io/v1/images/{name}

Table 2.8. Global path parameters

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

HTTP method

DELETE

Description

delete an Image

Table 2.9. 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

Table 2.10. HTTP responses

HTTP code	Reponse body
200 - OK	**Status_v5** schema
202 - Accepted	**Status_v5** schema
401 - Unauthorized	Empty

HTTP method

GET

Description

read the specified Image

Table 2.11. HTTP responses

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

HTTP method

PATCH

Description

partially update the specified Image

Table 2.12. 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
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Table 2.13. HTTP responses

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

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

HTTP method

PUT

Description

replace the specified Image

Table 2.14. Query parameters

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

Table 2.15. Body parameters

Parameter	Type	Description
body	**Image** schema	

Table 2.16. HTTP responses

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

2.2.4. /apis/image.openshift.io/v1/watch/images/{name}

Table 2.17. Global path parameters

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

HTTP method

GET

Description

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

Table 2.18. HTTP responses

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

CHAPTER 3. IMAGESIGNATURE [IMAGE.OPENSHIFT.IO/V1]

Description

ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **type**
- **content**

3.1. 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
conditions	**array**	Conditions represent the latest available observations of a signature's current state.
conditions[]	**object**	SignatureCondition describes an image signature condition of particular kind at particular probe time.
content	**string**	Required: An opaque binary string which is an image's signature.
created	**Time**	If specified, it is the time of signature's creation.

Property	Type	Description
imageIdentity	**string**	A human readable string representing image's identity. It could be a product name and version, or an image pull spec (e.g. "registry.access.redhat.com/rhel7/rhel:7.2").
issuedBy	**object**	SignatureIssuer holds information about an issuer of signing certificate or key.
issuedTo	**object**	SignatureSubject holds information about a person or entity who created the signature.
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signedClaims	**object (string)**	Contains claims from the signature.
type	**string**	Required: Describes a type of stored blob.

3.1.1. .conditions

Description

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

Type

array

3.1.2. .conditions[]

Description

SignatureCondition describes an image signature condition of particular kind at particular probe time.

Type

object

Required

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

Property	Type	Description
lastProbeTime	Time	Last time the condition was checked.
lastTransitionTime	Time	Last time the condition transit from one status to another.
message	**string**	Human readable message indicating details about last transition.
reason	**string**	(brief) reason for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of signature condition, Complete or Failed.

3.1.3. .issuedBy

Description

SignatureIssuer holds information about an issuer of signing certificate or key.

Type

object

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.

3.1.4. .issuedTo

Description

SignatureSubject holds information about a person or entity who created the signature.

Type

object

Required

- **publicKeyID**

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.
publicKeyID	**string**	If present, it is a human readable key id of public key belonging to the subject used to verify image signature. It should contain at least 64 lowest bits of public key's fingerprint (e.g. 0x685ebe62bf278440).

3.2. API ENDPOINTS

The following API endpoints are available:

- **/apis/image.openshift.io/v1/imagesignatures**
  - **POST**: create an ImageSignature
- **/apis/image.openshift.io/v1/imagesignatures/{name}**
  - **DELETE**: delete an ImageSignature

3.2.1. /apis/image.openshift.io/v1/imagesignatures

Table 3.1. Global query parameters

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

Parameter	Type	Description
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP method

POST

Description

create an ImageSignature

Table 3.2. Body parameters

Parameter	Type	Description
body	**ImageSignature** schema	

Table 3.3. HTTP responses

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

3.2.2. /apis/image.openshift.io/v1/imagesignatures/{name}

Table 3.4. Global path parameters

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

Table 3.5. Global query parameters

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

HTTP method

DELETE

Description

delete an ImageSignature

Table 3.6. HTTP responses

HTTP code	Reponse body
200 - OK	**Status_v5** schema
202 - Accepted	**Status_v5** schema
401 - Unauthorized	Empty

CHAPTER 4. IMAGESTREAMIMAGE [IMAGE.OPENSHIFT.IO/V1]

Description

ImageStreamImage represents an Image that is retrieved by image name from an ImageStream. User interfaces and regular users can use this resource to access the metadata details of a tagged image in the image stream history for viewing, since Image resources are not directly accessible to end users. A not found error will be returned if no such image is referenced by a tag within the ImageStream. Images are created when spec tags are set on an image stream that represent an image in an external registry, when pushing to the integrated registry, or when tagging an existing image from one image stream to another. The name of an image stream image is in the form "<STREAM>@<DIGEST>", where the digest is the content addressible identifier for the image (sha256:xxxxx...). You can use ImageStreamImages as the from.kind of an image stream spec tag to reference an image exactly. The only operations supported on the imagestreamimage endpoint are retrieving the image.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **image**

4.1. 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
image	**object**	Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata

4.1.1. .image

Description

Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format,

content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Property	Type	Description
apiVersion	**string**	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources
dockerImageConfig	**string**	DockerImageConfig is a JSON blob that the runtime uses to set up the container. This is a part of manifest schema v2. Will not be set when the image represents a manifest list.
dockerImageLayers	**array**	DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.
dockerImageLayers[]	**object**	ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.
dockerImageManifest	**string**	DockerImageManifest is the raw JSON of the manifest
dockerImageManifestMediaType	**string**	DockerImageManifestMediaType specifies the mediaType of manifest. This is a part of manifest schema v2.

Property	Type	Description
dockerImageManifests	**array**	DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.
dockerImageManifests[]	**object**	ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.
dockerImageMetadata	**RawExtension**	DockerImageMetadata contains metadata about this image
dockerImageMetadataVersion	**string**	DockerImageMetadataVersion conveys the version of the object, which if empty defaults to "1.0"
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image.
dockerImageSignatures	**array (string)**	DockerImageSignatures provides the signatures as opaque blobs. This is a part of manifest schema v1.
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signatures	**array**	Signatures holds all signatures of the image.

Property	Type	Description
signatures[]	**object**	ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

4.1.2. .image.dockerImageLayers

Description

DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.

Type

array

4.1.3. .image.dockerImageLayers[]

Description

ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.

Type

object

Required

- **name**
- **size**
- **mediaType**

Property	Type	Description
mediaType	**string**	MediaType of the referenced object.

Property	Type	Description
name	**string**	Name of the layer as defined by the underlying store.
size	**integer**	Size of the layer in bytes as defined by the underlying store.

4.1.4. .image.dockerImageManifests

Description

DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.

Type

array

4.1.5. .image.dockerImageManifests[]

Description

ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Type

object

Required

- **digest**
- **mediaType**
- **manifestSize**
- **architecture**
- **os**

Property	Type	Description
architecture	**string**	Architecture specifies the supported CPU architecture, for example **amd64** or **ppc64le**.
digest	**string**	Digest is the unique identifier for the manifest. It refers to an Image object.

Property	Type	Description
manifestSize	**integer**	ManifestSize represents the size of the raw object contents, in bytes.
mediaType	**string**	MediaType defines the type of the manifest, possible values are application/vnd.oci.image.manifest.v1+json, application/vnd.docker.distribution.manifest.v2+json or application/vnd.docker.distribution.manifest.v1+json.
os	**string**	OS specifies the operating system, for example **linux**.
variant	**string**	Variant is an optional field repreenting a variant of the CPU, for example v6 to specify a particular CPU variant of the ARM CPU.

4.1.6. .image.signatures

Description

Signatures holds all signatures of the image.

Type

array

4.1.7. .image.signatures[]

Description

ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **type**
- **content**

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
conditions	**array**	Conditions represent the latest available observations of a signature's current state.
conditions[]	**object**	SignatureCondition describes an image signature condition of particular kind at particular probe time.
content	**string**	Required: An opaque binary string which is an image's signature.
created	**Time**	If specified, it is the time of signature's creation.
imageIdentity	**string**	A human readable string representing image's identity. It could be a product name and version, or an image pull spec (e.g. "registry.access.redhat.com/rhel7/rhel:7.2").
issuedBy	**object**	SignatureIssuer holds information about an issuer of signing certificate or key.
issuedTo	**object**	SignatureSubject holds information about a person or entity who created the signature.

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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signedClaims	**object (string)**	Contains claims from the signature.
type	**string**	Required: Describes a type of stored blob.

4.1.8. .image.signatures[].conditions

Description

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

Type

array

4.1.9. .image.signatures[].conditions[]

Description

SignatureCondition describes an image signature condition of particular kind at particular probe time.

Type

object

Required

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

Property	Type	Description
lastProbeTime	Time	Last time the condition was checked.
lastTransitionTime	Time	Last time the condition transit from one status to another.
message	**string**	Human readable message indicating details about last transition.
reason	**string**	(brief) reason for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of signature condition, Complete or Failed.

4.1.10. .image.signatures[].issuedBy

Description

SignatureIssuer holds information about an issuer of signing certificate or key.

Type

object

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.

4.1.11. .image.signatures[].issuedTo

Description

SignatureSubject holds information about a person or entity who created the signature.

Type

object

Required

- **publicKeyID**

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.
publicKeyID	**string**	If present, it is a human readable key id of public key belonging to the subject used to verify image signature. It should contain at least 64 lowest bits of public key's fingerprint (e.g. 0x685ebe62bf278440).

4.2. API ENDPOINTS

The following API endpoints are available:

- **/apis/image.openshift.io/v1/namespaces/{namespace}/imagestreamimages/{name}**
  - **GET**: read the specified ImageStreamImage

4.2.1. /apis/image.openshift.io/v1/namespaces/{namespace}/imagestreamimages/{na

Table 4.1. Global path parameters

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

HTTP method

GET

Description

read the specified ImageStreamImage

Table 4.2. HTTP responses

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

CHAPTER 5. IMAGESTREAMIMPORT [IMAGE.OPENSHIFT.IO/V1]

Description

The image stream import resource provides an easy way for a user to find and import container images from other container image registries into the server. Individual images or an entire image repository may be imported, and users may choose to see the results of the import prior to tagging the resulting images into the specified image stream.
This API is intended for end-user tools that need to see the metadata of the image prior to import (for instance, to generate an application from it). Clients that know the desired image can continue to create spec.tags directly into their image streams.

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**
- **status**

5.1. 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
kind	**string**	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

Property	Type	Description
metadata	**ObjectMeta_v2**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
spec	**object**	ImageStreamImportSpec defines what images should be imported.
status	**object**	ImageStreamImportStatus contains information about the status of an image stream import.

5.1.1. .spec

Description

ImageStreamImportSpec defines what images should be imported.

Type

object

Required

- **import**

Property	Type	Description
images	**array**	Images are a list of individual images to import.
images[]	**object**	ImageImportSpec describes a request to import a specific image.
import	**boolean**	Import indicates whether to perform an import - if so, the specified tags are set on the spec and status of the image stream defined by the type meta.
repository	**object**	RepositoryImportSpec describes a request to import images from a container image repository.

5.1.2. .spec.images

Description

Images are a list of individual images to import.

Type

array

5.1.3. .spec.images[]

Description

ImageImportSpec describes a request to import a specific image.

Type

object

Required

- **from**

Property	Type	Description
from	ObjectReference	From is the source of an image to import; only kind DockerImage is allowed
importPolicy	**object**	TagImportPolicy controls how images related to this tag will be imported.
includeManifest	**boolean**	IncludeManifest determines if the manifest for each image is returned in the response
referencePolicy	**object**	TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.
to	LocalObjectReference_v2	To is a tag in the current image stream to assign the imported image to, if name is not specified the default tag from from.name will be used

5.1.4. .spec.images[].importPolicy

Description

TagImportPolicy controls how images related to this tag will be imported.

Type

object

Property	Type	Description
importMode	**string**	ImportMode describes how to import an image manifest.
insecure	**boolean**	Insecure is true if the server may bypass certificate verification or connect directly over HTTP during image import.
scheduled	**boolean**	Scheduled indicates to the server that this tag should be periodically checked to ensure it is up to date, and imported

5.1.5. .spec.images[].referencePolicy

Description

TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.

Type

object

Required

- **type**

Property	Type	Description

Property	Type	Description
type	**string**	Type determines how the image pull spec should be transformed when the image stream tag is used in deployment config triggers or new builds. The default value is **Source**, indicating the original location of the image should be used (if imported). The user may also specify **Local**, indicating that the pull spec should point to the integrated container image registry and leverage the registry's ability to proxy the pull to an upstream registry. **Local** allows the credentials used to pull this image to be managed from the image stream's namespace, so others on the platform can access a remote image but have no access to the remote secret. It also allows the image layers to be mirrored into the local registry which the images can still be pulled even if the upstream registry is unavailable.

5.1.6. .spec.repository

Description

RepositoryImportSpec describes a request to import images from a container image repository.

Type

object

Required

- **from**

Property	Type	Description
from	ObjectReference	From is the source for the image repository to import; only kind DockerImage and a name of a container image repository is allowed

Property	Type	Description
importPolicy	**object**	TagImportPolicy controls how images related to this tag will be imported.
includeManifest	**boolean**	IncludeManifest determines if the manifest for each image is returned in the response
referencePolicy	**object**	TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.

5.1.7. .spec.repository.importPolicy

Description

TagImportPolicy controls how images related to this tag will be imported.

Type

object

Property	Type	Description
importMode	**string**	ImportMode describes how to import an image manifest.
insecure	**boolean**	Insecure is true if the server may bypass certificate verification or connect directly over HTTP during image import.
scheduled	**boolean**	Scheduled indicates to the server that this tag should be periodically checked to ensure it is up to date, and imported

5.1.8. .spec.repository.referencePolicy

Description

TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.

Type

object

Required

- **type**

Property	Type	Description
type	**string**	Type determines how the image pull spec should be transformed when the image stream tag is used in deployment config triggers or new builds. The default value is **Source**, indicating the original location of the image should be used (if imported). The user may also specify **Local**, indicating that the pull spec should point to the integrated container image registry and leverage the registry's ability to proxy the pull to an upstream registry. **Local** allows the credentials used to pull this image to be managed from the image stream's namespace, so others on the platform can access a remote image but have no access to the remote secret. It also allows the image layers to be mirrored into the local registry which the images can still be pulled even if the upstream registry is unavailable.

5.1.9. .status

Description

ImageStreamImportStatus contains information about the status of an image stream import.

Type

object

Property	Type	Description
images	**array**	Images is set with the result of importing spec.images
images[]	**object**	ImageImportStatus describes the result of an image import.

Property	Type	Description
import	**object**	An ImageStream stores a mapping of tags to images, metadata overrides that are applied when images are tagged in a stream, and an optional reference to a container image repository on a registry. Users typically update the spec.tags field to point to external images which are imported from container registries using credentials in your namespace with the pull secret type, or to existing image stream tags and images which are immediately accessible for tagging or pulling. The history of images applied to a tag is visible in the status.tags field and any user who can view an image stream is allowed to tag that image into their own image streams. Access to pull images from the integrated registry is granted by having the "get imagestreams/layers" permission on a given image stream. Users may remove a tag by deleting the imagestreamtag resource, which causes both spec and status for that tag to be removed. Image stream history is retained until an administrator runs the prune operation, which removes references that are no longer in use. To preserve a historical image, ensure there is a tag in spec pointing to that image by its digest. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).
repository	**object**	RepositoryImportStatus describes the result of an image repository import

5.1.10. .status.images

Description

Images is set with the result of importing spec.images

Type

array

5.1.11. .status.images[]

Description

ImageImportStatus describes the result of an image import.

Type

object

Required

- **status**

Property	Type	Description
image	**object**	Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).
manifests	**array**	Manifests holds sub-manifests metadata when importing a manifest list

Property	Type	Description
manifests[]	**object**	Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).
status	**Status_v5**	Status is the status of the image import, including errors encountered while retrieving the image
tag	**string**	Tag is the tag this image was located under, if any

5.1.12. .status.images[].image

Description

Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents.

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

Type

object

Property	Type	Description
apiVersion	**string**	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources
dockerImageConfig	**string**	DockerImageConfig is a JSON blob that the runtime uses to set up the container. This is a part of manifest schema v2. Will not be set when the image represents a manifest list.
dockerImageLayers	**array**	DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.
dockerImageLayers[]	**object**	ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.
dockerImageManifest	**string**	DockerImageManifest is the raw JSON of the manifest
dockerImageManifestMediaType	**string**	DockerImageManifestMediaType specifies the mediaType of manifest. This is a part of manifest schema v2.
dockerImageManifests	**array**	DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.
dockerImageManifests[]	**object**	ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Property	Type	Description
dockerImageMetadata	**RawExtension**	DockerImageMetadata contains metadata about this image
dockerImageMetadataVersion	**string**	DockerImageMetadataVersion conveys the version of the object, which if empty defaults to "1.0"
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image.
dockerImageSignatures	**array (string)**	DockerImageSignatures provides the signatures as opaque blobs. This is a part of manifest schema v1.
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signatures	**array**	Signatures holds all signatures of the image.

Property	Type	Description
signatures[]	**object**	ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

5.1.13. .status.images[].image.dockerImageLayers

Description

DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.

Type

array

5.1.14. .status.images[].image.dockerImageLayers[]

Description

ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.

Type

object

Required

- **name**
- **size**
- **mediaType**

Property	Type	Description
mediaType	**string**	MediaType of the referenced object.

Property	Type	Description
name	**string**	Name of the layer as defined by the underlying store.
size	**integer**	Size of the layer in bytes as defined by the underlying store.

5.1.15. .status.images[].image.dockerImageManifests

Description

DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.

Type

array

5.1.16. .status.images[].image.dockerImageManifests[]

Description

ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Type

object

Required

- **digest**
- **mediaType**
- **manifestSize**
- **architecture**
- **os**

Property	Type	Description
architecture	**string**	Architecture specifies the supported CPU architecture, for example **amd64** or **ppc64le**.
digest	**string**	Digest is the unique identifier for the manifest. It refers to an Image object.

Property	Type	Description
manifestSize	**integer**	ManifestSize represents the size of the raw object contents, in bytes.
mediaType	**string**	MediaType defines the type of the manifest, possible values are application/vnd.oci.image.manifest.v1+json, application/vnd.docker.distribution.manifest.v2+json or application/vnd.docker.distribution.manifest.v1+json.
os	**string**	OS specifies the operating system, for example **linux**.
variant	**string**	Variant is an optional field repreenting a variant of the CPU, for example v6 to specify a particular CPU variant of the ARM CPU.

5.1.17. .status.images[].image.signatures

Description

Signatures holds all signatures of the image.

Type

array

5.1.18. .status.images[].image.signatures[]

Description

ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **type**
- **content**

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
conditions	**array**	Conditions represent the latest available observations of a signature's current state.
conditions[]	**object**	SignatureCondition describes an image signature condition of particular kind at particular probe time.
content	**string**	Required: An opaque binary string which is an image's signature.
created	**Time**	If specified, it is the time of signature's creation.
imageIdentity	**string**	A human readable string representing image's identity. It could be a product name and version, or an image pull spec (e.g. "registry.access.redhat.com/rhel7/rhel:7.2").
issuedBy	**object**	SignatureIssuer holds information about an issuer of signing certificate or key.
issuedTo	**object**	SignatureSubject holds information about a person or entity who created the signature.

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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signedClaims	**object (string)**	Contains claims from the signature.
type	**string**	Required: Describes a type of stored blob.

5.1.19. .status.images[].image.signatures[].conditions

Description

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

Type

array

5.1.20. .status.images[].image.signatures[].conditions[]

Description

SignatureCondition describes an image signature condition of particular kind at particular probe time.

Type

object

Required

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

Property	Type	Description
lastProbeTime	**Time**	Last time the condition was checked.
lastTransitionTime	**Time**	Last time the condition transit from one status to another.
message	**string**	Human readable message indicating details about last transition.
reason	**string**	(brief) reason for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of signature condition, Complete or Failed.

5.1.21. .status.images[].image.signatures[].issuedBy

Description

SignatureIssuer holds information about an issuer of signing certificate or key.

Type

object

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.

5.1.22. .status.images[].image.signatures[].issuedTo

Description

SignatureSubject holds information about a person or entity who created the signature.

Type

object

Required

- **publicKeyID**

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.
publicKeyID	**string**	If present, it is a human readable key id of public key belonging to the subject used to verify image signature. It should contain at least 64 lowest bits of public key's fingerprint (e.g. 0x685ebe62bf278440).

5.1.23. .status.images[].manifests

Description

Manifests holds sub-manifests metadata when importing a manifest list

Type

array

5.1.24. .status.images[].manifests[]

Description

Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Property	Type	Description

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
dockerImageConfig	**string**	DockerImageConfig is a JSON blob that the runtime uses to set up the container. This is a part of manifest schema v2. Will not be set when the image represents a manifest list.
dockerImageLayers	**array**	DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.
dockerImageLayers[]	**object**	ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.
dockerImageManifest	**string**	DockerImageManifest is the raw JSON of the manifest
dockerImageManifestMediaType	**string**	DockerImageManifestMediaType specifies the mediaType of manifest. This is a part of manifest schema v2.
dockerImageManifests	**array**	DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.
dockerImageManifests[]	**object**	ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Property	Type	Description
dockerImageMetadata	**RawExtension**	DockerImageMetadata contains metadata about this image
dockerImageMetadataVersion	**string**	DockerImageMetadataVersion conveys the version of the object, which if empty defaults to "1.0"
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image.
dockerImageSignatures	**array (string)**	DockerImageSignatures provides the signatures as opaque blobs. This is a part of manifest schema v1.
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signatures	**array**	Signatures holds all signatures of the image.

Property	Type	Description
signatures[]	**object**	ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

5.1.25. .status.images[].manifests[].dockerImageLayers

Description

DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.

Type

array

5.1.26. .status.images[].manifests[].dockerImageLayers[]

Description

ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.

Type

object

Required

- **name**
- **size**
- **mediaType**

Property	Type	Description
mediaType	**string**	MediaType of the referenced object.

Property	Type	Description
name	**string**	Name of the layer as defined by the underlying store.
size	**integer**	Size of the layer in bytes as defined by the underlying store.

5.1.27. .status.images[].manifests[].dockerImageManifests

Description

DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.

Type

array

5.1.28. .status.images[].manifests[].dockerImageManifests[]

Description

ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Type

object

Required

- **digest**
- **mediaType**
- **manifestSize**
- **architecture**
- **os**

Property	Type	Description
architecture	**string**	Architecture specifies the supported CPU architecture, for example **amd64** or **ppc64le**.
digest	**string**	Digest is the unique identifier for the manifest. It refers to an Image object.

Property	Type	Description
manifestSize	**integer**	ManifestSize represents the size of the raw object contents, in bytes.
mediaType	**string**	MediaType defines the type of the manifest, possible values are application/vnd.oci.image.manifest.v1+json, application/vnd.docker.distribution.manifest.v2+json or application/vnd.docker.distribution.manifest.v1+json.
os	**string**	OS specifies the operating system, for example **linux**.
variant	**string**	Variant is an optional field repreenting a variant of the CPU, for example v6 to specify a particular CPU variant of the ARM CPU.

5.1.29. .status.images[].manifests[].signatures

Description

Signatures holds all signatures of the image.

Type

array

5.1.30. .status.images[].manifests[].signatures[]

Description

ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **type**
- **content**

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
conditions	**array**	Conditions represent the latest available observations of a signature's current state.
conditions[]	**object**	SignatureCondition describes an image signature condition of particular kind at particular probe time.
content	**string**	Required: An opaque binary string which is an image's signature.
created	**Time**	If specified, it is the time of signature's creation.
imageIdentity	**string**	A human readable string representing image's identity. It could be a product name and version, or an image pull spec (e.g. "registry.access.redhat.com/rhel7/rhel:7.2").
issuedBy	**object**	SignatureIssuer holds information about an issuer of signing certificate or key.
issuedTo	**object**	SignatureSubject holds information about a person or entity who created the signature.

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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signedClaims	**object (string)**	Contains claims from the signature.
type	**string**	Required: Describes a type of stored blob.

5.1.31. .status.images[].manifests[].signatures[].conditions

Description

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

Type

array

5.1.32. .status.images[].manifests[].signatures[].conditions[]

Description

SignatureCondition describes an image signature condition of particular kind at particular probe time.

Type

object

Required

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

Property	Type	Description
lastProbeTime	Time	Last time the condition was checked.
lastTransitionTime	Time	Last time the condition transit from one status to another.
message	**string**	Human readable message indicating details about last transition.
reason	**string**	(brief) reason for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of signature condition, Complete or Failed.

5.1.33. .status.images[].manifests[].signatures[].issuedBy

Description

SignatureIssuer holds information about an issuer of signing certificate or key.

Type

object

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.

5.1.34. .status.images[].manifests[].signatures[].issuedTo

Description

SignatureSubject holds information about a person or entity who created the signature.

Type

object

Required

- **publicKeyID**

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.
publicKeyID	**string**	If present, it is a human readable key id of public key belonging to the subject used to verify image signature. It should contain at least 64 lowest bits of public key's fingerprint (e.g. 0x685ebe62bf278440).

5.1.35. .status.import

Description

An ImageStream stores a mapping of tags to images, metadata overrides that are applied when images are tagged in a stream, and an optional reference to a container image repository on a registry. Users typically update the spec.tags field to point to external images which are imported from container registries using credentials in your namespace with the pull secret type, or to existing image stream tags and images which are immediately accessible for tagging or pulling. The history of images applied to a tag is visible in the status.tags field and any user who can view an image stream is allowed to tag that image into their own image streams. Access to pull images from the integrated registry is granted by having the "get imagestreams/layers" permission on a given image stream. Users may remove a tag by deleting the imagestreamtag resource, which causes both spec and status for that tag to be removed. Image stream history is retained until an administrator runs the prune operation, which removes references that are no longer in use. To preserve a historical image, ensure there is a tag in spec pointing to that image by its digest.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

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

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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
spec	**object**	ImageStreamSpec represents options for ImageStreams.
status	**object**	ImageStreamStatus contains information about the state of this image stream.

5.1.36. .status.import.spec

Description

ImageStreamSpec represents options for ImageStreams.

Type

object

Property	Type	Description
dockerImageRepository	**string**	dockerImageRepository is optional, if specified this stream is backed by a container repository on this server Deprecated: This field is deprecated as of v3.7 and will be removed in a future release. Specify the source for the tags to be imported in each tag via the spec.tags.from reference instead.

Property	Type	Description
lookupPolicy	**object**	ImageLookupPolicy describes how an image stream can be used to override the image references used by pods, builds, and other resources in a namespace.
tags	**array**	tags map arbitrary string values to specific image locators
tags[]	**object**	TagReference specifies optional annotations for images using this tag and an optional reference to an ImageStreamTag, ImageStreamImage, or DockerImage this tag should track.

5.1.37. .status.import.spec.lookupPolicy

Description

ImageLookupPolicy describes how an image stream can be used to override the image references used by pods, builds, and other resources in a namespace.

Type

object

Required

- **local**

Property	Type	Description
local	**boolean**	local will change the docker short image references (like "mysql" or "php:latest") on objects in this namespace to the image ID whenever they match this image stream, instead of reaching out to a remote registry. The name will be fully qualified to an image ID if found. The tag's referencePolicy is taken into account on the replaced value. Only works within the current namespace.

5.1.38. .status.import.spec.tags

Description

tags map arbitrary string values to specific image locators

Type

array

5.1.39. .status.import.spec.tags[]

Description

TagReference specifies optional annotations for images using this tag and an optional reference to an ImageStreamTag, ImageStreamImage, or DockerImage this tag should track.

Type

object

Required

- **name**

Property	Type	Description
annotations	**object (string)**	Optional; if specified, annotations that are applied to images retrieved via ImageStreamTags.
from	**ObjectReference**	Optional; if specified, a reference to another image that this tag should point to. Valid values are ImageStreamTag, ImageStreamImage, and DockerImage. ImageStreamTag references can only reference a tag within this same ImageStream.

Property	Type	Description
generation	**integer**	Generation is a counter that tracks mutations to the spec tag (user intent). When a tag reference is changed the generation is set to match the current stream generation (which is incremented every time spec is changed). Other processes in the system like the image importer observe that the generation of spec tag is newer than the generation recorded in the status and use that as a trigger to import the newest remote tag. To trigger a new import, clients may set this value to zero which will reset the generation to the latest stream generation. Legacy clients will send this value as nil which will be merged with the current tag generation.
importPolicy	**object**	TagImportPolicy controls how images related to this tag will be imported.
name	**string**	Name of the tag
reference	**boolean**	Reference states if the tag will be imported. Default value is false, which means the tag will be imported.
referencePolicy	**object**	TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.

5.1.40. .status.import.spec.tags[].importPolicy

Description

TagImportPolicy controls how images related to this tag will be imported.

Type

object

Property	Type	Description
importMode	**string**	ImportMode describes how to import an image manifest.
insecure	**boolean**	Insecure is true if the server may bypass certificate verification or connect directly over HTTP during image import.
scheduled	**boolean**	Scheduled indicates to the server that this tag should be periodically checked to ensure it is up to date, and imported

5.1.41. .status.import.spec.tags[].referencePolicy

Description

TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.

Type

object

Required

- **type**

Property	Type	Description

Property	Type	Description
type	**string**	Type determines how the image pull spec should be transformed when the image stream tag is used in deployment config triggers or new builds. The default value is **Source**, indicating the original location of the image should be used (if imported). The user may also specify **Local**, indicating that the pull spec should point to the integrated container image registry and leverage the registry's ability to proxy the pull to an upstream registry. **Local** allows the credentials used to pull this image to be managed from the image stream's namespace, so others on the platform can access a remote image but have no access to the remote secret. It also allows the image layers to be mirrored into the local registry which the images can still be pulled even if the upstream registry is unavailable.

5.1.42. .status.import.status

Description

ImageStreamStatus contains information about the state of this image stream.

Type

object

Required

- **dockerImageRepository**

Property	Type	Description
dockerImageRepository	**string**	DockerImageRepository represents the effective location this stream may be accessed at. May be empty until the server determines where the repository is located

Property	Type	Description
publicDockerImageRepository	**string**	PublicDockerImageRepository represents the public location from where the image can be pulled outside the cluster. This field may be empty if the administrator has not exposed the integrated registry externally.
tags	**array**	Tags are a historical record of images associated with each tag. The first entry in the TagEvent array is the currently tagged image.
tags[]	**object**	NamedTagEventList relates a tag to its image history.

5.1.43. .status.import.status.tags

Description

Tags are a historical record of images associated with each tag. The first entry in the TagEvent array is the currently tagged image.

Type

array

5.1.44. .status.import.status.tags[]

Description

NamedTagEventList relates a tag to its image history.

Type

object

Required

- **tag**
- **items**

Property	Type	Description
conditions	**array**	Conditions is an array of conditions that apply to the tag event list.

Property	Type	Description
conditions[]	**object**	TagEventCondition contains condition information for a tag event.
items	**array**	Standard object's metadata.
items[]	**object**	TagEvent is used by ImageStreamStatus to keep a historical record of images associated with a tag.
tag	**string**	Tag is the tag for which the history is recorded

5.1.45. .status.import.status.tags[].conditions

Description

Conditions is an array of conditions that apply to the tag event list.

Type

array

5.1.46. .status.import.status.tags[].conditions[]

Description

TagEventCondition contains condition information for a tag event.

Type

object

Required

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

Property	Type	Description
generation	**integer**	Generation is the spec tag generation that this status corresponds to

Property	Type	Description
lastTransitionTime	Time	LastTransitionTIme is the time the condition transitioned from one status to another.
message	**string**	Message is a human readable description of the details about last transition, complementing reason.
reason	**string**	Reason is a brief machine readable explanation for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of tag event condition, currently only ImportSuccess

5.1.47. .status.import.status.tags[].items

Description

Standard object's metadata.

Type

array

5.1.48. .status.import.status.tags[].items[]

Description

TagEvent is used by ImageStreamStatus to keep a historical record of images associated with a tag.

Type

object

Required

- **created**
- **dockerImageReference**
- **image**
- **generation**

Property	Type	Description
created	Time	Created holds the time the TagEvent was created

Property	Type	Description
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image
generation	**integer**	Generation is the spec tag generation that resulted in this tag being updated
image	**string**	Image is the image

5.1.49. .status.repository

Description

RepositoryImportStatus describes the result of an image repository import

Type

object

Property	Type	Description
additionalTags	**array (string)**	AdditionalTags are tags that exist in the repository but were not imported because a maximum limit of automatic imports was applied.
images	**array**	Images is a list of images successfully retrieved by the import of the repository.
images[]	**object**	ImageImportStatus describes the result of an image import.
status	**Status_v5**	Status reflects whether any failure occurred during import

5.1.50. .status.repository.images

Description

Images is a list of images successfully retrieved by the import of the repository.

Type

array

5.1.51. .status.repository.images[]

Description

ImageImportStatus describes the result of an image import.

Type

object

Required

- **status**

Property	Type	Description
image	**object**	Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).
manifests	**array**	Manifests holds sub-manifests metadata when importing a manifest list

Property	Type	Description
manifests[]	**object**	Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).
status	**Status_v5**	Status is the status of the image import, including errors encountered while retrieving the image
tag	**string**	Tag is the tag this image was located under, if any

5.1.52. .status.repository.images[].image

Description

Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Property	Type	Description
apiVersion	**string**	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources
dockerImageConfig	**string**	DockerImageConfig is a JSON blob that the runtime uses to set up the container. This is a part of manifest schema v2. Will not be set when the image represents a manifest list.
dockerImageLayers	**array**	DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.
dockerImageLayers[]	**object**	ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.
dockerImageManifest	**string**	DockerImageManifest is the raw JSON of the manifest
dockerImageManifestMediaType	**string**	DockerImageManifestMediaType specifies the mediaType of manifest. This is a part of manifest schema v2.
dockerImageManifests	**array**	DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.
dockerImageManifests[]	**object**	ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Property	Type	Description
dockerImageMetadata	**RawExtension**	DockerImageMetadata contains metadata about this image
dockerImageMetadataVersion	**string**	DockerImageMetadataVersion conveys the version of the object, which if empty defaults to "1.0"
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image.
dockerImageSignatures	**array (string)**	DockerImageSignatures provides the signatures as opaque blobs. This is a part of manifest schema v1.
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signatures	**array**	Signatures holds all signatures of the image.

Property	Type	Description
signatures[]	**object**	ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

5.1.53. .status.repository.images[].image.dockerImageLayers

Description

DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.

Type

array

5.1.54. .status.repository.images[].image.dockerImageLayers[]

Description

ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.

Type

object

Required

- **name**
- **size**
- **mediaType**

Property	Type	Description
mediaType	**string**	MediaType of the referenced object.

Property	Type	Description
name	**string**	Name of the layer as defined by the underlying store.
size	**integer**	Size of the layer in bytes as defined by the underlying store.

5.1.55. .status.repository.images[].image.dockerImageManifests

Description

DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.

Type

array

5.1.56. .status.repository.images[].image.dockerImageManifests[]

Description

ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Type

object

Required

- **digest**
- **mediaType**
- **manifestSize**
- **architecture**
- **os**

Property	Type	Description
architecture	**string**	Architecture specifies the supported CPU architecture, for example **amd64** or **ppc64le**.
digest	**string**	Digest is the unique identifier for the manifest. It refers to an Image object.

Property	Type	Description
manifestSize	**integer**	ManifestSize represents the size of the raw object contents, in bytes.
mediaType	**string**	MediaType defines the type of the manifest, possible values are application/vnd.oci.image.manifest.v1+json, application/vnd.docker.distribution.manifest.v2+json or application/vnd.docker.distribution.manifest.v1+json.
os	**string**	OS specifies the operating system, for example **linux**.
variant	**string**	Variant is an optional field repreenting a variant of the CPU, for example v6 to specify a particular CPU variant of the ARM CPU.

5.1.57. .status.repository.images[].image.signatures

Description

Signatures holds all signatures of the image.

Type

array

5.1.58. .status.repository.images[].image.signatures[]

Description

ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **type**
- **content**

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
conditions	**array**	Conditions represent the latest available observations of a signature's current state.
conditions[]	**object**	SignatureCondition describes an image signature condition of particular kind at particular probe time.
content	**string**	Required: An opaque binary string which is an image's signature.
created	**Time**	If specified, it is the time of signature's creation.
imageIdentity	**string**	A human readable string representing image's identity. It could be a product name and version, or an image pull spec (e.g. "registry.access.redhat.com/rhel7/rhel:7.2").
issuedBy	**object**	SignatureIssuer holds information about an issuer of signing certificate or key.
issuedTo	**object**	SignatureSubject holds information about a person or entity who created the signature.

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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signedClaims	**object (string)**	Contains claims from the signature.
type	**string**	Required: Describes a type of stored blob.

5.1.59. .status.repository.images[].image.signatures[].conditions

Description

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

Type

array

5.1.60. .status.repository.images[].image.signatures[].conditions[]

Description

SignatureCondition describes an image signature condition of particular kind at particular probe time.

Type

object

Required

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

Property	Type	Description
lastProbeTime	**Time**	Last time the condition was checked.
lastTransitionTime	**Time**	Last time the condition transit from one status to another.
message	**string**	Human readable message indicating details about last transition.
reason	**string**	(brief) reason for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of signature condition, Complete or Failed.

5.1.61. .status.repository.images[].image.signatures[].issuedBy

Description

SignatureIssuer holds information about an issuer of signing certificate or key.

Type

object

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.

5.1.62. .status.repository.images[].image.signatures[].issuedTo

Description

SignatureSubject holds information about a person or entity who created the signature.

Type

object

Required

- **publicKeyID**

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.
publicKeyID	**string**	If present, it is a human readable key id of public key belonging to the subject used to verify image signature. It should contain at least 64 lowest bits of public key's fingerprint (e.g. 0x685ebe62bf278440).

5.1.63. .status.repository.images[].manifests

Description

Manifests holds sub-manifests metadata when importing a manifest list

Type

array

5.1.64. .status.repository.images[].manifests[]

Description

Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Property	Type	Description

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
dockerImageConfig	**string**	DockerImageConfig is a JSON blob that the runtime uses to set up the container. This is a part of manifest schema v2. Will not be set when the image represents a manifest list.
dockerImageLayers	**array**	DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.
dockerImageLayers[]	**object**	ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.
dockerImageManifest	**string**	DockerImageManifest is the raw JSON of the manifest
dockerImageManifestMediaType	**string**	DockerImageManifestMediaType specifies the mediaType of manifest. This is a part of manifest schema v2.
dockerImageManifests	**array**	DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.
dockerImageManifests[]	**object**	ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Property	Type	Description
dockerImageMetadata	**RawExtension**	DockerImageMetadata contains metadata about this image
dockerImageMetadataVersion	**string**	DockerImageMetadataVersion conveys the version of the object, which if empty defaults to "1.0"
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image.
dockerImageSignatures	**array (string)**	DockerImageSignatures provides the signatures as opaque blobs. This is a part of manifest schema v1.
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signatures	**array**	Signatures holds all signatures of the image.

Property	Type	Description
signatures[]	**object**	ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

5.1.65. .status.repository.images[].manifests[].dockerImageLayers

Description

DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.

Type

array

5.1.66. .status.repository.images[].manifests[].dockerImageLayers[]

Description

ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.

Type

object

Required

- **name**
- **size**
- **mediaType**

Property	Type	Description
mediaType	**string**	MediaType of the referenced object.

Property	Type	Description
name	**string**	Name of the layer as defined by the underlying store.
size	**integer**	Size of the layer in bytes as defined by the underlying store.

5.1.67. .status.repository.images[].manifests[].dockerImageManifests

Description

DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.

Type

array

5.1.68. .status.repository.images[].manifests[].dockerImageManifests[]

Description

ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Type

object

Required

- **digest**
- **mediaType**
- **manifestSize**
- **architecture**
- **os**

Property	Type	Description
architecture	**string**	Architecture specifies the supported CPU architecture, for example **amd64** or **ppc64le**.
digest	**string**	Digest is the unique identifier for the manifest. It refers to an Image object.

Property	Type	Description
manifestSize	**integer**	ManifestSize represents the size of the raw object contents, in bytes.
mediaType	**string**	MediaType defines the type of the manifest, possible values are application/vnd.oci.image.manifest.v1+json, application/vnd.docker.distribution.manifest.v2+json or application/vnd.docker.distribution.manifest.v1+json.
os	**string**	OS specifies the operating system, for example **linux**.
variant	**string**	Variant is an optional field repreenting a variant of the CPU, for example v6 to specify a particular CPU variant of the ARM CPU.

5.1.69. .status.repository.images[].manifests[].signatures

Description

Signatures holds all signatures of the image.

Type

array

5.1.70. .status.repository.images[].manifests[].signatures[]

Description

ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **type**
- **content**

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
conditions	**array**	Conditions represent the latest available observations of a signature's current state.
conditions[]	**object**	SignatureCondition describes an image signature condition of particular kind at particular probe time.
content	**string**	Required: An opaque binary string which is an image's signature.
created	**Time**	If specified, it is the time of signature's creation.
imageIdentity	**string**	A human readable string representing image's identity. It could be a product name and version, or an image pull spec (e.g. "registry.access.redhat.com/rhel7/rhel:7.2").
issuedBy	**object**	SignatureIssuer holds information about an issuer of signing certificate or key.
issuedTo	**object**	SignatureSubject holds information about a person or entity who created the signature.

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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signedClaims	**object (string)**	Contains claims from the signature.
type	**string**	Required: Describes a type of stored blob.

5.1.71. .status.repository.images[].manifests[].signatures[].conditions

Description

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

Type

array

5.1.72. .status.repository.images[].manifests[].signatures[].conditions[]

Description

SignatureCondition describes an image signature condition of particular kind at particular probe time.

Type

object

Required

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

Property	Type	Description
lastProbeTime	Time	Last time the condition was checked.
lastTransitionTime	Time	Last time the condition transit from one status to another.
message	**string**	Human readable message indicating details about last transition.
reason	**string**	(brief) reason for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of signature condition, Complete or Failed.

5.1.73. .status.repository.images[].manifests[].signatures[].issuedBy

Description

SignatureIssuer holds information about an issuer of signing certificate or key.

Type

object

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.

5.1.74. .status.repository.images[].manifests[].signatures[].issuedTo

Description

SignatureSubject holds information about a person or entity who created the signature.

Type

object

Required

- **publicKeyID**

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.
publicKeyID	**string**	If present, it is a human readable key id of public key belonging to the subject used to verify image signature. It should contain at least 64 lowest bits of public key's fingerprint (e.g. 0x685ebe62bf278440).

5.2. API ENDPOINTS

The following API endpoints are available:

- **/apis/image.openshift.io/v1/namespaces/{namespace}/imagestreamimports**
  - **POST**: create an ImageStreamImport

5.2.1. /apis/image.openshift.io/v1/namespaces/{namespace}/imagestreamimports

Table 5.1. Global query parameters

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

Parameter	Type	Description
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP method

POST

Description

create an ImageStreamImport

Table 5.2. Body parameters

Parameter	Type	Description
body	**ImageStreamImport** schema	

Table 5.3. HTTP responses

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

CHAPTER 6. IMAGESTREAMLAYERS [IMAGE.OPENSHIFT.IO/V1]

Description

ImageStreamLayers describes information about the layers referenced by images in this image stream.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **blobs**
- **images**

6.1. 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
blobs	**object**	blobs is a map of blob name to metadata about the blob.
blobs{}	**object**	ImageLayerData contains metadata about an image layer.
images	**object**	images is a map between an image name and the names of the blobs and config that comprise the image.
images{}	**object**	ImageBlobReferences describes the blob references within an image.

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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata

6.1.1. .blobs

Description

blobs is a map of blob name to metadata about the blob.

Type

object

6.1.2. .blobs{}

Description

ImageLayerData contains metadata about an image layer.

Type

object

Required

- **size**
- **mediaType**

Property	Type	Description
mediaType	**string**	MediaType of the referenced object.

Property	Type	Description
size	**integer**	Size of the layer in bytes as defined by the underlying store. This field is optional if the necessary information about size is not available.

6.1.3. .images

Description

images is a map between an image name and the names of the blobs and config that comprise the image.

Type

object

6.1.4. .images{}

Description

ImageBlobReferences describes the blob references within an image.

Type

object

Property	Type	Description
config	**string**	config, if set, is the blob that contains the image config. Some images do not have separate config blobs and this field will be set to nil if so.
imageMissing	**boolean**	imageMissing is true if the image is referenced by the image stream but the image object has been deleted from the API by an administrator. When this field is set, layers and config fields may be empty and callers that depend on the image metadata should consider the image to be unavailable for download or viewing.

Property	Type	Description
layers	**array (string)**	layers is the list of blobs that compose this image, from base layer to top layer. All layers referenced by this array will be defined in the blobs map. Some images may have zero layers.
manifests	**array (string)**	manifests is the list of other image names that this image points to. For a single architecture image, it is empty. For a multi-arch image, it consists of the digests of single architecture images, such images shouldn't have layers nor config.

6.2. API ENDPOINTS

The following API endpoints are available:

- **/apis/image.openshift.io/v1/namespaces/{namespace}/imagestreams/{name}/layers**
  - **GET**: read layers of the specified ImageStream

6.2.1. /apis/image.openshift.io/v1/namespaces/{namespace}/imagestreams/{name}/l

Table 6.1. Global path parameters

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

HTTP method

GET

Description

read layers of the specified ImageStream

Table 6.2. HTTP responses

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

CHAPTER 7. IMAGESTREAMMAPPING [IMAGE.OPENSHIFT.IO/V1]

Description

ImageStreamMapping represents a mapping from a single image stream tag to a container image as well as the reference to the container image stream the image came from. This resource is used by privileged integrators to create an image resource and to associate it with an image stream in the status tags field. Creating an ImageStreamMapping will allow any user who can view the image stream to tag or pull that image, so only create mappings where the user has proven they have access to the image contents directly. The only operation supported for this resource is create and the metadata name and namespace should be set to the image stream containing the tag that should be updated.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **image**
- **tag**

7.1. 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
image	**object**	Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
tag	**string**	Tag is a string value this image can be located with inside the stream.

7.1.1. .image

Description

Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents.

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

Type

object

Property	Type	Description
apiVersion	**string**	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources
dockerImageConfig	**string**	DockerImageConfig is a JSON blob that the runtime uses to set up the container. This is a part of manifest schema v2. Will not be set when the image represents a manifest list.
dockerImageLayers	**array**	DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.
dockerImageLayers[]	**object**	ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.
dockerImageManifest	**string**	DockerImageManifest is the raw JSON of the manifest
dockerImageManifestMediaType	**string**	DockerImageManifestMediaType specifies the mediaType of manifest. This is a part of manifest schema v2.

Property	Type	Description
dockerImageManifests	**array**	DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.
dockerImageManifests[]	**object**	ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.
dockerImageMetadata	**RawExtension**	DockerImageMetadata contains metadata about this image
dockerImageMetadataVersion	**string**	DockerImageMetadataVersion conveys the version of the object, which if empty defaults to "1.0"
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image.
dockerImageSignatures	**array (string)**	DockerImageSignatures provides the signatures as opaque blobs. This is a part of manifest schema v1.
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signatures	**array**	Signatures holds all signatures of the image.

Property	Type	Description
signatures[]	**object**	ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

7.1.2. .image.dockerImageLayers

Description

DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.

Type

array

7.1.3. .image.dockerImageLayers[]

Description

ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.

Type

object

Required

- **name**
- **size**
- **mediaType**

Property	Type	Description
mediaType	**string**	MediaType of the referenced object.
name	**string**	Name of the layer as defined by the underlying store.
size	**integer**	Size of the layer in bytes as defined by the underlying store.

7.1.4. .image.dockerImageManifests

Description

DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.

Type

array

7.1.5. .image.dockerImageManifests[]

Description

ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Type

object

Required

- **digest**
- **mediaType**
- **manifestSize**
- **architecture**
- **os**

Property	Type	Description
architecture	**string**	Architecture specifies the supported CPU architecture, for example **amd64** or **ppc64le**.

Property	Type	Description
digest	**string**	Digest is the unique identifier for the manifest. It refers to an Image object.
manifestSize	**integer**	ManifestSize represents the size of the raw object contents, in bytes.
mediaType	**string**	MediaType defines the type of the manifest, possible values are application/vnd.oci.image.manifest.v1+json, application/vnd.docker.distribution.manifest.v2+json or application/vnd.docker.distribution.manifest.v1+json.
os	**string**	OS specifies the operating system, for example **linux**.
variant	**string**	Variant is an optional field repreenting a variant of the CPU, for example v6 to specify a particular CPU variant of the ARM CPU.

7.1.6. .image.signatures

Description

Signatures holds all signatures of the image.

Type

array

7.1.7. .image.signatures[]

Description

ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **type**

- type
- content

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
conditions	array	Conditions represent the latest available observations of a signature's current state.
conditions[]	object	SignatureCondition describes an image signature condition of particular kind at particular probe time.
content	string	Required: An opaque binary string which is an image's signature.
created	Time	If specified, it is the time of signature's creation.
imageIdentity	string	A human readable string representing image's identity. It could be a product name and version, or an image pull spec (e.g. "registry.access.redhat.com/rhel7/rhel:7.2").
issuedBy	object	SignatureIssuer holds information about an issuer of signing certificate or key.
issuedTo	object	SignatureSubject holds information about a person or entity who created the signature.

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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signedClaims	**object (string)**	Contains claims from the signature.
type	**string**	Required: Describes a type of stored blob.

7.1.8. .image.signatures[].conditions

Description

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

Type

array

7.1.9. .image.signatures[].conditions[]

Description

SignatureCondition describes an image signature condition of particular kind at particular probe time.

Type

object

Required

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

Property	Type	Description
lastProbeTime	**Time**	Last time the condition was checked.
lastTransitionTime	**Time**	Last time the condition transit from one status to another.
message	**string**	Human readable message indicating details about last transition.
reason	**string**	(brief) reason for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of signature condition, Complete or Failed.

7.1.10. .image.signatures[].issuedBy

Description

SignatureIssuer holds information about an issuer of signing certificate or key.

Type

object

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.

7.1.11. .image.signatures[].issuedTo

Description

SignatureSubject holds information about a person or entity who created the signature.

Type

object

Required

- **publicKeyID**

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.
publicKeyID	**string**	If present, it is a human readable key id of public key belonging to the subject used to verify image signature. It should contain at least 64 lowest bits of public key's fingerprint (e.g. 0x685ebe62bf278440).

7.2. API ENDPOINTS

The following API endpoints are available:

- **/apis/image.openshift.io/v1/namespaces/{namespace}/imagestreammappings**
  - **POST**: create an ImageStreamMapping

7.2.1. /apis/image.openshift.io/v1/namespaces/{namespace}/imagestreammappings

Table 7.1. Global query parameters

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

Parameter	Type	Description
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP method

POST

Description

create an ImageStreamMapping

Table 7.2. Body parameters

Parameter	Type	Description
body	**ImageStreamMapping** schema	

Table 7.3. HTTP responses

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

CHAPTER 8. IMAGESTREAM [IMAGE.OPENSHIFT.IO/V1]

Description

An ImageStream stores a mapping of tags to images, metadata overrides that are applied when images are tagged in a stream, and an optional reference to a container image repository on a registry. Users typically update the spec.tags field to point to external images which are imported from container registries using credentials in your namespace with the pull secret type, or to existing image stream tags and images which are immediately accessible for tagging or pulling. The history of images applied to a tag is visible in the status.tags field and any user who can view an image stream is allowed to tag that image into their own image streams. Access to pull images from the integrated registry is granted by having the "get imagestreams/layers" permission on a given image stream. Users may remove a tag by deleting the imagestreamtag resource, which causes both spec and status for that tag to be removed. Image stream history is retained until an administrator runs the prune operation, which removes references that are no longer in use. To preserve a historical image, ensure there is a tag in spec pointing to that image by its digest.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

8.1. 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
kind	**string**	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

Property	Type	Description
metadata	**ObjectMeta_v2**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
spec	**object**	ImageStreamSpec represents options for ImageStreams.
status	**object**	ImageStreamStatus contains information about the state of this image stream.

8.1.1. .spec

Description

ImageStreamSpec represents options for ImageStreams.

Type

object

Property	Type	Description
dockerImageRepository	**string**	dockerImageRepository is optional, if specified this stream is backed by a container repository on this server Deprecated: This field is deprecated as of v3.7 and will be removed in a future release. Specify the source for the tags to be imported in each tag via the spec.tags.from reference instead.
lookupPolicy	**object**	ImageLookupPolicy describes how an image stream can be used to override the image references used by pods, builds, and other resources in a namespace.
tags	**array**	tags map arbitrary string values to specific image locators

Property	Type	Description
tags[]	**object**	TagReference specifies optional annotations for images using this tag and an optional reference to an ImageStreamTag, ImageStreamImage, or DockerImage this tag should track.

8.1.2. .spec.lookupPolicy

Description

ImageLookupPolicy describes how an image stream can be used to override the image references used by pods, builds, and other resources in a namespace.

Type

object

Required

- **local**

Property	Type	Description
local	**boolean**	local will change the docker short image references (like "mysql" or "php:latest") on objects in this namespace to the image ID whenever they match this image stream, instead of reaching out to a remote registry. The name will be fully qualified to an image ID if found. The tag's referencePolicy is taken into account on the replaced value. Only works within the current namespace.

8.1.3. .spec.tags

Description

tags map arbitrary string values to specific image locators

Type

array

8.1.4. .spec.tags[]

Description

TagReference specifies optional annotations for images using this tag and an optional reference to an ImageStreamTag, ImageStreamImage, or DockerImage this tag should track.

Type

object

Required

- **name**

Property	Type	Description
annotations	**object (string)**	Optional; if specified, annotations that are applied to images retrieved via ImageStreamTags.
from	**ObjectReference**	Optional; if specified, a reference to another image that this tag should point to. Valid values are ImageStreamTag, ImageStreamImage, and DockerImage. ImageStreamTag references can only reference a tag within this same ImageStream.
generation	**integer**	Generation is a counter that tracks mutations to the spec tag (user intent). When a tag reference is changed the generation is set to match the current stream generation (which is incremented every time spec is changed). Other processes in the system like the image importer observe that the generation of spec tag is newer than the generation recorded in the status and use that as a trigger to import the newest remote tag. To trigger a new import, clients may set this value to zero which will reset the generation to the latest stream generation. Legacy clients will send this value as nil which will be merged with the current tag generation.
importPolicy	**object**	TagImportPolicy controls how images related to this tag will be imported.
name	**string**	Name of the tag

Property	Type	Description
reference	**boolean**	Reference states if the tag will be imported. Default value is false, which means the tag will be imported.
referencePolicy	**object**	TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.

8.1.5. .spec.tags[].importPolicy

Description

TagImportPolicy controls how images related to this tag will be imported.

Type

object

Property	Type	Description
importMode	**string**	ImportMode describes how to import an image manifest.
insecure	**boolean**	Insecure is true if the server may bypass certificate verification or connect directly over HTTP during image import.
scheduled	**boolean**	Scheduled indicates to the server that this tag should be periodically checked to ensure it is up to date, and imported

8.1.6. .spec.tags[].referencePolicy

Description

TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.

Type

object

Required

- **type**

Property	Type	Description
type	**string**	Type determines how the image pull spec should be transformed when the image stream tag is used in deployment config triggers or new builds. The default value is **Source**, indicating the original location of the image should be used (if imported). The user may also specify **Local**, indicating that the pull spec should point to the integrated container image registry and leverage the registry's ability to proxy the pull to an upstream registry. **Local** allows the credentials used to pull this image to be managed from the image stream's namespace, so others on the platform can access a remote image but have no access to the remote secret. It also allows the image layers to be mirrored into the local registry which the images can still be pulled even if the upstream registry is unavailable.

8.1.7. .status

Description

ImageStreamStatus contains information about the state of this image stream.

Type

object

Required

- **dockerImageRepository**

Property	Type	Description

Property	Type	Description
dockerImageRepository	**string**	DockerImageRepository represents the effective location this stream may be accessed at. May be empty until the server determines where the repository is located
publicDockerImageRepository	**string**	PublicDockerImageRepository represents the public location from where the image can be pulled outside the cluster. This field may be empty if the administrator has not exposed the integrated registry externally.
tags	**array**	Tags are a historical record of images associated with each tag. The first entry in the TagEvent array is the currently tagged image.
tags[]	**object**	NamedTagEventList relates a tag to its image history.

8.1.8. .status.tags

Description

Tags are a historical record of images associated with each tag. The first entry in the TagEvent array is the currently tagged image.

Type

array

8.1.9. .status.tags[]

Description

NamedTagEventList relates a tag to its image history.

Type

object

Required

- **tag**
- **items**

Property	Type	Description
conditions	**array**	Conditions is an array of conditions that apply to the tag event list.
conditions[]	**object**	TagEventCondition contains condition information for a tag event.
items	**array**	Standard object's metadata.
items[]	**object**	TagEvent is used by ImageStreamStatus to keep a historical record of images associated with a tag.
tag	**string**	Tag is the tag for which the history is recorded

8.1.10. .status.tags[].conditions

Description

Conditions is an array of conditions that apply to the tag event list.

Type

array

8.1.11. .status.tags[].conditions[]

Description

TagEventCondition contains condition information for a tag event.

Type

object

Required

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

Property	Type	Description
generation	**integer**	Generation is the spec tag generation that this status corresponds to

Property	Type	Description
lastTransitionTime	Time	LastTransitionTIme is the time the condition transitioned from one status to another.
message	**string**	Message is a human readable description of the details about last transition, complementing reason.
reason	**string**	Reason is a brief machine readable explanation for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of tag event condition, currently only ImportSuccess

8.1.12. .status.tags[].items

Description

Standard object's metadata.

Type

array

8.1.13. .status.tags[].items[]

Description

TagEvent is used by ImageStreamStatus to keep a historical record of images associated with a tag.

Type

object

Required

- **created**
- **dockerImageReference**
- **image**
- **generation**

Property	Type	Description
created	Time	Created holds the time the TagEvent was created
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image
generation	**integer**	Generation is the spec tag generation that resulted in this tag being updated
image	**string**	Image is the image

8.2. API ENDPOINTS

The following API endpoints are available:

- **/apis/image.openshift.io/v1/imagestreams**
  - **GET**: list or watch objects of kind ImageStream
- **/apis/image.openshift.io/v1/watch/imagestreams**
  - **GET**: watch individual changes to a list of ImageStream. deprecated: use the 'watch' parameter with a list operation instead.
- **/apis/image.openshift.io/v1/namespaces/{namespace}/imagestreams**
  - **DELETE**: delete collection of ImageStream
  - **GET**: list or watch objects of kind ImageStream
  - **POST**: create an ImageStream
- **/apis/image.openshift.io/v1/watch/namespaces/{namespace}/imagestreams**
  - **GET**: watch individual changes to a list of ImageStream. deprecated: use the 'watch' parameter with a list operation instead.
- **/apis/image.openshift.io/v1/namespaces/{namespace}/imagestreams/{name}**
  - **DELETE**: delete an ImageStream
  - **GET**: read the specified ImageStream
  - **PATCH**: partially update the specified ImageStream
  - **PUT**: replace the specified ImageStream
- **/apis/image.openshift.io/v1/watch/namespaces/{namespace}/imagestreams/{name}**

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

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

8.2.1. /apis/image.openshift.io/v1/imagestreams

HTTP method

GET

Description

list or watch objects of kind ImageStream

Table 8.1. HTTP responses

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

8.2.2. /apis/image.openshift.io/v1/watch/imagestreams

HTTP method

GET

Description

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

Table 8.2. HTTP responses

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

8.2.3. /apis/image.openshift.io/v1/namespaces/{namespace}/imagestreams

HTTP method

DELETE

Description

delete collection of ImageStream

Table 8.3. 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

Table 8.4. HTTP responses

HTTP code	Reponse body
200 - OK	**Status_v5** schema
401 - Unauthorized	Empty

HTTP method

GET

Description

list or watch objects of kind ImageStream

Table 8.5. HTTP responses

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

HTTP method

POST

Description

create an ImageStream

Table 8.6. 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

Parameter	Type	Description
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Table 8.7. Body parameters

Parameter	Type	Description
body	**ImageStream** schema	

Table 8.8. HTTP responses

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

8.2.4. /apis/image.openshift.io/v1/watch/namespaces/{namespace}/imagestreams

HTTP method

GET

Description

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

Table 8.9. HTTP responses

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

8.2.5. /apis/image.openshift.io/v1/namespaces/{namespace}/imagestreams/{name}

Table 8.10. Global path parameters

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

HTTP method

DELETE

Description

delete an ImageStream

Table 8.11. 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

Table 8.12. HTTP responses

HTTP code	Reponse body
200 - OK	**Status_v5** schema
202 - Accepted	**Status_v5** schema
401 - Unauthorized	Empty

HTTP method

GET

Description

read the specified ImageStream

Table 8.13. HTTP responses

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

HTTP method

PATCH

Description

partially update the specified ImageStream

Table 8.14. Query parameters

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

Table 8.15. HTTP responses

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

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

HTTP method

PUT

Description

replace the specified ImageStream

Table 8.16. Query parameters

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

Table 8.17. Body parameters

Parameter	Type	Description
body	**ImageStream** schema	

Table 8.18. HTTP responses

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

8.2.6. /apis/image.openshift.io/v1/watch/namespaces/{namespace}/imagestreams/{

Table 8.19. Global path parameters

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

HTTP method

GET

Description

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

Table 8.20. HTTP responses

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

8.2.7. /apis/image.openshift.io/v1/namespaces/{namespace}/imagestreams/{name}/

Table 8.21. Global path parameters

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

HTTP method

GET

Description

read status of the specified ImageStream

Table 8.22. HTTP responses

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

HTTP method

PATCH

Description

partially update status of the specified ImageStream

Table 8.23. Query parameters

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

Table 8.24. HTTP responses

HTTP code	Reponse body
200 - OK	**ImageStream** schema
201 - Created	**ImageStream** schema

HTTP code	Reponse body
401 - Unauthorized	Empty

HTTP method

PUT

Description

replace status of the specified ImageStream

Table 8.25. Query parameters

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

Table 8.26. Body parameters

Parameter	Type	Description
body	**ImageStream** schema	

Table 8.27. HTTP responses

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

CHAPTER 9. IMAGESTREAMTAG [IMAGE.OPENSHIFT.IO/V1]

Description

ImageStreamTag represents an Image that is retrieved by tag name from an ImageStream. Use this resource to interact with the tags and images in an image stream by tag, or to see the image details for a particular tag. The image associated with this resource is the most recently successfully tagged, imported, or pushed image (as described in the image stream status.tags.items list for this tag). If an import is in progress or has failed the previous image will be shown. Deleting an image stream tag clears both the status and spec fields of an image stream. If no image can be retrieved for a given tag, a not found error will be returned.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **tag**
- **generation**
- **lookupPolicy**
- **image**

9.1. 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
conditions	**array**	conditions is an array of conditions that apply to the image stream tag.
conditions[]	**object**	TagEventCondition contains condition information for a tag event.

Property	Type	Description
generation	**integer**	generation is the current generation of the tagged image - if tag is provided and this value is not equal to the tag generation, a user has requested an import that has not completed, or conditions will be filled out indicating any error.
image	**object**	Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).
kind	**string**	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

Property	Type	Description
lookupPolicy	**object**	ImageLookupPolicy describes how an image stream can be used to override the image references used by pods, builds, and other resources in a namespace.
metadata	**ObjectMeta_v2**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
tag	**object**	TagReference specifies optional annotations for images using this tag and an optional reference to an ImageStreamTag, ImageStreamImage, or DockerImage this tag should track.

9.1.1. .conditions

Description

conditions is an array of conditions that apply to the image stream tag.

Type

array

9.1.2. .conditions[]

Description

TagEventCondition contains condition information for a tag event.

Type

object

Required

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

Property	Type	Description
generation	**integer**	Generation is the spec tag generation that this status corresponds to
lastTransitionTime	Time	LastTransitionTIme is the time the condition transitioned from one status to another.
message	**string**	Message is a human readable description of the details about last transition, complementing reason.
reason	**string**	Reason is a brief machine readable explanation for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of tag event condition, currently only ImportSuccess

9.1.3. .image

Description

Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents.

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

Type

object

Property	Type	Description

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
dockerImageConfig	**string**	DockerImageConfig is a JSON blob that the runtime uses to set up the container. This is a part of manifest schema v2. Will not be set when the image represents a manifest list.
dockerImageLayers	**array**	DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.
dockerImageLayers[]	**object**	ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.
dockerImageManifest	**string**	DockerImageManifest is the raw JSON of the manifest
dockerImageManifestMediaType	**string**	DockerImageManifestMediaType specifies the mediaType of manifest. This is a part of manifest schema v2.
dockerImageManifests	**array**	DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.
dockerImageManifests[]	**object**	ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Property	Type	Description
dockerImageMetadata	**RawExtension**	DockerImageMetadata contains metadata about this image
dockerImageMetadataVersion	**string**	DockerImageMetadataVersion conveys the version of the object, which if empty defaults to "1.0"
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image.
dockerImageSignatures	**array (string)**	DockerImageSignatures provides the signatures as opaque blobs. This is a part of manifest schema v1.
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signatures	**array**	Signatures holds all signatures of the image.

Property	Type	Description
signatures[]	**object**	ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

9.1.4. .image.dockerImageLayers

Description

DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.

Type

array

9.1.5. .image.dockerImageLayers[]

Description

ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.

Type

object

Required

- **name**
- **size**
- **mediaType**

Property	Type	Description
mediaType	**string**	MediaType of the referenced object.

Property	Type	Description
name	**string**	Name of the layer as defined by the underlying store.
size	**integer**	Size of the layer in bytes as defined by the underlying store.

9.1.6. .image.dockerImageManifests

Description

DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.

Type

array

9.1.7. .image.dockerImageManifests[]

Description

ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Type

object

Required

- **digest**
- **mediaType**
- **manifestSize**
- **architecture**
- **os**

Property	Type	Description
architecture	**string**	Architecture specifies the supported CPU architecture, for example **amd64** or **ppc64le**.
digest	**string**	Digest is the unique identifier for the manifest. It refers to an Image object.

Property	Type	Description
manifestSize	**integer**	ManifestSize represents the size of the raw object contents, in bytes.
mediaType	**string**	MediaType defines the type of the manifest, possible values are application/vnd.oci.image.manifest.v1+json, application/vnd.docker.distribution.manifest.v2+json or application/vnd.docker.distribution.manifest.v1+json.
os	**string**	OS specifies the operating system, for example **linux**.
variant	**string**	Variant is an optional field repreenting a variant of the CPU, for example v6 to specify a particular CPU variant of the ARM CPU.

9.1.8. .image.signatures

Description

Signatures holds all signatures of the image.

Type

array

9.1.9. .image.signatures[]

Description

ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **type**
- **content**

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
conditions	**array**	Conditions represent the latest available observations of a signature's current state.
conditions[]	**object**	SignatureCondition describes an image signature condition of particular kind at particular probe time.
content	**string**	Required: An opaque binary string which is an image's signature.
created	**Time**	If specified, it is the time of signature's creation.
imageIdentity	**string**	A human readable string representing image's identity. It could be a product name and version, or an image pull spec (e.g. "registry.access.redhat.com/rhel7/rhel:7.2").
issuedBy	**object**	SignatureIssuer holds information about an issuer of signing certificate or key.
issuedTo	**object**	SignatureSubject holds information about a person or entity who created the signature.

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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signedClaims	**object (string)**	Contains claims from the signature.
type	**string**	Required: Describes a type of stored blob.

9.1.10. .image.signatures[].conditions

Description

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

Type

array

9.1.11. .image.signatures[].conditions[]

Description

SignatureCondition describes an image signature condition of particular kind at particular probe time.

Type

object

Required

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

Property	Type	Description
lastProbeTime	**Time**	Last time the condition was checked.
lastTransitionTime	**Time**	Last time the condition transit from one status to another.
message	**string**	Human readable message indicating details about last transition.
reason	**string**	(brief) reason for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of signature condition, Complete or Failed.

9.1.12. .image.signatures[].issuedBy

Description

SignatureIssuer holds information about an issuer of signing certificate or key.

Type

object

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.

9.1.13. .image.signatures[].issuedTo

Description

SignatureSubject holds information about a person or entity who created the signature.

Type

object

Required

- **publicKeyID**

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.
publicKeyID	**string**	If present, it is a human readable key id of public key belonging to the subject used to verify image signature. It should contain at least 64 lowest bits of public key's fingerprint (e.g. 0x685ebe62bf278440).

9.1.14. .lookupPolicy

Description

ImageLookupPolicy describes how an image stream can be used to override the image references used by pods, builds, and other resources in a namespace.

Type

object

Required

- **local**

Property	Type	Description
local	**boolean**	local will change the docker short image references (like "mysql" or "php:latest") on objects in this namespace to the image ID whenever they match this image stream, instead of reaching out to a remote registry. The name will be fully qualified to an image ID if found. The tag's referencePolicy is taken into account on the replaced value. Only works within the current namespace.

9.1.15. .tag

Description

TagReference specifies optional annotations for images using this tag and an optional reference to an ImageStreamTag, ImageStreamImage, or DockerImage this tag should track.

Type

object

Required

- **name**

Property	Type	Description
annotations	**object (string)**	Optional; if specified, annotations that are applied to images retrieved via ImageStreamTags.
from	**ObjectReference**	Optional; if specified, a reference to another image that this tag should point to. Valid values are ImageStreamTag, ImageStreamImage, and DockerImage. ImageStreamTag references can only reference a tag within this same ImageStream.
generation	**integer**	Generation is a counter that tracks mutations to the spec tag (user intent). When a tag reference is changed the generation is set to match the current stream generation (which is incremented every time spec is changed). Other processes in the system like the image importer observe that the generation of spec tag is newer than the generation recorded in the status and use that as a trigger to import the newest remote tag. To trigger a new import, clients may set this value to zero which will reset the generation to the latest stream generation. Legacy clients will send this value as nil which will be merged with the current tag generation.
importPolicy	**object**	TagImportPolicy controls how images related to this tag will be imported.
name	**string**	Name of the tag
reference	**boolean**	Reference states if the tag will be imported. Default value is false, which means the tag will be imported.

Property	Type	Description
referencePolicy	**object**	TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.

9.1.16. .tag.importPolicy

Description

TagImportPolicy controls how images related to this tag will be imported.

Type

object

Property	Type	Description
importMode	**string**	ImportMode describes how to import an image manifest.
insecure	**boolean**	Insecure is true if the server may bypass certificate verification or connect directly over HTTP during image import.
scheduled	**boolean**	Scheduled indicates to the server that this tag should be periodically checked to ensure it is up to date, and imported

9.1.17. .tag.referencePolicy

Description

TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.

Type

object

Required

- **type**

Property	Type	Description
type	**string**	Type determines how the image pull spec should be transformed when the image stream tag is used in deployment config triggers or new builds. The default value is **Source**, indicating the original location of the image should be used (if imported). The user may also specify **Local**, indicating that the pull spec should point to the integrated container image registry and leverage the registry's ability to proxy the pull to an upstream registry. **Local** allows the credentials used to pull this image to be managed from the image stream's namespace, so others on the platform can access a remote image but have no access to the remote secret. It also allows the image layers to be mirrored into the local registry which the images can still be pulled even if the upstream registry is unavailable.

9.2. API ENDPOINTS

The following API endpoints are available:

- **/apis/image.openshift.io/v1/imagestreamtags**
  - **GET**: list objects of kind ImageStreamTag
- **/apis/image.openshift.io/v1/namespaces/{namespace}/imagestreamtags**
  - **GET**: list objects of kind ImageStreamTag
  - **POST**: create an ImageStreamTag
- **/apis/image.openshift.io/v1/namespaces/{namespace}/imagestreamtags/{name}**
  - **DELETE**: delete an ImageStreamTag
  - **GET**: read the specified ImageStreamTag
  - **PATCH**: partially update the specified ImageStreamTag
  - **PUT**: replace the specified ImageStreamTag

9.2.1. /apis/image.openshift.io/v1/imagestreamtags

HTTP method

GET

Description

list objects of kind ImageStreamTag

Table 9.1. HTTP responses

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

9.2.2. /apis/image.openshift.io/v1/namespaces/{namespace}/imagestreamtags

HTTP method

GET

Description

list objects of kind ImageStreamTag

Table 9.2. HTTP responses

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

HTTP method

POST

Description

create an ImageStreamTag

Table 9.3. Query parameters

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

Parameter	Type	Description
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Table 9.4. Body parameters

Parameter	Type	Description
body	**ImageStreamTag** schema	

Table 9.5. HTTP responses

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

9.2.3. /apis/image.openshift.io/v1/namespaces/{namespace}/imagestreamtags/{nam

Table 9.6. Global path parameters

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

HTTP method

DELETE

Description

delete an ImageStreamTag

Table 9.7. 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

Table 9.8. HTTP responses

HTTP code	Reponse body
200 - OK	**Status_v5** schema
202 - Accepted	**Status_v5** schema
401 - Unauthorized	Empty

HTTP method

GET

Description

read the specified ImageStreamTag

Table 9.9. HTTP responses

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

HTTP method

PATCH

Description

partially update the specified ImageStreamTag

Table 9.10. Query parameters

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

Table 9.11. HTTP responses

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

HTTP method

PUT

Description

replace the specified ImageStreamTag

Table 9.12. 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
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Table 9.13. Body parameters

Parameter	Type	Description
body	**ImageStreamTag** schema	

Table 9.14. HTTP responses

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

CHAPTER 10. IMAGETAG [IMAGE.OPENSHIFT.IO/V1]

Description

ImageTag represents a single tag within an image stream and includes the spec, the status history, and the currently referenced image (if any) of the provided tag. This type replaces the ImageStreamTag by providing a full view of the tag. ImageTags are returned for every spec or status tag present on the image stream. If no tag exists in either form a not found error will be returned by the API. A create operation will succeed if no spec tag has already been defined and the spec field is set. Delete will remove both spec and status elements from the image stream.
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**
- **status**
- **image**

10.1. 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
image	**object**	Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata

Property	Type	Description
spec	**object**	TagReference specifies optional annotations for images using this tag and an optional reference to an ImageStreamTag, ImageStreamImage, or DockerImage this tag should track.
status	**object**	NamedTagEventList relates a tag to its image history.

10.1.1. .image

Description

Image is an immutable representation of a container image and metadata at a point in time. Images are named by taking a hash of their contents (metadata and content) and any change in format, content, or metadata results in a new name. The images resource is primarily for use by cluster administrators and integrations like the cluster image registry - end users instead access images via the imagestreamtags or imagestreamimages resources. While image metadata is stored in the API, any integration that implements the container image registry API must provide its own storage for the raw manifest data, image config, and layer contents.

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

Type

object

Property	Type	Description
apiVersion	**string**	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources
dockerImageConfig	**string**	DockerImageConfig is a JSON blob that the runtime uses to set up the container. This is a part of manifest schema v2. Will not be set when the image represents a manifest list.

Property	Type	Description
dockerImageLayers	**array**	DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.
dockerImageLayers[]	**object**	ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.
dockerImageManifest	**string**	DockerImageManifest is the raw JSON of the manifest
dockerImageManifestMediaType	**string**	DockerImageManifestMediaType specifies the mediaType of manifest. This is a part of manifest schema v2.
dockerImageManifests	**array**	DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.
dockerImageManifests[]	**object**	ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.
dockerImageMetadata	**RawExtension**	DockerImageMetadata contains metadata about this image
dockerImageMetadataVersion	**string**	DockerImageMetadataVersion conveys the version of the object, which if empty defaults to "1.0"
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image.
dockerImageSignatures	**array (string)**	DockerImageSignatures provides the signatures as opaque blobs. This is a part of manifest schema v1.

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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signatures	**array**	Signatures holds all signatures of the image.
signatures[]	**object**	ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing image verification. The others are parsed from signature's content by the server. They serve just an informative purpose. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

10.1.2. .image.dockerImageLayers

Description

DockerImageLayers represents the layers in the image. May not be set if the image does not define that data or if the image represents a manifest list.

Type

array

10.1.3. .image.dockerImageLayers[]

Description

ImageLayer represents a single layer of the image. Some images may have multiple layers. Some may have none.

Type

object

Required

- **name**
- **size**
- **mediaType**

Property	Type	Description
mediaType	**string**	MediaType of the referenced object.
name	**string**	Name of the layer as defined by the underlying store.
size	**integer**	Size of the layer in bytes as defined by the underlying store.

10.1.4. .image.dockerImageManifests

Description

DockerImageManifests holds information about sub-manifests when the image represents a manifest list. When this field is present, no DockerImageLayers should be specified.

Type

array

10.1.5. .image.dockerImageManifests[]

Description

ImageManifest represents sub-manifests of a manifest list. The Digest field points to a regular Image object.

Type

object

Required

- **digest**
- **mediaType**
- **manifestSize**

- **architecture**
- **os**

Property	Type	Description
architecture	**string**	Architecture specifies the supported CPU architecture, for example **amd64** or **ppc64le**.
digest	**string**	Digest is the unique identifier for the manifest. It refers to an Image object.
manifestSize	**integer**	ManifestSize represents the size of the raw object contents, in bytes.
mediaType	**string**	MediaType defines the type of the manifest, possible values are application/vnd.oci.image.manifest.v1+json, application/vnd.docker.distribution.manifest.v2+json or application/vnd.docker.distribution.manifest.v1+json.
os	**string**	OS specifies the operating system, for example **linux**.
variant	**string**	Variant is an optional field repreenting a variant of the CPU, for example v6 to specify a particular CPU variant of the ARM CPU.

10.1.6. .image.signatures

Description

Signatures holds all signatures of the image.

Type

array

10.1.7. .image.signatures[]

Description

ImageSignature holds a signature of an image. It allows to verify image identity and possibly other claims as long as the signature is trusted. Based on this information it is possible to restrict runnable images to those matching cluster-wide policy. Mandatory fields should be parsed by clients doing

image verification. The others are parsed from signature's content by the server. They serve just an informative purpose.
Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

Type

object

Required

- **type**
- **content**

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
conditions	**array**	Conditions represent the latest available observations of a signature's current state.
conditions[]	**object**	SignatureCondition describes an image signature condition of particular kind at particular probe time.
content	**string**	Required: An opaque binary string which is an image's signature.
created	**Time**	If specified, it is the time of signature's creation.
imageIdentity	**string**	A human readable string representing image's identity. It could be a product name and version, or an image pull spec (e.g. "registry.access.redhat.com/rhel7/rhel:7.2").
issuedBy	**object**	SignatureIssuer holds information about an issuer of signing certificate or key.

Property	Type	Description
issuedTo	**object**	SignatureSubject holds information about a person or entity who created the signature.
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**	metadata is the standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata
signedClaims	**object (string)**	Contains claims from the signature.
type	**string**	Required: Describes a type of stored blob.

10.1.8. .image.signatures[].conditions

Description

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

Type

array

10.1.9. .image.signatures[].conditions[]

Description

SignatureCondition describes an image signature condition of particular kind at particular probe time.

Type

object

Required

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

Property	Type	Description
lastProbeTime	**Time**	Last time the condition was checked.
lastTransitionTime	**Time**	Last time the condition transit from one status to another.
message	**string**	Human readable message indicating details about last transition.
reason	**string**	(brief) reason for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of signature condition, Complete or Failed.

10.1.10. .image.signatures[].issuedBy

Description

SignatureIssuer holds information about an issuer of signing certificate or key.

Type

object

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.

10.1.11. .image.signatures[].issuedTo

Description

SignatureSubject holds information about a person or entity who created the signature.

Type

object

Required

- **publicKeyID**

Property	Type	Description
commonName	**string**	Common name (e.g. openshift-signing-service).
organization	**string**	Organization name.
publicKeyID	**string**	If present, it is a human readable key id of public key belonging to the subject used to verify image signature. It should contain at least 64 lowest bits of public key's fingerprint (e.g. 0x685ebe62bf278440).

10.1.12. .spec

Description

TagReference specifies optional annotations for images using this tag and an optional reference to an ImageStreamTag, ImageStreamImage, or DockerImage this tag should track.

Type

object

Required

- **name**

Property	Type	Description
annotations	**object (string)**	Optional; if specified, annotations that are applied to images retrieved via ImageStreamTags.
from	**ObjectReference**	Optional; if specified, a reference to another image that this tag should point to. Valid values are ImageStreamTag, ImageStreamImage, and DockerImage. ImageStreamTag references can only reference a tag within this same ImageStream.

Property	Type	Description
generation	**integer**	Generation is a counter that tracks mutations to the spec tag (user intent). When a tag reference is changed the generation is set to match the current stream generation (which is incremented every time spec is changed). Other processes in the system like the image importer observe that the generation of spec tag is newer than the generation recorded in the status and use that as a trigger to import the newest remote tag. To trigger a new import, clients may set this value to zero which will reset the generation to the latest stream generation. Legacy clients will send this value as nil which will be merged with the current tag generation.
importPolicy	**object**	TagImportPolicy controls how images related to this tag will be imported.
name	**string**	Name of the tag
reference	**boolean**	Reference states if the tag will be imported. Default value is false, which means the tag will be imported.
referencePolicy	**object**	TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.

10.1.13. .spec.importPolicy

Description

TagImportPolicy controls how images related to this tag will be imported.

Type

object

Property	Type	Description
importMode	**string**	ImportMode describes how to import an image manifest.
insecure	**boolean**	Insecure is true if the server may bypass certificate verification or connect directly over HTTP during image import.
scheduled	**boolean**	Scheduled indicates to the server that this tag should be periodically checked to ensure it is up to date, and imported

10.1.14. .spec.referencePolicy

Description

TagReferencePolicy describes how pull-specs for images in this image stream tag are generated when image change triggers in deployment configs or builds are resolved. This allows the image stream author to control how images are accessed.

Type

object

Required

- **type**

Property	Type	Description

Property	Type	Description
type	**string**	Type determines how the image pull spec should be transformed when the image stream tag is used in deployment config triggers or new builds. The default value is **Source**, indicating the original location of the image should be used (if imported). The user may also specify **Local**, indicating that the pull spec should point to the integrated container image registry and leverage the registry's ability to proxy the pull to an upstream registry. **Local** allows the credentials used to pull this image to be managed from the image stream's namespace, so others on the platform can access a remote image but have no access to the remote secret. It also allows the image layers to be mirrored into the local registry which the images can still be pulled even if the upstream registry is unavailable.

10.1.15. .status

Description

NamedTagEventList relates a tag to its image history.

Type

object

Required

- **tag**
- **items**

Property	Type	Description
conditions	**array**	Conditions is an array of conditions that apply to the tag event list.

Property	Type	Description
conditions[]	**object**	TagEventCondition contains condition information for a tag event.
items	**array**	Standard object's metadata.
items[]	**object**	TagEvent is used by ImageStreamStatus to keep a historical record of images associated with a tag.
tag	**string**	Tag is the tag for which the history is recorded

10.1.16. .status.conditions

Description

Conditions is an array of conditions that apply to the tag event list.

Type

array

10.1.17. .status.conditions[]

Description

TagEventCondition contains condition information for a tag event.

Type

object

Required

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

Property	Type	Description
generation	**integer**	Generation is the spec tag generation that this status corresponds to
lastTransitionTime	Time	LastTransitionTIme is the time the condition transitioned from one status to another.

Property	Type	Description
message	**string**	Message is a human readable description of the details about last transition, complementing reason.
reason	**string**	Reason is a brief machine readable explanation for the condition's last transition.
status	**string**	Status of the condition, one of True, False, Unknown.
type	**string**	Type of tag event condition, currently only ImportSuccess

10.1.18. .status.items

Description

Standard object's metadata.

Type

array

10.1.19. .status.items[]

Description

TagEvent is used by ImageStreamStatus to keep a historical record of images associated with a tag.

Type

object

Required

- **created**
- **dockerImageReference**
- **image**
- **generation**

Property	Type	Description
created	Time	Created holds the time the TagEvent was created

Property	Type	Description
dockerImageReference	**string**	DockerImageReference is the string that can be used to pull this image
generation	**integer**	Generation is the spec tag generation that resulted in this tag being updated
image	**string**	Image is the image

10.2. API ENDPOINTS

The following API endpoints are available:

- **/apis/image.openshift.io/v1/imagetags**
  - **GET**: list objects of kind ImageTag
- **/apis/image.openshift.io/v1/namespaces/{namespace}/imagetags**
  - **GET**: list objects of kind ImageTag
  - **POST**: create an ImageTag
- **/apis/image.openshift.io/v1/namespaces/{namespace}/imagetags/{name}**
  - **DELETE**: delete an ImageTag
  - **GET**: read the specified ImageTag
  - **PATCH**: partially update the specified ImageTag
  - **PUT**: replace the specified ImageTag

10.2.1. /apis/image.openshift.io/v1/imagetags

HTTP method

GET

Description

list objects of kind ImageTag

Table 10.1. HTTP responses

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

HTTP code	Reponse body
401 - Unauthorized	Empty

10.2.2. /apis/image.openshift.io/v1/namespaces/{namespace}/imagetags

HTTP method

GET

Description

list objects of kind ImageTag

Table 10.2. HTTP responses

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

HTTP method

POST

Description

create an ImageTag

Table 10.3. Query parameters

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

Parameter	Type	Description
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Table 10.4. Body parameters

Parameter	Type	Description
body	**ImageTag** schema	

Table 10.5. HTTP responses

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

10.2.3. /apis/image.openshift.io/v1/namespaces/{namespace}/imagetags/{name}

Table 10.6. Global path parameters

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

HTTP method

DELETE

Description

delete an ImageTag

Table 10.7. 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

Table 10.8. HTTP responses

HTTP code	Reponse body
200 - OK	**Status_v5** schema
202 - Accepted	**Status_v5** schema
401 - Unauthorized	Empty

HTTP method

GET

Description

read the specified ImageTag

Table 10.9. HTTP responses

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

HTTP method

PATCH

Description

partially update the specified ImageTag

Table 10.10. Query parameters

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

Table 10.11. HTTP responses

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

HTTP method

PUT

Description

replace the specified ImageTag

Table 10.12. 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
fieldValidation	**string**	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Table 10.13. Body parameters

Parameter	Type	Description
body	**ImageTag** schema	

Table 10.14. HTTP responses

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

CHAPTER 11. SECRETLIST [IMAGE.OPENSHIFT.IO/V1]

Description

SecretList is a list of Secret.

Type

object

Required

- **items**

11.1. 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
items	**array (Secret)**	Items is a list of secret objects. More info: https://kubernetes.io/docs/concepts/configuration/secret
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	**ListMeta**	Standard list metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

11.2. API ENDPOINTS

The following API endpoints are available:

- **/apis/image.openshift.io/v1/namespaces/{namespace}/imagestreams/{name}/secrets**
  - **GET**: read secrets of the specified ImageStream

11.2.1. /apis/image.openshift.io/v1/namespaces/{namespace}/imagestreams/{name}/

Table 11.1. Global path parameters

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

HTTP method

GET

Description

read secrets of the specified ImageStream

Table 11.2. HTTP responses

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