Manage NVMe namespaces

REST API reference

NetApp
September 09, 2025

Table of Contents

Manage NVMe namespaces

Storage namespaces endpoint overview

Overview

An NVMe namespace is a collection of addressable logical blocks presented to hosts connected to the storage virtual machine using the NVMe over Fabrics protocol.

The NVMe namespace REST API allows you to create, update, delete and discover NVMe namespaces.

An NVMe namespace must be mapped to an NVMe subsystem to grant access to the subsystem's hosts. Hosts can then access the NVMe namespace and perform I/O using the NVMe over Fabrics protocol.

See the NVMe namespace object model to learn more about each of the properties supported by the NVMe namespace REST API. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume.

NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional.

An NVMe namespace is created to a specified size using thin or thick provisioning as determined by the volume on which it is created. An NVMe namespace can then be resized or cloned. An NVMe namespace cannot be renamed, or moved to a different volume. NVMe namespaces do not support the assignment of a QoS policy for performance management, but a QoS policy can be assigned to the volume containing the namespace.

Performance monitoring

Performance of an NVMe namespace can be monitored by observing the `metric.*` and `statistics.*` properties. These properties show the space utilization and performance of an NVMe namespace in terms of IOPS, latency, and throughput. The `metric.*` properties denote an average, whereas `statistics.*` properties denote a real-time monotonically increasing value aggregated across all nodes.

Examples

Creating an NVMe namespace

This example creates a 300 gigabyte NVMe namespace, with 4096-byte blocks, in SVM *svm1*, volume *vol1*, configured for use by *linux* hosts. The `return_records` query parameter is used to retrieve properties of the newly created NVMe namespace in the POST response.

```
# The API:
POST /api/storage/namespaces

# The call:
curl -X POST 'https://<mgmt-
ip>/api/storage/namespaces?return_records=true' -H 'Accept:
application/hal+json' -d '{ "svm": { "name": "svm1" }, "os_type": "linux",
"space": { "block_size": "4096", "size": "300G" }, "name" :
```

```
"/vol/vol1/namespace1" }'

# The response:
{
"num_records": 1,
"records": [
  {
    "uuid": "dccdc3e6-cf4e-498f-bec6-f7897f945669",
    "svm": {
      "uuid": "6bf967fd-2a1c-11e9-b682-005056bbc17d",
      "name": "svm1",
      "_links": {
        "self": {
          "href": "/api/svm/svms/6bf967fd-2a1c-11e9-b682-005056bbc17d"
        }
      }
    },
    "name": "/vol/vol1/namespace1",
    "location": {
      "namespace": "namespace1",
      "volume": {
        "uuid": "71cd0dba-2a1c-11e9-b682-005056bbc17d",
        "name": "vol1",
        "_links": {
          "self": {
            "href": "/api/storage/volumes/71cd0dba-2a1c-11e9-b682-
005056bbc17d"
          }
        }
      }
    },
    "enabled": true,
    "os_type": "linux",
    "space": {
      "block_size": 4096,
      "size": 322122547200,
      "used": 0,
      "guarantee": {
        "requested": false,
        "reserved": false
      }
    },
    "status": {
      "container_state": "online",
      "read_only": false,
      "state": "online"
```

```
    },
    "_links": {
      "self": {
        "href": "/api/storage/namespaces/dccdc3e6-cf4e-498f-bec6-
f7897f945669"
      }
    }
  }
]
}
```

Updating an NVMe namespace comment

This example sets the `comment` property of an NVMe namespace.

```
# The API:
PATCH /api/storage/namespaces/{uuid}

# The call:
```

Updating the size of an NVMe namespace

This example increases the size of an NVMe namespace.

```
# The API:
PATCH /api/storage/namespaces/{uuid}

# The call:
curl -X PATCH 'https://<mgmt-ip>/api/storage/namespaces/dccdc3e6-cf4e-
498f-bec6-f7897f945669' -H 'Accept: application/hal+json' -d '{ "space": {
"size": "1073741824" } }'
```

Retrieving NVMe namespaces

This example retrieves summary information for all online NVMe namespaces in SVM *svm1*. The `svm.name` and `status.state` query parameters are to find the desired NVMe namespaces.

```
# The API:
GET /api/storage/namespaces

# The call:
```

```
curl -X GET 'https://<mgmt-
ip>/api/storage/namespaces?svm.name=svm1&status.state=online' -H 'Accept:
application/hal+json'

# The response:
{
"records": [
  {
    "uuid": "5c254d22-96a6-42ac-aad8-0cd9ebd126b6",
    "svm": {
      "name": "svm1"
    },
    "name": "/vol/vol1/namespace2",
    "status": {
      "state": "online"
    },
    "_links": {
      "self": {
        "href": "/api/storage/namespaces/5c254d22-96a6-42ac-aad8-
0cd9ebd126b6"
      }
    }
  },
  {
    "uuid": "dccdc3e6-cf4e-498f-bec6-f7897f945669",
    "svm": {
      "name": "svm1"
    },
    "name": "/vol/vol1/namespace1",
    "status": {
      "state": "online"
    },
    "_links": {
      "self": {
        "href": "/api/storage/namespaces/dccdc3e6-cf4e-498f-bec6-
f7897f945669"
      }
    }
  },
  {
    "uuid": "be732687-20cf-47d2-a0e2-2a989d15661d",
    "svm": {
      "name": "svm1"
    },
    "name": "/vol/vol2/namespace3",
    "status": {
```

```
      "state": "online"
    },
    "_links": {
      "self": {
        "href": "/api/storage/namespaces/be732687-20cf-47d2-a0e2-
2a989d15661d"
      }
    }
  }
],
"num_records": 3,
"_links": {
  "self": {
    "href": "/api/storage/namespaces?svm.name=svm1&status.state=online"
  }
}
}
```

Retrieving details for a specific NVMe namespace

In this example, the `fields` query parameter is used to request all fields, including advanced fields, that would not otherwise be returned by default for the NVMe namespace.

```
# The API:
GET /api/storage/namespaces/{uuid}

# The call:
curl -X GET 'https://<mgmt-ip>/api/storage/namespaces/dccdc3e6-cf4e-498f-
bec6-f7897f945669?fields=**' -H 'Accept: application/hal+json'

# The response:
{
"uuid": "dccdc3e6-cf4e-498f-bec6-f7897f945669",
"svm": {
  "uuid": "6bf967fd-2a1c-11e9-b682-005056bbc17d",
  "name": "svm1",
  "_links": {
    "self": {
      "href": "/api/svm/svms/6bf967fd-2a1c-11e9-b682-005056bbc17d"
    }
  }
},
"name": "/vol/vol1/namespace1",
"location": {
```

```
  "namespace": "namespace1",
  "volume": {
    "uuid": "71cd0dba-2a1c-11e9-b682-005056bbc17d",
    "name": "vol1",
    "_links": {
      "self": {
        "href": "/api/storage/volumes/71cd0dba-2a1c-11e9-b682-
005056bbc17d"
      }
    }
  }
},
"auto_delete": false,
"enabled": true,
"comment": "Data for the research department.",
"os_type": "linux",
"space": {
  "block_size": 4096,
  "size": 322122547200,
  "used": 0,
  "guarantee": {
    "requested": false,
    "reserved": false
  }
},
"status": {
  "container_state": "online",
  "mapped": true,
  "read_only": false,
  "state": "online"
},
"subsystem_map": {
  "nsid": "00000001h",
  "anagrpid": "00000001h",
  "subsystem": {
    "uuid": "01f17d05-2be9-11e9-bed2-005056bbc17d",
    "name": "subsystem1",
    "_links": {
      "self": {
        "href": "/api/protocols/nvme/subsystems/01f17d05-2be9-11e9-bed2-
005056bbc17d"
      }
    }
  },
  "_links": {
    "self": {
```

```
      "href": "/api/protocols/nvme/subsystem-maps/dccdc3e6-cf4e-498f-bec6-
f7897f945669/01f17d05-2be9-11e9-bed2-005056bbc17d"
    }
  }
},
"metric": {
  "timestamp": "2019-04-09T05:50:15Z",
  "duration": "PT15S",
  "status": "ok",
  "latency": {
    "other": 0,
    "total": 0,
    "read": 0,
    "write": 0
  },
  "iops": {
    "read": 0,
    "write": 0,
    "other": 0,
    "total": 0
  },
  "throughput": {
    "read": 0,
    "write": 0,
    "total": 0
  }
},
"statistics": {
  "timestamp": "2019-04-09T05:50:42Z",
  "status": "ok",
  "latency_raw": {
    "other": 38298,
    "total": 38298,
    "read": 0,
    "write": 0
  },
  "iops_raw": {
    "read": 0,
    "write": 0,
    "other": 3,
    "total": 3
  },
  "throughput_raw": {
    "read": 0,
    "write": 0,
    "total": 0
```

```
  }
},
"_links": {
  "self": {
    "href": "/api/storage/namespaces/dccdc3e6-cf4e-498f-bec6-
f7897f945669?fields=**"
  }
}
}
```

Cloning NVMe namespaces

A clone of an NVMe namespace is an independent "copy" of the namespace that shares unchanged data blocks with the original. As blocks of the source and clone are modified, unique blocks are written for each. NVMe namespace clones can be created quickly and consume very little space initially. They can be created for the purpose of back-up, or to replicate data for multiple consumers.

An NVMe namespace clone can also be set to auto-delete by setting the `auto_delete` property. If the namespace's volume is configured for automatic deletion, NVMe namespaces that have auto-delete enabled are deleted when a volume is nearly full to reclaim a target amount of free space in the volume.

Creating a new NVMe namespace clone

You create an NVMe namespace clone as you create any NVMe namespace — a POST to /storage/namespaces. Set `clone.source.uuid` or `clone.source.name` to identify the source NVMe namespace from which the clone is created. The NVMe namespace clone and its source must reside in the same volume.

The source NVMe namespace can reside in a snapshot, in which case, the `clone.source.name` field must be used to identify it. Add `/.snapshot/<snapshot_name>` to the path after the volume name to identify the snapshot. For example `/vol/vol1/.snapshot/snap1/namespace1`.

```
# The API:
POST /api/storage/namespaces

# The call:
curl -X POST 'https://<mgmt-ip>/api/storage/namespaces' -H 'Accept:
application/hal+json' -d '{ "svm": { "name": "svm1" }, "name":
"/vol/vol1/namespace2clone1", "clone": { "source": { "name":
"/vol/vol1/namespace2" } } }'
```

Over-writing an existing NVMe namespace's data as a clone of another

You can over-write an existing NVMe namespace as a clone of another. You do this as a PATCH on the NVMe namespace to overwrite — a PATCH to /storage/namespaces/{uuid}. Set the `clone.source.uuid` or

`clone.source.name` property to identify the source NVMe namespace from which the clone data is taken. The NVMe namespace clone and its source must reside in the same volume.

When used in a PATCH, the patched NVMe namespace's data is over-written as a clone of the source and the following properties are preserved from the patched namespace unless otherwise specified as part of the PATCH: `auto_delete`, `subsystem_map`, `status.state`, and `uuid`.

```
# The API:
PATCH /api/storage/namespaces/{uuid}

# The call:
curl -X PATCH 'https://<mgmt-ip>/api/storage/namespaces/dccdc3e6-cf4e-
498f-bec6-f7897f945669' -H 'Accept: application/hal+json' -d '{ "clone": {
"source": { "name": "/vol/vol1/namespace2" } } }'
```

Converting a LUN into an NVMe namespace

An existing LUN can be converted in-place to an NVMe namespace with no modification to the data blocks. In other words, there is no additional copy created for the data blocks. There are certain requirements when converting a LUN to an NVMe namespace. For instance, the LUN should not be mapped to an initiator group, or exist as a protocol endpoint LUN, or in a foreign LUN import relationship. If the LUN exists as a VM volume, it should not be bound to a protocol endpoint LUN. Furthermore, only LUN with a supported operating system type for NVMe namespace can be converted.

The conversion process updates the metadata to the LUN, making it an NVMe namespace. The conversion is both time and space efficient. After conversion, the new namespace behaves as a regular namespace and may be mapped to an NVMe subsystem.

Convert a LUN into an NVMe namespace

You convert a LUN into an NVMe namespace by calling a POST to /storage/namespaces. Set `convert.lun.uuid` or `convert.lun.name` to identify the source LUN which is to be converted in-place into an NVMe namespace.

```
# The API:
POST /api/storage/namespaces

# The call:
curl -X POST 'https://<mgmt-ip>/api/storage/namespaces' -H 'Accept:
application/hal+json' -d '{ "svm": { "name": "svm1" }, "convert": { "lun":
{ "name": "/vol/vol1/lun1" } } }'
```

Deleting an NVMe namespace

```
# The API:
DELETE /api/storage/namespaces/{uuid}

# The call:
curl -X DELETE 'https://<mgmt-ip>/api/storage/namespaces/5c254d22-96a6-
42ac-aad8-0cd9ebd126b6' -H 'Accept: application/hal+json'
```

Retrieve NVMe namespaces

GET `/storage/namespaces`

Introduced In: 9.6

Retrieves NVMe namespaces.

Expensive properties

There is an added computational cost to retrieving values for these properties. They are not included by default in GET results and must be explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.

- `auto_delete`
- `space.physical_used`
- `space.physical_used_by_snapshots`
- `space.efficiency_ratio`
- `subsystem_map.*`
- `status.mapped`
- `statistics.*`
- `metric.*`

Related ONTAP commands

- `vserver nvme namespace show`
- `vserver nvme subsystem map show`

Learn more

- DOC /storage/namespaces to learn more and examples.

Parameters

Name	Type	In	Required	Description
comment	string	query	False	Filter by comment • maxLength: 254 • minLength: 0
subsystem_map.anagrpid	string	query	False	Filter by subsystem_map.anagrpid
subsystem_map.subsystem.os_type	string	query	False	Filter by subsystem_map.subsystem.os_type • Introduced in: 9.16
subsystem_map.subsystem.uuid	string	query	False	Filter by subsystem_map.subsystem.uuid
subsystem_map.subsystem.comment	string	query	False	Filter by subsystem_map.subsystem.comment • maxLength: 255 • minLength: 0 • Introduced in: 9.16
subsystem_map.subsystem.hosts.dh_hmac_chap.group_size	string	query	False	Filter by subsystem_map.subsystem.hosts.dh_hmac_chap.group_size • Introduced in: 9.16
subsystem_map.subsystem.hosts.dh_hmac_chap.mode	string	query	False	Filter by subsystem_map.subsystem.hosts.dh_hmac_chap.mode • Introduced in: 9.16

Name	Type	In	Required	Description
subsystem_map.subsystem.hosts.dh_hmac_chap.hash_function	string	query	False	Filter by subsystem_map.subsystem.hosts.dh_hmac_chap.hash_function • Introduced in: 9.16
subsystem_map.subsystem.hosts.nqn	string	query	False	Filter by subsystem_map.subsystem.hosts.nqn • Introduced in: 9.16
subsystem_map.subsystem.hosts.priority	string	query	False	Filter by subsystem_map.subsystem.hosts.priority • Introduced in: 9.16
subsystem_map.subsystem.hosts.tls.key_type	string	query	False	Filter by subsystem_map.subsystem.hosts.tls.key_type • Introduced in: 9.16
subsystem_map.subsystem.name	string	query	False	Filter by subsystem_map.subsystem.name • maxLength: 64 • minLength: 1
subsystem_map.nsid	string	query	False	Filter by subsystem_map.nsid
os_type	string	query	False	Filter by os_type
auto_delete	boolean	query	False	Filter by auto_delete
status.mapped	boolean	query	False	Filter by status.mapped

Name	Type	In	Required	Description
status.read_only	boolean	query	False	Filter by status.read_only
status.state	string	query	False	Filter by status.state
status.container_state	string	query	False	Filter by status.container_state
uuid	string	query	False	Filter by uuid
space.physical_used	integer	query	False	Filter by space.physical_used • Introduced in: 9.16
space.guarantee.requested	boolean	query	False	Filter by space.guarantee.requested
space.guarantee.reserved	boolean	query	False	Filter by space.guarantee.reserved
space.block_size	integer	query	False	Filter by space.block_size
space.efficiency_ratio	number	query	False	Filter by space.efficiency_ratio • Introduced in: 9.16
space.used	integer	query	False	Filter by space.used
space.size	integer	query	False	Filter by space.size • Max value: 140737488355328 • Min value: 4096

Name	Type	In	Required	Description
space.physical_used_by_snapshots	integer	query	False	Filter by space.physical_used_by_snapshots • Introduced in: 9.16
create_time	string	query	False	Filter by create_time • Introduced in: 9.7
statistics.iops_raw.write	integer	query	False	Filter by statistics.iops_raw.write • Introduced in: 9.8
statistics.iops_raw.total	integer	query	False	Filter by statistics.iops_raw.total • Introduced in: 9.8
statistics.iops_raw.read	integer	query	False	Filter by statistics.iops_raw.read • Introduced in: 9.8
statistics.iops_raw.other	integer	query	False	Filter by statistics.iops_raw.other • Introduced in: 9.8
statistics.throughput_raw.total	integer	query	False	Filter by statistics.throughput_raw.total • Introduced in: 9.8

Name	Type	In	Required	Description
statistics.throughput_raw.write	integer	query	False	Filter by statistics.throughput_raw.write • Introduced in: 9.8
statistics.throughput_raw.read	integer	query	False	Filter by statistics.throughput_raw.read • Introduced in: 9.8
statistics.timestamp	string	query	False	Filter by statistics.timestamp • Introduced in: 9.8
statistics.latency_raw.write	integer	query	False	Filter by statistics.latency_raw.write • Introduced in: 9.8
statistics.latency_raw.total	integer	query	False	Filter by statistics.latency_raw.total • Introduced in: 9.8
statistics.latency_raw.read	integer	query	False	Filter by statistics.latency_raw.read • Introduced in: 9.8
statistics.latency_raw.other	integer	query	False	Filter by statistics.latency_raw.other • Introduced in: 9.8

Name	Type	In	Required	Description
statistics.status	string	query	False	Filter by statistics.status • Introduced in: 9.8
consistency_group.uuid	string	query	False	Filter by consistency_group.uuid • Introduced in: 9.16
consistency_group.name	string	query	False	Filter by consistency_group.name • Introduced in: 9.16
name	string	query	False	Filter by name
metric.status	string	query	False	Filter by metric.status • Introduced in: 9.8
metric.timestamp	string	query	False	Filter by metric.timestamp • Introduced in: 9.8
metric.iops.write	integer	query	False	Filter by metric.iops.write • Introduced in: 9.8
metric.iops.total	integer	query	False	Filter by metric.iops.total • Introduced in: 9.8

Name	Type	In	Required	Description
metric.iops.read	integer	query	False	Filter by metric.iops.read • Introduced in: 9.8
metric.iops.other	integer	query	False	Filter by metric.iops.other • Introduced in: 9.8
metric.latency.write	integer	query	False	Filter by metric.latency.write • Introduced in: 9.8
metric.latency.total	integer	query	False	Filter by metric.latency.total • Introduced in: 9.8
metric.latency.read	integer	query	False	Filter by metric.latency.read • Introduced in: 9.8
metric.latency.other	integer	query	False	Filter by metric.latency.other • Introduced in: 9.8
metric.duration	string	query	False	Filter by metric.duration • Introduced in: 9.8
metric.throughput.total	integer	query	False	Filter by metric.throughput.total • Introduced in: 9.8

Name	Type	In	Required	Description
metric.throughput.write	integer	query	False	Filter by metric.throughput.write • Introduced in: 9.8
metric.throughput.read	integer	query	False	Filter by metric.throughput.read • Introduced in: 9.8
svm.name	string	query	False	Filter by svm.name
svm.uuid	string	query	False	Filter by svm.uuid
location.volume.uuid	string	query	False	Filter by location.volume.uuid
location.volume.name	string	query	False	Filter by location.volume.name
location.namespace	string	query	False	Filter by location.namespace
location.node.uuid	string	query	False	Filter by location.node.uuid • Introduced in: 9.10
location.node.name	string	query	False	Filter by location.node.name • Introduced in: 9.10
location.qtree.name	string	query	False	Filter by location.qtree.name

Name	Type	In	Required	Description
location.qtree.id	integer	query	False	Filter by location.qtree.id • Max value: 4994 • Min value: 0
enabled	boolean	query	False	Filter by enabled
fields	array[string]	query	False	Specify the fields to return.
max_records	integer	query	False	Limit the number of records returned.
return_records	boolean	query	False	The default is true for GET calls. When set to false, only the number of records is returned. • Default value: 1
return_timeout	integer	query	False	The number of seconds to allow the call to execute before returning. When iterating over a collection, the default is 15 seconds. ONTAP returns earlier if either max records or the end of the collection is reached. • Max value: 120 • Min value: 0 • Default value: 1
order_by	array[string]	query	False	Order results by specified fields and optional [asc

Response

```
Status: 200, Ok
```

Name	Type	Description
_links	_links	
num_records	integer	The number of records in the response.
records	array[nvme_namespace]	

Example response


```
{
  "_links": {
    "next": {
      "href": "/api/resourcelink"
    },
    "self": {
      "href": "/api/resourcelink"
    }
  },
  "num_records": 1,
  "records": [
    {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "comment": "string",
      "consistency_group": {
        "_links": {
          "self": {
            "href": "/api/resourcelink"
          }
        },
        "name": "cg1",
        "uuid": "4abc2317-4332-9d37-93a0-20bd29c22df0"
      },
      "create_time": "2018-06-04 15:00:00 -0400",
      "location": {
        "namespace": "namespace1",
        "node": {
          "_links": {
            "self": {
              "href": "/api/resourcelink"
            }
          },
          "name": "node1",
          "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
        },
        "qtree": {
          "_links": {
            "self": {
              "href": "/api/resourcelink"
            }
```

```
        },
        "id": 1,
        "name": "qt1"
      },
      "volume": {
        "_links": {
          "self": {
            "href": "/api/resourcelink"
          }
        },
        "name": "volume1",
        "uuid": "028baa66-41bd-11e9-81d5-00a0986138f7"
      }
    },
    "metric": {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "duration": "PT15S",
      "iops": {
        "read": 200,
        "total": 1000,
        "write": 100
      },
      "latency": {
        "read": 200,
        "total": 1000,
        "write": 100
      },
      "status": "ok",
      "throughput": {
        "read": 200,
        "total": 1000,
        "write": 100
      },
      "timestamp": "2017-01-25 06:20:13 -0500"
    },
    "name": "/vol/volume1/qtree1/namespace1",
    "os_type": "string",
    "space": {
      "block_size": 512,
      "efficiency_ratio": 2.5,
      "physical_used": 1073741824,
      "physical_used_by_snapshots": 1073741824,
```

```
      "size": 1073741824,
      "used": 0
    },
    "statistics": {
      "iops_raw": {
        "read": 200,
        "total": 1000,
        "write": 100
      },
      "latency_raw": {
        "read": 200,
        "total": 1000,
        "write": 100
      },
      "status": "ok",
      "throughput_raw": {
        "read": 200,
        "total": 1000,
        "write": 100
      },
      "timestamp": "2017-01-25 06:20:13 -0500"
    },
    "status": {
      "container_state": "string",
      "state": "online"
    },
    "subsystem_map": {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "anagrpid": "00103050h",
      "nsid": "00000001h",
      "subsystem": {
        "_links": {
          "self": {
            "href": "/api/resourcelink"
          }
        },
        "comment": "string",
        "hosts": [
          {
            "dh_hmac_chap": {
              "group_size": "string",
              "hash_function": "string",
```

```
                "mode": "bidirectional"
              },
              "nqn": "nqn.1992-01.example.com:string",
              "priority": "string",
              "tls": {
                "key_type": "configured"
              }
            }
          ],
          "name": "subsystem1",
          "os_type": "string",
          "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
        }
      },
      "svm": {
        "_links": {
          "self": {
            "href": "/api/resourcelink"
          }
        },
        "name": "svm1",
        "uuid": "02c9e252-41be-11e9-81d5-00a0986138f7"
      },
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    }
  ]
}
```


Error

```
Status: Default, Error
```

Name	Type	Description
error	returned_error	

Example error


```
{
  "error": {
    "arguments": [
      {
        "code": "string",
        "message": "string"
      }
    ],
    "code": "4",
    "message": "entry doesn't exist",
    "target": "uuid"
  }
}
```


Definitions

See Definitions

href

Name	Type	Description
href	string	

_links

Name	Type	Description
next	href	
self	href	

_links

Name	Type	Description
self	href	

source

The source NVMe namespace for a namespace clone operation. This can be specified using property `clone.source.uuid` or `clone.source.name`. If both properties are supplied, they must refer to the same namespace.

Valid in POST to create a new NVMe namespace as a clone of the source.

Valid in PATCH to overwrite an existing NVMe namespace's data as a clone of another.

clone

This sub-object is used in POST to create a new NVMe namespace as a clone of an existing namespace, or PATCH to overwrite an existing namespace as a clone of another. Setting a property in this sub-object indicates that a namespace clone is desired.

When used in a PATCH, the patched NVMe namespace's data is over-written as a clone of the source and the following properties are preserved from the patched namespace unless otherwise specified as part of the PATCH: `auto_delete` (unless specified in the request), `subsystem_map`, `status.state`, and `uuid`.

consistency_group

The namespace's consistency group. This property is populated for namespaces that are members of a consistency group. If the namespace is a member of a child consistency group, the parent consistency group is reported. A namespace's consistency group is the consistency group of its containing volume.

Name	Type	Description
_links	_links	

Name	Type	Description
name	string	The name of the consistency group.
uuid	string	The unique identifier of the consistency group.

lun

The source LUN for convert operation. This can be specified using property `convert.lun.uuid` or `convert.lun.name`. If both properties are supplied, they must refer to the same LUN.

Valid in POST. A convert request from LUN to NVMe namespace cannot be combined with setting any other namespace properties. All other properties of the converted NVMe namespace come from the source LUN.

convert

This sub-object is used in POST to convert a valid in-place LUN to an NVMe namespace. Setting a property in this sub-object indicates that a conversion from the specified LUN to NVMe namespace is desired.

node

The cluster node that hosts the NVMe namespace.

Name	Type	Description
_links	_links	
name	string	
uuid	string	

qtree

The qtree in which the NVMe namespace is optionally located. Valid in POST.

If properties `name` and `location.qtree.name` and/or `location.qtree.uuid` are specified in the same request, they must refer to the same qtree.

NVMe namespaces do not support rename.

Name	Type	Description
_links	_links	
id	integer	The identifier for the qtree, unique within the qtree's volume.
name	string	The name of the qtree.

volume

The volume in which the NVMe namespace is located. Valid in POST.

If properties `name` and `location.volume.name` and/or `location.volume.uuid` are specified in the same request, they must refer to the same volume.

NVMe namespaces do not support movement between volumes.

Name	Type	Description
_links	_links	
name	string	The name of the volume. This field cannot be specified in a PATCH method.
uuid	string	Unique identifier for the volume. This corresponds to the instance-uuid that is exposed in the CLI and ONTAPI. It does not change due to a volume move. • example: 028baa66-41bd-11e9-81d5-00a0986138f7 • Introduced in: 9.6 • x-nullable: true

location

The location of the NVMe namespace within the ONTAP cluster. NVMe namespaces do not support rename, or movement between volumes. Valid in POST.

Name	Type	Description
namespace	string	The base name component of the NVMe namespace. Valid in POST. If properties `name` and `location.namespace` are specified in the same request, they must refer to the base name. NVMe namespaces do not support rename.
node	node	The cluster node that hosts the NVMe namespace.

Name	Type	Description
qtree	qtree	The qtree in which the NVMe namespace is optionally located. Valid in POST. If properties `name` and `location.qtree.name` and/or `location.qtree.uuid` are specified in the same request, they must refer to the same qtree. NVMe namespaces do not support rename.
volume	volume	The volume in which the NVMe namespace is located. Valid in POST. If properties `name` and `location.volume.name` and/or `location.volume.uuid` are specified in the same request, they must refer to the same volume. NVMe namespaces do not support movement between volumes.

iops

The rate of I/O operations observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

latency

The round trip latency in microseconds observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

throughput

The rate of throughput bytes per second observed at the storage object.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

metric

Performance numbers, such as IOPS latency and throughput

Name	Type	Description
_links	_links	
duration	string	The duration over which this sample is calculated. The time durations are represented in the ISO-8601 standard format. Samples can be calculated over the following durations:

Name	Type	Description
iops	iops	The rate of I/O operations observed at the storage object.
latency	latency	The round trip latency in microseconds observed at the storage object.
status	string	Any errors associated with the sample. For example, if the aggregation of data over multiple nodes fails then any of the partial errors might be returned, "ok" on success, or "error" on any internal uncategorized failure. Whenever a sample collection is missed but done at a later time, it is back filled to the previous 15 second timestamp and tagged with "backfilled_data". "Inconsistent_ delta_time" is encountered when the time between two collections is not the same for all nodes. Therefore, the aggregated value might be over or under inflated. "Negative_delta" is returned when an expected monotonically increasing value has decreased in value. "Inconsistent_old_data" is returned when one or more nodes do not have the latest data.
throughput	throughput	The rate of throughput bytes per second observed at the storage object.
timestamp	string	The timestamp of the performance data.

self_link

Name	Type	Description
self	href	

qos_policy

The QoS policy for the volume provisioned to host the namespace. This property is only supported when the request provisions a new volume. If no `qos_policy` is provided, a policy is be set based on the `storage_service.name`, which defaults to the most performant service available.

Name	Type	Description
_links	self_link	
name	string	The QoS policy group name. This is mutually exclusive with UUID and other QoS attributes during POST and PATCH.
uuid	string	The QoS policy group UUID. This is mutually exclusive with name and other QoS attributes during POST and PATCH.

snapshot_policy_reference

This is a reference to the snapshot policy.

Name	Type	Description
_links	_links	
name	string	
uuid	string	

storage_service

Determines the placement of the namespace based on the value specified. Valid in POST.

Name	Type	Description
name	string	Storage service name. If not specified, the default value is the most performant for the platform.

object_stores

tiering

The tiering placement and policy definitions for the volume provisioned to host the namespace. This property is only supported when the request provisions a new volume.

Name	Type	Description
object_stores	array[object_stores]	Object stores to use. Used for placement.

Name	Type	Description
policy	string	Policy that determines whether the user data blocks of a volume in a FabricPool will be tiered to the cloud store when they become cold. FabricPool combines flash (performance tier) with a cloud store into a single aggregate. Temperature of a volume block increases if it is accessed frequently and decreases when it is not. Valid in POST or PATCH. all ‐ Allows tiering of both snapshots and active file system user data to the cloud store as soon as possible by ignoring the temperature on the volume blocks. auto ‐ Allows tiering of both snapshot and active file system user data to the cloud store none ‐ Volume blocks are not be tiered to the cloud store. snapshot_only ‐ Allows tiering of only the volume snapshots not associated with the active file system. The default tiering policy is "snapshot-only" for a FlexVol volume and "none" for a FlexGroup volume. The default minimum cooling period for the "snapshot-only" tiering policy is 2 days and for the "auto" tiering policy it is 31 days.

provisioning_options

Options that are applied to the operation.

guarantee

Properties that request and report the space guarantee for the NVMe namespace.

Name	Type	Description
requested	boolean	The requested space reservation policy for the NVMe namespace. If *true*, a space reservation is requested for the namespace; if *false*, the namespace is thin provisioned. Guaranteeing a space reservation request for a namespace requires that the volume in which the namespace resides also be space reserved and that the fractional reserve for the volume be 100%. The space reservation policy for an NVMe namespace is determined by ONTAP. • readOnly: 1 • Introduced in: 9.6 • x-nullable: true
reserved	boolean	Reports if the NVMe namespace is space guaranteed. This property is *true* if a space guarantee is requested and the containing volume and aggregate support the request. This property is *false* if a space guarantee is not requested or if a space guarantee is requested and either the containing volume and aggregate do not support the request.

space

The storage space related properties of the NVMe namespace.

Name	Type	Description
block_size	integer	The size of blocks in the namespace in bytes. The default for namespaces with an `os_type` of *vmware* is *512*. All other namespaces default to *4096*. Valid in POST when creating an NVMe namespace that is not a clone of another. Disallowed in POST when creating a namespace clone. Valid in POST.
efficiency_ratio	number	The storage efficiency ratio of the namespace without snapshots. (Logical Used / Used) This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.
guarantee	guarantee	Properties that request and report the space guarantee for the NVMe namespace.
physical_used	integer	The number of bytes consumed on the disk by the namespace, excluding snapshots. This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.
physical_used_by_snapshots	integer	The number of bytes consumed on the disk by the namespace's snapshots. This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.

Name	Type	Description
size	integer	The total provisioned size of the NVMe namespace. Valid in POST and PATCH. The NVMe namespace size can be increased but not be made smaller using the REST interface. The maximum and minimum sizes listed here are the absolute maximum and absolute minimum sizes in bytes. The maximum size is variable with respect to large NVMe namespace support in ONTAP. If large namespaces are supported, the maximum size is 128 TB (140737488355328 bytes) and if not supported, the maximum size is just under 16 TB (17557557870592 bytes). The minimum size supported is always 4096 bytes. For more information, see *Size properties* in the *docs* section of the ONTAP REST API documentation. • example: 1073741824 • format: int64 • Max value: 140737488355328 • Min value: 4096 • Introduced in: 9.6 • x-nullable: true

Name	Type	Description
used	integer	The amount of space consumed by the main data stream of the NVMe namespace. This value is the total space consumed in the volume by the NVMe namespace, including filesystem overhead, but excluding prefix and suffix streams. Due to internal filesystem overhead and the many ways NVMe filesystems and applications utilize blocks within a namespace, this value does not necessarily reflect actual consumption/availability from the perspective of the filesystem or application. Without specific knowledge of how the namespace blocks are utilized outside of ONTAP, this property should not be used and an indicator for an out-of-space condition. For more information, see *Size properties* in the *docs* section of the ONTAP REST API documentation. • format: int64 • readOnly: 1 • Introduced in: 9.6 • x-nullable: true

iops_raw

The number of I/O operations observed at the storage object. This should be used along with delta time to calculate the rate of I/O operations per unit of time.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

latency_raw

The raw latency in microseconds observed at the storage object. This should be divided by the raw IOPS value to calculate the average latency per I/O operation.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

throughput_raw

Throughput bytes observed at the storage object. This should be used along with delta time to calculate the rate of throughput bytes per unit of time.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

statistics

These are raw performance numbers, such as IOPS latency and throughput. These numbers are aggregated across all nodes in the cluster and increase with the uptime of the cluster.

Name	Type	Description
iops_raw	iops_raw	The number of I/O operations observed at the storage object. This should be used along with delta time to calculate the rate of I/O operations per unit of time.
latency_raw	latency_raw	The raw latency in microseconds observed at the storage object. This should be divided by the raw IOPS value to calculate the average latency per I/O operation.
status	string	Any errors associated with the sample. For example, if the aggregation of data over multiple nodes fails then any of the partial errors might be returned, "ok" on success, or "error" on any internal uncategorized failure. Whenever a sample collection is missed but done at a later time, it is back filled to the previous 15 second timestamp and tagged with "backfilled_data". "Inconsistent_delta_time" is encountered when the time between two collections is not the same for all nodes. Therefore, the aggregated value might be over or under inflated. "Negative_delta" is returned when an expected monotonically increasing value has decreased in value. "Inconsistent_old_data" is returned when one or more nodes do not have the latest data.
throughput_raw	throughput_raw	Throughput bytes observed at the storage object. This should be used along with delta time to calculate the rate of throughput bytes per unit of time.

Name	Type	Description
timestamp	string	The timestamp of the performance data.

status

Status information about the NVMe namespace.

Name	Type	Description
container_state	string	The state of the volume and aggregate that contain the NVMe namespace. Namespaces are only available when their containers are available.
mapped	boolean	Reports if the NVMe namespace is mapped to an NVMe subsystem. There is an added computational cost to retrieving this property's value. It is not populated for a GET request unless it is explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.
read_only	boolean	Reports if the NVMe namespace allows only read access.
state	string	The state of the NVMe namespace. Normal states for a namespace are *online* and *offline*. Other states indicate errors.

consistency_group_nvme_host_dh_hmac_chap

A container for the configuration of NVMe in-band authentication using the DH-HMAC-CHAP protocol for a host.

Name	Type	Description
group_size	string	The Diffie-Hellman group size for NVMe in-band authentication. When property `host_secret_key` is provided, this property defaults to `2048_bit`. When supplied, the property `host_secret_key` must also be supplied. Optional in POST.
hash_function	string	The hash function for NVMe in-band authentication. When property `host_secret_key` is provided, this property defaults to `sha_256`. When supplied, the property `host_secret_key` must also be supplied. Optional in POST.
mode	string	The expected NVMe in-band authentication mode for the host. This property is an indication of which secrets are configured for the host. When set to: • none: The host has neither the host nor controller secret configured, and no authentication is performed. • unidirectional: The host has a host secret configured. The controller will authenticate the host. • bidirectional: The host has both a host and controller secret configured. The controller will authenticate the host and the host will authenticate the controller.

tls

A container for the configuration for NVMe/TCP-TLS transport session for the host.

Name	Type	Description
key_type	string	The method by which the TLS pre-shared key (PSK) is configured for the host. Optional in POST. The values for property `key_type` and property `configured_psk` must logically agree. Possible values: • `none` - TLS is not configured for the host connection. No value is allowed for property `configured_psk`. • `configured` - A user supplied PSK is configured for the NVMe/TCP-TLS transport connection between the host and the NVMe subsystem. A valid value for property `configured_psk` is required. This property defaults to `none` unless a value is supplied for `configured_psk` in which case it defaults to `configured`.

hosts

The NVMe host provisioned to access NVMe namespaces mapped to a subsystem.

Name	Type	Description
dh_hmac_chap	consistency_group_nvme_host_dh_hmac_chap	A container for the configuration of NVMe in-band authentication using the DH-HMAC-CHAP protocol for a host.
nqn	string	The NVMe qualified name (NQN) used to identify the NVMe storage target.

Name	Type	Description
priority	string	The host priority setting allocates appropriate NVMe I/O queues (count and depth) for the host to submit I/O commands. Absence of this property in GET implies io_queue count and I/O queue depth are being used.
tls	tls	A container for the configuration for NVMe/TCP-TLS transport session for the host.

subsystem

The NVMe subsystem to which the NVMe namespace is mapped.

Name	Type	Description
_links	_links	
comment	string	A configurable comment for the NVMe subsystem. Optional in POST.
hosts	array[hosts]	The NVMe hosts configured for access to the NVMe subsystem. Optional in POST.
name	string	The name of the NVMe subsystem. Once created, an NVMe subsystem cannot be renamed. Optional in POST.
os_type	string	The host operating system of the NVMe subsystem's hosts. Optional in POST.
uuid	string	The unique identifier of the NVMe subsystem.

subsystem_map

The NVMe subsystem with which the NVMe namespace is associated. A namespace can be mapped to zero (0) or one (1) subsystems.

There is an added computational cost to retrieving property values for `subsystem_map`. They are not populated for a GET request unless explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more. These properties are supported only for GET.

Name	Type	Description
_links	_links	
anagrpid	string	The Asymmetric Namespace Access Group ID (ANAGRPID) of the NVMe namespace. The format for an ANAGRPID is 8 hexadecimal digits (zero-filled) followed by a lower case "h".
nsid	string	The NVMe namespace identifier. This is an identifier used by an NVMe controller to provide access to the NVMe namespace. The format for an NVMe namespace identifier is 8 hexadecimal digits (zero-filled) followed by a lower case "h".
subsystem	subsystem	The NVMe subsystem to which the NVMe namespace is mapped.

svm

SVM, applies only to SVM-scoped objects.

Name	Type	Description
_links	_links	
name	string	The name of the SVM. This field cannot be specified in a PATCH method.
uuid	string	The unique identifier of the SVM. This field cannot be specified in a PATCH method.

nvme_namespace

An NVMe namespace is a collection of addressable logical blocks presented to hosts connected to the storage virtual machine using the NVMe over Fabrics protocol.

An NVMe namespace must be mapped to an NVMe subsystem to grant access to the subsystem's hosts. Hosts can then access the NVMe namespace and perform I/O using the NVMe over Fabrics protocol.

See the NVMe namespace object model to learn more about each of the properties supported by the NVMe namespace REST API. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume.

NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional.

An NVMe namespace is created to a specified size using thin or thick provisioning as determined by the volume on which it is created. An NVMe namespace can then be resized or cloned. An NVMe namespace cannot be renamed, or moved to a different volume. NVMe namespaces do not support the assignment of a QoS policy for performance management, but a QoS policy can be assigned to the volume containing the namespace.

</namespace></qtree></volume>

Name	Type	Description
_links	_links	
auto_delete	boolean	This property marks the NVMe namespace for auto deletion when the volume containing the namespace runs out of space. This is most commonly set on namespace clones. When set to *true*, the NVMe namespace becomes eligible for automatic deletion when the volume runs out of space. Auto deletion only occurs when the volume containing the namespace is also configured for auto deletion and free space in the volume decreases below a particular threshold. This property is optional in POST and PATCH. The default value for a new NVMe namespace is *false*. There is an added computational cost to retrieving this property's value. It is not populated for a GET request unless it is explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.
comment	string	A configurable comment available for use by the administrator. Valid in POST and PATCH.

Name	Type	Description
consistency_group	consistency_group	The namespace's consistency group. This property is populated for namespaces that are members of a consistency group. If the namespace is a member of a child consistency group, the parent consistency group is reported. A namespace's consistency group is the consistency group of its containing volume.
create_time	string	The time the NVMe namespace was created.
enabled	boolean	The enabled state of the NVMe namespace. Certain error conditions cause the namespace to become disabled. If the namespace is disabled, check the `status.state` property to determine what error disabled the namespace. An NVMe namespace is enabled automatically when it is created.
location	location	The location of the NVMe namespace within the ONTAP cluster. NVMe namespaces do not support rename, or movement between volumes. Valid in POST. • Introduced in: 9.6 • readCreate: 1
metric	metric	Performance numbers, such as IOPS latency and throughput

Name	Type	Description
name	string	The name of the NVMe namespace. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume. NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional. Renaming an NVMe namespace is not supported. Valid in POST.
os_type	string	The operating system type of the NVMe namespace. Required in POST when creating an NVMe namespace that is not a clone of another. Disallowed in POST when creating a namespace clone.
space	space	The storage space related properties of the NVMe namespace.
statistics	statistics	These are raw performance numbers, such as IOPS latency and throughput. These numbers are aggregated across all nodes in the cluster and increase with the uptime of the cluster.
status	status	Status information about the NVMe namespace.

Name	Type	Description
subsystem_map	subsystem_map	The NVMe subsystem with which the NVMe namespace is associated. A namespace can be mapped to zero (0) or one (1) subsystems. There is an added computational cost to retrieving property values for `subsystem_map`. They are not populated for a GET request unless explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more. These properties are supported only for GET.
svm	svm	SVM, applies only to SVM-scoped objects.
uuid	string	The unique identifier of the NVMe namespace.

error_arguments

Name	Type	Description
code	string	Argument code
message	string	Message argument

returned_error

Name	Type	Description
arguments	array[error_arguments]	Message arguments
code	string	Error code
message	string	Error message
target	string	The target parameter that caused the error.

Create an NVMe namespace

POST `/storage/namespaces`

Introduced In: 9.6

Creates an NVMe namespace.

Required properties

- `svm.uuid` or `svm.name` - Existing SVM in which to create the NVMe namespace.
- `name`, `location.volume.name` or `location.volume.uuid` - Existing volume in which to create the NVMe namespace.
- `name` or `location.namespace` - Base name for the NVMe namespace.
- `os_type` - Operating system from which the NVMe namespace will be accessed. (Not used for clones, which are created based on the `os_type` of the source NVMe namespace.)
- `space.size` - Size for the NVMe namespace. (Not used for clones, which are created based on the size of the source NVMe namespace.)

Default property values

If not specified in POST, the following default property values are assigned:

- `auto_delete` - *false*
- `space.block_size` - *4096* (*512* when 'os_type' is *vmware*)

Related ONTAP commands

- `volume file clone autodelete`
- `volume file clone create`
- `vserver nvme namespace convert-from-lun`
- `vserver nvme namespace create`

POST is asynchronous when creating a new namespace. It is synchronous when converting a LUN to a namespace via the `convert` property.

Learn more

- DOC /storage/namespaces

Parameters

Name	Type	In	Required	Description
return_timeout	integer	query	False	The number of seconds to allow the call to execute before returning. When doing a POST, PATCH, or DELETE operation on a single record, the default is 0 seconds. This means that if an asynchronous operation is started, the server immediately returns HTTP code 202 (Accepted) along with a link to the job. If a non-zero value is specified for POST, PATCH, or DELETE operations, ONTAP waits that length of time to see if the job completes so it can return something other than 202. • Default value: 1 • Max value: 120 • Min value: 0
return_records	boolean	query	False	The default is false. If set to true, the records are returned. • Default value:

Request Body

Name	Type	Description
auto_delete	boolean	This property marks the NVMe namespace for auto deletion when the volume containing the namespace runs out of space. This is most commonly set on namespace clones. When set to *true*, the NVMe namespace becomes eligible for automatic deletion when the volume runs out of space. Auto deletion only occurs when the volume containing the namespace is also configured for auto deletion and free space in the volume decreases below a particular threshold. This property is optional in POST and PATCH. The default value for a new NVMe namespace is *false*. There is an added computational cost to retrieving this property's value. It is not populated for a GET request unless it is explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.
clone	clone	This sub-object is used in POST to create a new NVMe namespace as a clone of an existing namespace, or PATCH to overwrite an existing namespace as a clone of another. Setting a property in this sub-object indicates that a namespace clone is desired. When used in a PATCH, the patched NVMe namespace's data is over-written as a clone of the source and the following properties are preserved from the patched namespace unless otherwise specified as part of the PATCH: `auto_delete` (unless specified in the request), `subsystem_map`, `status.state`, and `uuid`.

Name	Type	Description
comment	string	A configurable comment available for use by the administrator. Valid in POST and PATCH.
consistency_group	consistency_group	The namespace's consistency group. This property is populated for namespaces that are members of a consistency group. If the namespace is a member of a child consistency group, the parent consistency group is reported. A namespace's consistency group is the consistency group of its containing volume.
convert	convert	This sub-object is used in POST to convert a valid in-place LUN to an NVMe namespace. Setting a property in this sub-object indicates that a conversion from the specified LUN to NVMe namespace is desired.
create_time	string	The time the NVMe namespace was created.
enabled	boolean	The enabled state of the NVMe namespace. Certain error conditions cause the namespace to become disabled. If the namespace is disabled, check the `status.state` property to determine what error disabled the namespace. An NVMe namespace is enabled automatically when it is created.
location	location	The location of the NVMe namespace within the ONTAP cluster. NVMe namespaces do not support rename, or movement between volumes. Valid in POST. • Introduced in: 9.6 • readCreate: 1

Name	Type	Description
name	string	The name of the NVMe namespace. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume. NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional. Renaming an NVMe namespace is not supported. Valid in POST.
os_type	string	The operating system type of the NVMe namespace. Required in POST when creating an NVMe namespace that is not a clone of another. Disallowed in POST when creating a namespace clone.
provisioning_options	provisioning_options	Options that are applied to the operation.
space	space	The storage space related properties of the NVMe namespace.
status	status	Status information about the NVMe namespace.
subsystem_map	subsystem_map	The NVMe subsystem with which the NVMe namespace is associated. A namespace can be mapped to zero (0) or one (1) subsystems. There is an added computational cost to retrieving property values for `subsystem_map`. They are not populated for a GET request unless explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more. These properties are supported only for GET.

Name	Type	Description
svm	svm	SVM, applies only to SVM-scoped objects.
uuid	string	The unique identifier of the NVMe namespace.

Example request


```
{
  "clone": {
    "source": {
      "name": "/vol/volume1/namespace1",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    }
  },
  "comment": "string",
  "consistency_group": {
    "name": "cg1",
    "uuid": "4abc2317-4332-9d37-93a0-20bd29c22df0"
  },
  "convert": {
    "lun": {
      "name": "/vol/volume1/lun1",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    }
  },
  "create_time": "2018-06-04 15:00:00 -0400",
  "location": {
    "namespace": "namespace1",
    "node": {
      "name": "node1",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "qtree": {
      "id": 1,
      "name": "qt1"
    },
    "volume": {
      "name": "volume1",
      "uuid": "028baa66-41bd-11e9-81d5-00a0986138f7"
    }
  },
  "name": "/vol/volume1/qtree1/namespace1",
  "os_type": "string",
  "provisioning_options": {
    "qos_policy": {
      "name": "performance",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "snapshot_policy": {
      "name": "default",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
```

```
    },
    "storage_service": {
      "name": "string"
    },
    "tiering": {
      "control": "string",
      "object_stores": [
        {
          "name": "string"
        }
      ],
      "policy": "string"
    }
  },
  "space": {
    "block_size": 512,
    "efficiency_ratio": 2.5,
    "physical_used": 1073741824,
    "physical_used_by_snapshots": 1073741824,
    "size": 1073741824,
    "used": 0
  },
  "status": {
    "container_state": "string",
    "state": "online"
  },
  "subsystem_map": {
    "anagrpid": "00103050h",
    "nsid": "00000001h",
    "subsystem": {
      "comment": "string",
      "hosts": [
        {
          "dh_hmac_chap": {
            "controller_secret_key": "DHHC-
1:00:ia6zGodOr4SEG0Zzaw398rpY0wqipUWj4jWjUh4HWUz6aQ2n:",
            "group_size": "string",
            "hash_function": "string",
            "host_secret_key": "DHHC-
1:00:ia6zGodOr4SEG0Zzaw398rpY0wqipUWj4jWjUh4HWUz6aQ2n:",
            "mode": "bidirectional"
          },
          "nqn": "nqn.1992-01.example.com:string",
          "priority": "string",
          "tls": {
            "configured_psk": "NVMeTLSkey-
```

```
1:01:VRLbtnN9AQb2WXW3c9+wEf/DRLz0QuLdbYvEhwtdWwNf9LrZ:",
            "key_type": "configured"
          }
        }
      ],
      "name": "subsystem1",
      "os_type": "string",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    }
  },
  "svm": {
    "name": "svm1",
    "uuid": "02c9e252-41be-11e9-81d5-00a0986138f7"
  },
  "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
}
```


Response

```
Status: 202, Accepted
```

Name	Type	Description
job	job_link	

Example response


```
{
  "job": {
    "uuid": "string"
  }
}
```


Headers

Name	Description	Type
Location	Useful for tracking the resource location	string

Response

```
Status: 201, Created
```

Error

```
Status: Default
```

ONTAP Error Response Codes

Error Code	Description
917927	The specified volume was not found.
918236	The specified `location.volume.uuid` and `location.volume.name` do not refer to the same volume.
1254197	The LUN specified for conversion to a namespace is mapped.
1260121	Cloning a namespace to a volume different than the source volume is not supported.
1260136	The specified destination for a clone operation already exists as a LUN, namespace, or file.
2621462	The supplied SVM does not exist.
2621706	The specified `svm.uuid` and `svm.name` do not refer to the same SVM.
2621707	No SVM was specified. Either `svm.name` or `svm.uuid` must be supplied.
5242927	The specified qtree was not found.
5242950	The specified `location.qtree.id` and `location.qtree.name` do not refer to the same qtree.
5374127	The specified namespace name is invalid.
5374140	LUN has a non-zero prefix and/or suffix size.
5374141	LUN is part of a SnapMirror active sync relationship.
5374156	A protocol endpoint LUN cannot be converted to an NVMe namespace.
5374157	LUN in an SVM with MetroCluster configured cannot be converted to an NVMe namespace.
5374158	LUN contains an operating system type that is not supported for NVMe namespace.
5374352	An invalid name was provided for the NVMe namespace.

Error Code	Description
5374858	The volume specified by `name` is not the same as that specified by `location.volume`.
5374860	The qtree specified by `name` is not the same as that specified by `location.qtree`.
5374861	The NVME namespace base name specified by `name` is not the same as that specified by `location.name`.
5374862	No NVMe namespace path base name was provided for the namespace.
5374876	The LUN specified for conversion to a namespace was not found.
5376461	The specified namespace name is invalid.
5376462	The specified namespace name is too long.
5376463	The snapshot portion of the specified namespace name is too long.
5440509	No suitable storage can be found for the specified requirements.
13565952	The NVMe namespace clone request failed.
72089636	Creating NVMe namespaces with `os_type` AIX is not supported until the effective cluster version is 9.13.1 or later.
72089720	NVMe namespaces cannot be created in snapshots.
72089721	The volume specified is in a load sharing mirror relationship. Namespaces are not supported in load sharing mirrors.
72089722	A negative size was provided for the NVMe namespace.
72089723	The specified size is too small for the NVMe namespace.
72089724	The specified size is too large for the NVMe namespace.
72089725	A LUN or NVMe namespace already exists at the specified path.
72089727	NVMe namespaces cannot be created on an SVM root volume.
72089728	NVMe namespaces cannot be created on a FlexGroup volume.
72089732	An NVMe namespace name can only contain characters A-Z, a-z, 0-9, "-", ".", "_", "{" and "}".

Error Code	Description
72090005	The specified `clone.source.uuid` and `clone.source.name` do not refer to the same NVMe namespace.
72090006	The specified `clone.source` was not found.
72090007	The specified `clone.source` was not found.
72090009	An error occurred after successfully creating the NVMe namespace. Some properties were not set.
72090012	The property cannot be specified when creating an NVMe namespace clone. The `target` property of the error object identifies the property.
72090013	The property is required except when creating an NVMe namespace clone. The `target` property of the error object identifies the property.
72090014	No volume was specified for the NVMe namespace.
72090015	An error occurred after successfully creating the NVMe namespace preventing the retrieval of its properties.
72090033	The `clone.source.uuid` property is not supported when specifying a source NVMe namespace from a snapshot.
72090039	The property cannot be specified at the same time when creating an NVMe namespace as a clone. The `target` property of the error object identifies the other property given with clone.
72090040	The property cannot be specified when converting a LUN into an NVMe namespace. The `target` property of the error object identifies the property.

Also see the table of common errors in the Response body overview section of this documentation.

Definitions

See Definitions

href

Name	Type	Description
href	string	

_links

source

The source NVMe namespace for a namespace clone operation. This can be specified using property `clone.source.uuid` or `clone.source.name`. If both properties are supplied, they must refer to the same namespace.

Valid in POST to create a new NVMe namespace as a clone of the source.

Valid in PATCH to overwrite an existing NVMe namespace's data as a clone of another.

Name	Type	Description
name	string	The name of the clone source NVMe namespace. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume. NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional. Valid in POST and PATCH.
uuid	string	The unique identifier of the clone source NVMe namespace. Valid in POST and PATCH.

clone

This sub-object is used in POST to create a new NVMe namespace as a clone of an existing namespace, or PATCH to overwrite an existing namespace as a clone of another. Setting a property in this sub-object indicates that a namespace clone is desired.

When used in a PATCH, the patched NVMe namespace's data is over-written as a clone of the source and the following properties are preserved from the patched namespace unless otherwise specified as part of the PATCH: `auto_delete` (unless specified in the request), `subsystem_map`, `status.state`, and `uuid`.

Name	Type	Description
source	source	The source NVMe namespace for a namespace clone operation. This can be specified using property `clone.source.uuid` or `clone.source.name`. If both properties are supplied, they must refer to the same namespace. Valid in POST to create a new NVMe namespace as a clone of the source. Valid in PATCH to overwrite an existing NVMe namespace's data as a clone of another.

consistency_group

The namespace's consistency group. This property is populated for namespaces that are members of a consistency group. If the namespace is a member of a child consistency group, the parent consistency group is reported. A namespace's consistency group is the consistency group of its containing volume.

Name	Type	Description
name	string	The name of the consistency group.
uuid	string	The unique identifier of the consistency group.

lun

The source LUN for convert operation. This can be specified using property `convert.lun.uuid` or `convert.lun.name`. If both properties are supplied, they must refer to the same LUN.

Valid in POST. A convert request from LUN to NVMe namespace cannot be combined with setting any other namespace properties. All other properties of the converted NVMe namespace come from the source LUN.

Name	Type	Description
name	string	The name of the source LUN. Valid in POST. A LUN is located within a volume. Optionally, it can be located within a qtree in a volume. LUN names are paths of the form "/vol/<volume>[/<qtree>]/<lun>" where the qtree name is optional.
uuid	string	The unique identifier of the source LUN. Valid in POST.

convert

This sub-object is used in POST to convert a valid in-place LUN to an NVMe namespace. Setting a property in this sub-object indicates that a conversion from the specified LUN to NVMe namespace is desired.

Name	Type	Description
lun	lun	The source LUN for convert operation. This can be specified using property `convert.lun.uuid` or `convert.lun.name`. If both properties are supplied, they must refer to the same LUN. Valid in POST. A convert request from LUN to NVMe namespace cannot be combined with setting any other namespace properties. All other properties of the converted NVMe namespace come from the source LUN.

node

The cluster node that hosts the NVMe namespace.

Name	Type	Description
name	string	
uuid	string	

qtree

The qtree in which the NVMe namespace is optionally located. Valid in POST.

If properties `name` and `location.qtree.name` and/or `location.qtree.uuid` are specified in the same request, they must refer to the same qtree.

NVMe namespaces do not support rename.

Name	Type	Description
id	integer	The identifier for the qtree, unique within the qtree's volume.
name	string	The name of the qtree.

volume

The volume in which the NVMe namespace is located. Valid in POST.

If properties `name` and `location.volume.name` and/or `location.volume.uuid` are specified in the same request, they must refer to the same volume.

NVMe namespaces do not support movement between volumes.

Name	Type	Description
name	string	The name of the volume. This field cannot be specified in a PATCH method.
uuid	string	Unique identifier for the volume. This corresponds to the instance-uuid that is exposed in the CLI and ONTAPI. It does not change due to a volume move. • example: 028baa66-41bd-11e9-81d5-00a0986138f7 • Introduced in: 9.6 • x-nullable: true

location

The location of the NVMe namespace within the ONTAP cluster. NVMe namespaces do not support rename, or movement between volumes. Valid in POST.

Name	Type	Description
namespace	string	The base name component of the NVMe namespace. Valid in POST. If properties `name` and `location.namespace` are specified in the same request, they must refer to the base name. NVMe namespaces do not support rename.
node	node	The cluster node that hosts the NVMe namespace.
qtree	qtree	The qtree in which the NVMe namespace is optionally located. Valid in POST. If properties `name` and `location.qtree.name` and/or `location.qtree.uuid` are specified in the same request, they must refer to the same qtree. NVMe namespaces do not support rename.
volume	volume	The volume in which the NVMe namespace is located. Valid in POST. If properties `name` and `location.volume.name` and/or `location.volume.uuid` are specified in the same request, they must refer to the same volume. NVMe namespaces do not support movement between volumes.

iops

The rate of I/O operations observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

latency

The round trip latency in microseconds observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

throughput

The rate of throughput bytes per second observed at the storage object.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.

Name	Type	Description
write	integer	Performance metric for write I/O operations.

metric

Performance numbers, such as IOPS latency and throughput

Name	Type	Description
duration	string	The duration over which this sample is calculated. The time durations are represented in the ISO-8601 standard format. Samples can be calculated over the following durations:
iops	iops	The rate of I/O operations observed at the storage object.
latency	latency	The round trip latency in microseconds observed at the storage object.
status	string	Any errors associated with the sample. For example, if the aggregation of data over multiple nodes fails then any of the partial errors might be returned, "ok" on success, or "error" on any internal uncategorized failure. Whenever a sample collection is missed but done at a later time, it is back filled to the previous 15 second timestamp and tagged with "backfilled_data". "Inconsistent_ delta_time" is encountered when the time between two collections is not the same for all nodes. Therefore, the aggregated value might be over or under inflated. "Negative_delta" is returned when an expected monotonically increasing value has decreased in value. "Inconsistent_old_data" is returned when one or more nodes do not have the latest data.

Name	Type	Description
throughput	throughput	The rate of throughput bytes per second observed at the storage object.
timestamp	string	The timestamp of the performance data.

self_link

qos_policy

The QoS policy for the volume provisioned to host the namespace. This property is only supported when the request provisions a new volume. If no `qos_policy` is provided, a policy is be set based on the `storage_service.name`, which defaults to the most performant service available.

Name	Type	Description
name	string	The QoS policy group name. This is mutually exclusive with UUID and other QoS attributes during POST and PATCH.
uuid	string	The QoS policy group UUID. This is mutually exclusive with name and other QoS attributes during POST and PATCH.

snapshot_policy_reference

This is a reference to the snapshot policy.

Name	Type	Description
name	string	
uuid	string	

storage_service

Determines the placement of the namespace based on the value specified. Valid in POST.

Name	Type	Description
name	string	Storage service name. If not specified, the default value is the most performant for the platform.

object_stores

Name	Type	Description
name	string	The name of the object store to use. Used for placement.

tiering

The tiering placement and policy definitions for the volume provisioned to host the namespace. This property is only supported when the request provisions a new volume.

Name	Type	Description
control	string	Storage tiering placement rules for the object.
object_stores	array[object_stores]	Object stores to use. Used for placement.

Name	Type	Description
policy	string	Policy that determines whether the user data blocks of a volume in a FabricPool will be tiered to the cloud store when they become cold. FabricPool combines flash (performance tier) with a cloud store into a single aggregate. Temperature of a volume block increases if it is accessed frequently and decreases when it is not. Valid in POST or PATCH. all ‐ Allows tiering of both snapshots and active file system user data to the cloud store as soon as possible by ignoring the temperature on the volume blocks. auto ‐ Allows tiering of both snapshot and active file system user data to the cloud store none ‐ Volume blocks are not be tiered to the cloud store. snapshot_only ‐ Allows tiering of only the volume snapshots not associated with the active file system. The default tiering policy is "snapshot-only" for a FlexVol volume and "none" for a FlexGroup volume. The default minimum cooling period for the "snapshot-only" tiering policy is 2 days and for the "auto" tiering policy it is 31 days.

provisioning_options

Options that are applied to the operation.

Name	Type	Description
auto	boolean	If the volume specified in the request does not exist, automatically provision one of appropriate size. If the volume does exist, resize it to accommodate the new namespace. This property is only supported on Unified ONTAP. The following behavior is different from a traditional POST request: • The operation is asynchronous. • The `qos_policy` property is supported and is applied to the provisioned volume. A default QoS policy is applied to the volume if one is not provided. • The `provisioning_options.count` property is supported, provisioning *count* namespaces on the volume using the specified properties. • The `subsystem_map` property is supported. If the specified subsystem does not exist, it is created. The namespace is mapped to this subsystem. If a subsystem is provisioned in this way, it is deleted after it is no longer mapped to any namespaces. • The `clone` and `convert` properties are not supported. • When performing `records` based operations, specifying this property in the query applies to the entire operation. Specifying it for an individual record within the request applies to only that record. • Many other `provisioning_options` properties are supported to control the placement of the namespace and the properties of the volume

Name	Type	Description
count	integer	The number of namespaces to provision with these properties. Only POST requests based on `space.size` are supported. When provided, the name is considered a prefix, and a suffix of the form _<*N*> is generated where N is the next available numeric index, starting with 1.
qos_policy	qos_policy	The QoS policy for the volume provisioned to host the namespace. This property is only supported when the request provisions a new volume. If no `qos_policy` is provided, a policy is be set based on the `storage_service.name`, which defaults to the most performant service available.
snapshot_policy	snapshot_policy_reference	This is a reference to the snapshot policy.
storage_service	storage_service	Determines the placement of the namespace based on the value specified. Valid in POST.
tiering	tiering	The tiering placement and policy definitions for the volume provisioned to host the namespace. This property is only supported when the request provisions a new volume.
use_mirrored_aggregates	boolean	Specifies whether mirrored aggregates are selected when provisioning the volume to host the namespace. Only mirrored aggregates are used if this parameter is set to *true* and only unmirrored aggregates are used if this parameter is set to *false*. The default value is *true* for a MetroCluster configuration and is *false* for a non-MetroCluster configuration.

guarantee

Properties that request and report the space guarantee for the NVMe namespace.

Name	Type	Description
requested	boolean	The requested space reservation policy for the NVMe namespace. If *true*, a space reservation is requested for the namespace; if *false*, the namespace is thin provisioned. Guaranteeing a space reservation request for a namespace requires that the volume in which the namespace resides also be space reserved and that the fractional reserve for the volume be 100%. The space reservation policy for an NVMe namespace is determined by ONTAP. • readOnly: 1 • Introduced in: 9.6 • x-nullable: true
reserved	boolean	Reports if the NVMe namespace is space guaranteed. This property is *true* if a space guarantee is requested and the containing volume and aggregate support the request. This property is *false* if a space guarantee is not requested or if a space guarantee is requested and either the containing volume and aggregate do not support the request.

space

The storage space related properties of the NVMe namespace.

Name	Type	Description
block_size	integer	The size of blocks in the namespace in bytes. The default for namespaces with an `os_type` of *vmware* is *512*. All other namespaces default to *4096*. Valid in POST when creating an NVMe namespace that is not a clone of another. Disallowed in POST when creating a namespace clone. Valid in POST.
efficiency_ratio	number	The storage efficiency ratio of the namespace without snapshots. (Logical Used / Used) This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.
guarantee	guarantee	Properties that request and report the space guarantee for the NVMe namespace.
physical_used	integer	The number of bytes consumed on the disk by the namespace, excluding snapshots. This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.
physical_used_by_snapshots	integer	The number of bytes consumed on the disk by the namespace's snapshots. This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.

Name	Type	Description
size	integer	The total provisioned size of the NVMe namespace. Valid in POST and PATCH. The NVMe namespace size can be increased but not be made smaller using the REST interface. The maximum and minimum sizes listed here are the absolute maximum and absolute minimum sizes in bytes. The maximum size is variable with respect to large NVMe namespace support in ONTAP. If large namespaces are supported, the maximum size is 128 TB (140737488355328 bytes) and if not supported, the maximum size is just under 16 TB (17557557870592 bytes). The minimum size supported is always 4096 bytes. For more information, see *Size properties* in the *docs* section of the ONTAP REST API documentation. • example: 1073741824 • format: int64 • Max value: 140737488355328 • Min value: 4096 • Introduced in: 9.6 • x-nullable: true

Name	Type	Description
used	integer	The amount of space consumed by the main data stream of the NVMe namespace. This value is the total space consumed in the volume by the NVMe namespace, including filesystem overhead, but excluding prefix and suffix streams. Due to internal filesystem overhead and the many ways NVMe filesystems and applications utilize blocks within a namespace, this value does not necessarily reflect actual consumption/availability from the perspective of the filesystem or application. Without specific knowledge of how the namespace blocks are utilized outside of ONTAP, this property should not be used and an indicator for an out-of-space condition. For more information, see *Size properties* in the *docs* section of the ONTAP REST API documentation. • format: int64 • readOnly: 1 • Introduced in: 9.6 • x-nullable: true

iops_raw

The number of I/O operations observed at the storage object. This should be used along with delta time to calculate the rate of I/O operations per unit of time.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

latency_raw

The raw latency in microseconds observed at the storage object. This should be divided by the raw IOPS value to calculate the average latency per I/O operation.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

throughput_raw

Throughput bytes observed at the storage object. This should be used along with delta time to calculate the rate of throughput bytes per unit of time.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

statistics

These are raw performance numbers, such as IOPS latency and throughput. These numbers are aggregated across all nodes in the cluster and increase with the uptime of the cluster.

Name	Type	Description
iops_raw	iops_raw	The number of I/O operations observed at the storage object. This should be used along with delta time to calculate the rate of I/O operations per unit of time.
latency_raw	latency_raw	The raw latency in microseconds observed at the storage object. This should be divided by the raw IOPS value to calculate the average latency per I/O operation.
status	string	Any errors associated with the sample. For example, if the aggregation of data over multiple nodes fails then any of the partial errors might be returned, "ok" on success, or "error" on any internal uncategorized failure. Whenever a sample collection is missed but done at a later time, it is back filled to the previous 15 second timestamp and tagged with "backfilled_data". "Inconsistent_delta_time" is encountered when the time between two collections is not the same for all nodes. Therefore, the aggregated value might be over or under inflated. "Negative_delta" is returned when an expected monotonically increasing value has decreased in value. "Inconsistent_old_data" is returned when one or more nodes do not have the latest data.
throughput_raw	throughput_raw	Throughput bytes observed at the storage object. This should be used along with delta time to calculate the rate of throughput bytes per unit of time.

Name	Type	Description
timestamp	string	The timestamp of the performance data.

status

Status information about the NVMe namespace.

Name	Type	Description
container_state	string	The state of the volume and aggregate that contain the NVMe namespace. Namespaces are only available when their containers are available.
mapped	boolean	Reports if the NVMe namespace is mapped to an NVMe subsystem. There is an added computational cost to retrieving this property's value. It is not populated for a GET request unless it is explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.
read_only	boolean	Reports if the NVMe namespace allows only read access.
state	string	The state of the NVMe namespace. Normal states for a namespace are *online* and *offline*. Other states indicate errors.

consistency_group_nvme_host_dh_hmac_chap

A container for the configuration of NVMe in-band authentication using the DH-HMAC-CHAP protocol for a host.

Name	Type	Description
controller_secret_key	string	The controller secret for NVMe in-band authentication. The value of this property is used by the NVMe host to authenticate the NVMe controller while establishing a connection. If unset, the controller is not authenticated. When supplied, the property `host_secret_key` must also be supplied. Optional in POST. This property is write-only. The `mode` property can be used to identify if a controller secret has been set for the host, but the controller secret value cannot be read. To change the value, the host must be deleted from the subsystem and re-added.
group_size	string	The Diffie-Hellman group size for NVMe in-band authentication. When property `host_secret_key` is provided, this property defaults to `2048_bit`. When supplied, the property `host_secret_key` must also be supplied. Optional in POST.
hash_function	string	The hash function for NVMe in-band authentication. When property `host_secret_key` is provided, this property defaults to `sha_256`. When supplied, the property `host_secret_key` must also be supplied. Optional in POST.

Name	Type	Description
host_secret_key	string	The host secret for NVMe in-band authentication. The value of this property is used by the NVMe controller to authenticate the NVMe host while establishing a connection. If unset, no authentication is performed by the host or controller. This property must be supplied if any other NVMe in-band authentication properties are supplied. Optional in POST. This property is write-only. The `mode` property can be used to identify if a host secret has been set for the host, but the host secret value cannot be read. To change the value, the host must be deleted from the subsystem and re-added.
mode	string	The expected NVMe in-band authentication mode for the host. This property is an indication of which secrets are configured for the host. When set to: • none: The host has neither the host nor controller secret configured, and no authentication is performed. • unidirectional: The host has a host secret configured. The controller will authenticate the host. • bidirectional: The host has both a host and controller secret configured. The controller will authenticate the host and the host will authenticate the controller.

tls

A container for the configuration for NVMe/TCP-TLS transport session for the host.

Name	Type	Description
configured_psk	string	A user supplied pre-shared key (PSK) value in PSK Interchange Format. Optional in POST. The values for property `key_type` and property `configured_psk` must logically agree. This property is only allowed when `key_type` is `configured`. If `configured_psk` is supplied and `key_type` is unset, `key_type` defaults to `configured`. This property is write-only. The `key_type` property can be used to identify if a configured PSK has been set for the host, but the PSK value cannot be read. To change the value, the host must be deleted from the subsystem and re-added.

Name	Type	Description
key_type	string	The method by which the TLS pre-shared key (PSK) is configured for the host. Optional in POST. The values for property `key_type` and property `configured_psk` must logically agree. Possible values: • `none` - TLS is not configured for the host connection. No value is allowed for property `configured_psk`. • `configured` - A user supplied PSK is configured for the NVMe/TCP-TLS transport connection between the host and the NVMe subsystem. A valid value for property `configured_psk` is required. This property defaults to `none` unless a value is supplied for `configured_psk` in which case it defaults to `configured`.

hosts

The NVMe host provisioned to access NVMe namespaces mapped to a subsystem.

Name	Type	Description
dh_hmac_chap	consistency_group_nvme_host_dh_hmac_chap	A container for the configuration of NVMe in-band authentication using the DH-HMAC-CHAP protocol for a host.
nqn	string	The NVMe qualified name (NQN) used to identify the NVMe storage target.

Name	Type	Description
priority	string	The host priority setting allocates appropriate NVMe I/O queues (count and depth) for the host to submit I/O commands. Absence of this property in GET implies io_queue count and I/O queue depth are being used.
tls	tls	A container for the configuration for NVMe/TCP-TLS transport session for the host.

subsystem

The NVMe subsystem to which the NVMe namespace is mapped.

Name	Type	Description
comment	string	A configurable comment for the NVMe subsystem. Optional in POST.
hosts	array[hosts]	The NVMe hosts configured for access to the NVMe subsystem. Optional in POST.
name	string	The name of the NVMe subsystem. Once created, an NVMe subsystem cannot be renamed. Optional in POST.
os_type	string	The host operating system of the NVMe subsystem's hosts. Optional in POST.
uuid	string	The unique identifier of the NVMe subsystem.

subsystem_map

The NVMe subsystem with which the NVMe namespace is associated. A namespace can be mapped to zero (0) or one (1) subsystems.

There is an added computational cost to retrieving property values for `subsystem_map`. They are not populated for a GET request unless explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more. These properties are supported only for GET.

Name	Type	Description
anagrpid	string	The Asymmetric Namespace Access Group ID (ANAGRPID) of the NVMe namespace. The format for an ANAGRPID is 8 hexadecimal digits (zero-filled) followed by a lower case "h".
nsid	string	The NVMe namespace identifier. This is an identifier used by an NVMe controller to provide access to the NVMe namespace. The format for an NVMe namespace identifier is 8 hexadecimal digits (zero-filled) followed by a lower case "h".
subsystem	subsystem	The NVMe subsystem to which the NVMe namespace is mapped.

svm

SVM, applies only to SVM-scoped objects.

Name	Type	Description
name	string	The name of the SVM. This field cannot be specified in a PATCH method.
uuid	string	The unique identifier of the SVM. This field cannot be specified in a PATCH method.

nvme_namespace

An NVMe namespace is a collection of addressable logical blocks presented to hosts connected to the storage virtual machine using the NVMe over Fabrics protocol.

An NVMe namespace must be mapped to an NVMe subsystem to grant access to the subsystem's hosts. Hosts can then access the NVMe namespace and perform I/O using the NVMe over Fabrics protocol.

See the NVMe namespace object model to learn more about each of the properties supported by the NVMe namespace REST API. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume.

NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional.

An NVMe namespace is created to a specified size using thin or thick provisioning as determined by the volume on which it is created. An NVMe namespace can then be resized or cloned. An NVMe namespace cannot be renamed, or moved to a different volume. NVMe namespaces do not support the assignment of a QoS policy for performance management, but a QoS policy can be assigned to the volume containing the namespace.

</namespace></qtree></volume>

Name	Type	Description
auto_delete	boolean	This property marks the NVMe namespace for auto deletion when the volume containing the namespace runs out of space. This is most commonly set on namespace clones. When set to *true*, the NVMe namespace becomes eligible for automatic deletion when the volume runs out of space. Auto deletion only occurs when the volume containing the namespace is also configured for auto deletion and free space in the volume decreases below a particular threshold. This property is optional in POST and PATCH. The default value for a new NVMe namespace is *false*. There is an added computational cost to retrieving this property's value. It is not populated for a GET request unless it is explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.

Name	Type	Description
clone	clone	This sub-object is used in POST to create a new NVMe namespace as a clone of an existing namespace, or PATCH to overwrite an existing namespace as a clone of another. Setting a property in this sub-object indicates that a namespace clone is desired. When used in a PATCH, the patched NVMe namespace's data is over-written as a clone of the source and the following properties are preserved from the patched namespace unless otherwise specified as part of the PATCH: `auto_delete` (unless specified in the request), `subsystem_map`, `status.state`, and `uuid`.
comment	string	A configurable comment available for use by the administrator. Valid in POST and PATCH.
consistency_group	consistency_group	The namespace's consistency group. This property is populated for namespaces that are members of a consistency group. If the namespace is a member of a child consistency group, the parent consistency group is reported. A namespace's consistency group is the consistency group of its containing volume.
convert	convert	This sub-object is used in POST to convert a valid in-place LUN to an NVMe namespace. Setting a property in this sub-object indicates that a conversion from the specified LUN to NVMe namespace is desired.
create_time	string	The time the NVMe namespace was created.

Name	Type	Description
enabled	boolean	The enabled state of the NVMe namespace. Certain error conditions cause the namespace to become disabled. If the namespace is disabled, check the `status.state` property to determine what error disabled the namespace. An NVMe namespace is enabled automatically when it is created.
location	location	The location of the NVMe namespace within the ONTAP cluster. NVMe namespaces do not support rename, or movement between volumes. Valid in POST. • Introduced in: 9.6 • readCreate: 1
name	string	The name of the NVMe namespace. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume. NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional. Renaming an NVMe namespace is not supported. Valid in POST.
os_type	string	The operating system type of the NVMe namespace. Required in POST when creating an NVMe namespace that is not a clone of another. Disallowed in POST when creating a namespace clone.
provisioning_options	provisioning_options	Options that are applied to the operation.

Name	Type	Description
space	space	The storage space related properties of the NVMe namespace.
status	status	Status information about the NVMe namespace.
subsystem_map	subsystem_map	The NVMe subsystem with which the NVMe namespace is associated. A namespace can be mapped to zero (0) or one (1) subsystems. There is an added computational cost to retrieving property values for `subsystem_map`. They are not populated for a GET request unless explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more. These properties are supported only for GET.
svm	svm	SVM, applies only to SVM-scoped objects.
uuid	string	The unique identifier of the NVMe namespace.

job_link

Name	Type	Description
uuid	string	The UUID of the asynchronous job that is triggered by a POST, PATCH, or DELETE operation.

error_arguments

Name	Type	Description
code	string	Argument code
message	string	Message argument

returned_error

Name	Type	Description
arguments	array[error_arguments]	Message arguments
code	string	Error code
message	string	Error message
target	string	The target parameter that caused the error.

Retrieve historical performance metrics for an NVMe namespace

GET `/storage/namespaces/{nvme_namespace.uuid}/metrics`

Introduced In: 9.14

Retrieves historical space and performance metrics for an NVMe namespace.

Related ONTAP commands

- `statistics namespace show`

Parameters

Name	Type	In	Required	Description
duration	string	query	False	Filter by duration
latency.write	integer	query	False	Filter by latency.write
latency.total	integer	query	False	Filter by latency.total
latency.read	integer	query	False	Filter by latency.read
latency.other	integer	query	False	Filter by latency.other
timestamp	string	query	False	Filter by timestamp
iops.write	integer	query	False	Filter by iops.write
iops.total	integer	query	False	Filter by iops.total

Name	Type	In	Required	Description
iops.read	integer	query	False	Filter by iops.read
iops.other	integer	query	False	Filter by iops.other
uuid	string	query	False	Filter by uuid
status	string	query	False	Filter by status
throughput.total	integer	query	False	Filter by throughput.total
throughput.write	integer	query	False	Filter by throughput.write
throughput.read	integer	query	False	Filter by throughput.read
nvme_namespace.uuid	string	path	True	The unique identifier of the NVMe namespace.

Name	Type	In	Required	Description
interval	string	query	False	The time range for the data. Examples can be 1h, 1d, 1m, 1w, 1y. The period for each time range is as follows: • 1h: Metrics over the most recent hour sampled over 15 seconds. • 1d: Metrics over the most recent day sampled over 5 minutes. • 1w: Metrics over the most recent week sampled over 30 minutes. • 1m: Metrics over the most recent month sampled over 2 hours. • 1y: Metrics over the most recent year sampled over a day. • Default value: 1 • enum: ["1h", "1d", "1w", "1m", "1y"]

Name	Type	In	Required	Description
return_timeout	integer	query	False	The number of seconds to allow the call to execute before returning. When iterating over a collection, the default is 15 seconds. ONTAP returns earlier if either max records or the end of the collection is reached. • Default value: 1 • Max value: 120 • Min value: 0
fields	array[string]	query	False	Specify the fields to return.
max_records	integer	query	False	Limit the number of records returned.
order_by	array[string]	query	False	Order results by specified fields and optional [asc
desc] direction. Default direction is 'asc' for ascending.	return_records	boolean	query	False

Response

```
Status: 200, Ok
```

Name	Type	Description
_links	_links	
num_records	integer	Number of records
records	array[records]	

Example response


```
{
  "_links": {
    "next": {
      "href": "/api/resourcelink"
    },
    "self": {
      "href": "/api/resourcelink"
    }
  },
  "num_records": 1,
  "records": [
    {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "duration": "PT15S",
      "iops": {
        "read": 200,
        "total": 1000,
        "write": 100
      },
      "latency": {
        "read": 200,
        "total": 1000,
        "write": 100
      },
      "status": "ok",
      "throughput": {
        "read": 200,
        "total": 1000,
        "write": 100
      },
      "timestamp": "2017-01-25 06:20:13 -0500",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    }
  ]
}
```

Error

```
Status: Default
```

ONTAP Error Response Codes

Error Code	Description
8585947	No metrics are available for the requested object.
8586225	An unexpected error occurred retrieving metrics for the requested object.

Also see the table of common errors in the Response body overview section of this documentation.

Name	Type	Description
error	returned_error	

Example error

```
{
  "error": {
    "arguments": [
      {
        "code": "string",
        "message": "string"
      }
    ],
    "code": "4",
    "message": "entry doesn't exist",
    "target": "uuid"
  }
}
```

Definitions

See Definitions

href

Name	Type	Description
href	string	

_links

Name	Type	Description
next	href	
self	href	

_links

Name	Type	Description
self	href	

iops

The rate of I/O operations observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

latency

The round trip latency in microseconds observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

throughput

The rate of throughput bytes per second observed at the storage object.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

records

Performance numbers, such as IOPS latency and throughput, for SVM protocols.

Name	Type	Description
_links	_links	
duration	string	The duration over which this sample is calculated. The time durations are represented in the ISO-8601 standard format. Samples can be calculated over the following durations:
iops	iops	The rate of I/O operations observed at the storage object.

Name	Type	Description
latency	latency	The round trip latency in microseconds observed at the storage object.
status	string	Any errors associated with the sample. For example, if the aggregation of data over multiple nodes fails then any of the partial errors might be returned, "ok" on success, or "error" on any internal uncategorized failure. Whenever a sample collection is missed but done at a later time, it is back filled to the previous 15 second timestamp and tagged with "backfilled_data". "Inconsistent_ delta_time" is encountered when the time between two collections is not the same for all nodes. Therefore, the aggregated value might be over or under inflated. "Negative_delta" is returned when an expected monotonically increasing value has decreased in value. "Inconsistent_old_data" is returned when one or more nodes do not have the latest data.
throughput	throughput	The rate of throughput bytes per second observed at the storage object.
timestamp	string	The timestamp of the performance data.
uuid	string	The unique identifier of the NVMe namespace.

error_arguments

Name	Type	Description
code	string	Argument code
message	string	Message argument

returned_error

Name	Type	Description
arguments	array[error_arguments]	Message arguments
code	string	Error code
message	string	Error message
target	string	The target parameter that caused the error.

Retrieve historical performance metrics for an NVMe namespace for a specific time

GET `/storage/namespaces/{nvme_namespace.uuid}/metrics/{timestamp}`

Introduced In: 9.14

Retrieves historical space and performance metrics for an NVMe namespace for a specific time.

Related ONTAP commands

- `statistics namespace show`

Parameters

Name	Type	In	Required	Description
nvme_namespace.uuid	string	path	True	The unique identifier of the NVMe namespace.
timestamp	string	path	True	The timestamp of the performance data. • format: date-time
fields	array[string]	query	False	Specify the fields to return.

Response

```
Status: 200, Ok
```

Name	Type	Description
_links	_links	
duration	string	The duration over which this sample is calculated. The time durations are represented in the ISO-8601 standard format. Samples can be calculated over the following durations:
iops	iops	The rate of I/O operations observed at the storage object.
latency	latency	The round trip latency in microseconds observed at the storage object.
status	string	Any errors associated with the sample. For example, if the aggregation of data over multiple nodes fails then any of the partial errors might be returned, "ok" on success, or "error" on any internal uncategorized failure. Whenever a sample collection is missed but done at a later time, it is back filled to the previous 15 second timestamp and tagged with "backfilled_data". "Inconsistent_ delta_time" is encountered when the time between two collections is not the same for all nodes. Therefore, the aggregated value might be over or under inflated. "Negative_delta" is returned when an expected monotonically increasing value has decreased in value. "Inconsistent_old_data" is returned when one or more nodes do not have the latest data.
throughput	throughput	The rate of throughput bytes per second observed at the storage object.
timestamp	string	The timestamp of the performance data.
uuid	string	The unique identifier of the NVMe namespace.

Example response


```
{
  "_links": {
    "self": {
      "href": "/api/resourcelink"
    }
  },
  "duration": "PT15S",
  "iops": {
    "read": 200,
    "total": 1000,
    "write": 100
  },
  "latency": {
    "read": 200,
    "total": 1000,
    "write": 100
  },
  "status": "ok",
  "throughput": {
    "read": 200,
    "total": 1000,
    "write": 100
  },
  "timestamp": "2017-01-25 06:20:13 -0500",
  "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
}
```


Error

```
Status: Default
```

ONTAP Error Response Codes

Error Code	Description
8585947	No metrics are available for the requested object.
8586225	An unexpected error occurred retrieving metrics for the requested object.

Also see the table of common errors in the Response body overview section of this documentation.

Name	Type	Description
error	returned_error	

Example error


```
{
  "error": {
    "arguments": [
      {
        "code": "string",
        "message": "string"
      }
    ],
    "code": "4",
    "message": "entry doesn't exist",
    "target": "uuid"
  }
}
```


Definitions

See Definitions

href

Name	Type	Description
href	string	

_links

Name	Type	Description
self	href	

iops

The rate of I/O operations observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

latency

The round trip latency in microseconds observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.

Name	Type	Description
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

throughput

The rate of throughput bytes per second observed at the storage object.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

error_arguments

Name	Type	Description
code	string	Argument code
message	string	Message argument

returned_error

Name	Type	Description
arguments	array[error_arguments]	Message arguments
code	string	Error code
message	string	Error message
target	string	The target parameter that caused the error.

Delete an NVMe namespace

DELETE `/storage/namespaces/{uuid}`

Introduced In: 9.6

Deletes an NVMe namespace.

Related ONTAP commands

- `vserver nvme namespace delete`

Learn more

- DOC /storage/namespaces

Parameters

Name	Type	In	Required	Description
uuid	string	path	True	The unique identifier of the NVMe namespace to delete.
allow_delete_while_mapped	boolean	query	False	Allows deletion of a mapped NVMe namespace. A mapped NVMe namespace might be in use. Deleting a mapped namespace also deletes the namespace map and makes the data no longer available, possibly causing a disruption in the availability of data. **This parameter should be used with caution.** • Default value:

Name	Type	In	Required	Description
return_timeout	integer	query	False	The number of seconds to allow the call to execute before returning. When doing a POST, PATCH, or DELETE operation on a single record, the default is 0 seconds. This means that if an asynchronous operation is started, the server immediately returns HTTP code 202 (Accepted) along with a link to the job. If a non-zero value is specified for POST, PATCH, or DELETE operations, ONTAP waits that length of time to see if the job completes so it can return something other than 202. • Default value: 1 • Max value: 120 • Min value: 0

Response

```
Status: 200, Ok
```

Response

```
Status: 202, Accepted
```

Error

```
Status: Default
```

ONTAP Error Response Codes

Error Code	Description
4	The specified namespace was not found.
72089796	The namespace must be unmapped before deletion.
72090016	The namespace's aggregate is offline. The aggregate must be online to modify or remove the namespace.
72090017	The namespace's volume is offline. The volume must be online to modify or remove the namespace.

Also see the table of common errors in the Response body overview section of this documentation.

Name	Type	Description
error	returned_error	

Example error


```
{
  "error": {
    "arguments": [
      {
        "code": "string",
        "message": "string"
      }
    ],
    "code": "4",
    "message": "entry doesn't exist",
    "target": "uuid"
  }
}
```


Definitions

See Definitions

error_arguments

Name	Type	Description
code	string	Argument code
message	string	Message argument

returned_error

Name	Type	Description
arguments	array[error_arguments]	Message arguments
code	string	Error code
message	string	Error message
target	string	The target parameter that caused the error.

Retrieve an NVMe namespace

GET `/storage/namespaces/{uuid}`

Introduced In: 9.6

Retrieves an NVMe namespace.

Expensive properties

There is an added computational cost to retrieving values for these properties. They are not included by default in GET results and must be explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.

- `auto_delete`
- `space.physical_used`
- `space.physical_used_by_snapshots`
- `space.efficiency_ratio`
- `subsystem_map.*`
- `status.mapped`
- `statistics.*`

- `metric.*`

Related ONTAP commands

- `vserver nvme namespace show`
- `vserver nvme subsystem map show`

Learn more

- DOC /storage/namespaces

Parameters

Name	Type	In	Required	Description
uuid	string	path	True	The unique identifier of the NVMe namespace to retrieve.
fields	array[string]	query	False	Specify the fields to return.

Response

```
Status: 200, Ok
```

Name	Type	Description
_links	_links	

Name	Type	Description
auto_delete	boolean	This property marks the NVMe namespace for auto deletion when the volume containing the namespace runs out of space. This is most commonly set on namespace clones. When set to *true*, the NVMe namespace becomes eligible for automatic deletion when the volume runs out of space. Auto deletion only occurs when the volume containing the namespace is also configured for auto deletion and free space in the volume decreases below a particular threshold. This property is optional in POST and PATCH. The default value for a new NVMe namespace is *false*. There is an added computational cost to retrieving this property's value. It is not populated for a GET request unless it is explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.
comment	string	A configurable comment available for use by the administrator. Valid in POST and PATCH.
consistency_group	consistency_group	The namespace's consistency group. This property is populated for namespaces that are members of a consistency group. If the namespace is a member of a child consistency group, the parent consistency group is reported. A namespace's consistency group is the consistency group of its containing volume.
create_time	string	The time the NVMe namespace was created.

Name	Type	Description
enabled	boolean	The enabled state of the NVMe namespace. Certain error conditions cause the namespace to become disabled. If the namespace is disabled, check the `status.state` property to determine what error disabled the namespace. An NVMe namespace is enabled automatically when it is created.
location	location	The location of the NVMe namespace within the ONTAP cluster. NVMe namespaces do not support rename, or movement between volumes. Valid in POST. • Introduced in: 9.6 • readCreate: 1
metric	metric	Performance numbers, such as IOPS latency and throughput
name	string	The name of the NVMe namespace. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume. NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional. Renaming an NVMe namespace is not supported. Valid in POST.
os_type	string	The operating system type of the NVMe namespace. Required in POST when creating an NVMe namespace that is not a clone of another. Disallowed in POST when creating a namespace clone.

Name	Type	Description
space	space	The storage space related properties of the NVMe namespace.
statistics	statistics	These are raw performance numbers, such as IOPS latency and throughput. These numbers are aggregated across all nodes in the cluster and increase with the uptime of the cluster.
status	status	Status information about the NVMe namespace.
subsystem_map	subsystem_map	The NVMe subsystem with which the NVMe namespace is associated. A namespace can be mapped to zero (0) or one (1) subsystems. There is an added computational cost to retrieving property values for `subsystem_map`. They are not populated for a GET request unless explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more. These properties are supported only for GET.
svm	svm	SVM, applies only to SVM-scoped objects.
uuid	string	The unique identifier of the NVMe namespace.

Example response


```
{
  "_links": {
    "self": {
      "href": "/api/resourcelink"
    }
  },
  "comment": "string",
  "consistency_group": {
    "_links": {
      "self": {
        "href": "/api/resourcelink"
      }
    },
    "name": "cg1",
    "uuid": "4abc2317-4332-9d37-93a0-20bd29c22df0"
  },
  "create_time": "2018-06-04 15:00:00 -0400",
  "location": {
    "namespace": "namespace1",
    "node": {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "name": "node1",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "qtree": {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "id": 1,
      "name": "qt1"
    },
    "volume": {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "name": "volume1",
```

```
      "uuid": "028baa66-41bd-11e9-81d5-00a0986138f7"
    }
  },
  "metric": {
    "_links": {
      "self": {
        "href": "/api/resourcelink"
      }
    },
    "duration": "PT15S",
    "iops": {
      "read": 200,
      "total": 1000,
      "write": 100
    },
    "latency": {
      "read": 200,
      "total": 1000,
      "write": 100
    },
    "status": "ok",
    "throughput": {
      "read": 200,
      "total": 1000,
      "write": 100
    },
    "timestamp": "2017-01-25 06:20:13 -0500"
  },
  "name": "/vol/volume1/qtree1/namespace1",
  "os_type": "string",
  "space": {
    "block_size": 512,
    "efficiency_ratio": 2.5,
    "physical_used": 1073741824,
    "physical_used_by_snapshots": 1073741824,
    "size": 1073741824,
    "used": 0
  },
  "statistics": {
    "iops_raw": {
      "read": 200,
      "total": 1000,
      "write": 100
    },
    "latency_raw": {
      "read": 200,
```

```
      "total": 1000,
      "write": 100
    },
    "status": "ok",
    "throughput_raw": {
      "read": 200,
      "total": 1000,
      "write": 100
    },
    "timestamp": "2017-01-25 06:20:13 -0500"
  },
  "status": {
    "container_state": "string",
    "state": "online"
  },
  "subsystem_map": {
    "_links": {
      "self": {
        "href": "/api/resourcelink"
      }
    },
    "anagrpid": "00103050h",
    "nsid": "00000001h",
    "subsystem": {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "comment": "string",
      "hosts": [
        {
          "dh_hmac_chap": {
            "group_size": "string",
            "hash_function": "string",
            "mode": "bidirectional"
          },
          "nqn": "nqn.1992-01.example.com:string",
          "priority": "string",
          "tls": {
            "key_type": "configured"
          }
        }
      ],
      "name": "subsystem1",
      "os_type": "string",
```

```
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    }
  },
  "svm": {
    "_links": {
      "self": {
        "href": "/api/resourcelink"
      }
    },
    "name": "svm1",
    "uuid": "02c9e252-41be-11e9-81d5-00a0986138f7"
  },
  "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
}
```


Error

```
Status: Default
```

ONTAP Error Response Codes

Error Code	Description
72090006	The specified namespace was not found.
72090007	The specified namespace was not found.

Also see the table of common errors in the Response body overview section of this documentation.

Name	Type	Description
error	returned_error	

Example error


```
{
  "error": {
    "arguments": [
      {
        "code": "string",
        "message": "string"
      }
    ],
    "code": "4",
    "message": "entry doesn't exist",
    "target": "uuid"
  }
}
```


Definitions

See Definitions

href

Name	Type	Description
href	string	

_links

Name	Type	Description
self	href	

source

The source NVMe namespace for a namespace clone operation. This can be specified using property `clone.source.uuid` or `clone.source.name`. If both properties are supplied, they must refer to the same namespace.

Valid in POST to create a new NVMe namespace as a clone of the source.

Valid in PATCH to overwrite an existing NVMe namespace's data as a clone of another.

clone

This sub-object is used in POST to create a new NVMe namespace as a clone of an existing namespace, or PATCH to overwrite an existing namespace as a clone of another. Setting a property in this sub-object indicates that a namespace clone is desired.

When used in a PATCH, the patched NVMe namespace's data is over-written as a clone of the source and the following properties are preserved from the patched namespace unless otherwise specified as part of the PATCH: `auto_delete` (unless specified in the request), `subsystem_map`, `status.state`, and `uuid`.

consistency_group

The namespace's consistency group. This property is populated for namespaces that are members of a consistency group. If the namespace is a member of a child consistency group, the parent consistency group is reported. A namespace's consistency group is the consistency group of its containing volume.

Name	Type	Description
_links	_links	
name	string	The name of the consistency group.
uuid	string	The unique identifier of the consistency group.

lun

The source LUN for convert operation. This can be specified using property `convert.lun.uuid` or

`convert.lun.name`. If both properties are supplied, they must refer to the same LUN.

Valid in POST. A convert request from LUN to NVMe namespace cannot be combined with setting any other namespace properties. All other properties of the converted NVMe namespace come from the source LUN.

convert

This sub-object is used in POST to convert a valid in-place LUN to an NVMe namespace. Setting a property in this sub-object indicates that a conversion from the specified LUN to NVMe namespace is desired.

node

The cluster node that hosts the NVMe namespace.

Name	Type	Description
_links	_links	
name	string	
uuid	string	

qtree

The qtree in which the NVMe namespace is optionally located. Valid in POST.

If properties `name` and `location.qtree.name` and/or `location.qtree.uuid` are specified in the same request, they must refer to the same qtree.

NVMe namespaces do not support rename.

Name	Type	Description
_links	_links	
id	integer	The identifier for the qtree, unique within the qtree's volume.
name	string	The name of the qtree.

volume

The volume in which the NVMe namespace is located. Valid in POST.

If properties `name` and `location.volume.name` and/or `location.volume.uuid` are specified in the same request, they must refer to the same volume.

NVMe namespaces do not support movement between volumes.

Name	Type	Description
_links	_links	

Name	Type	Description
name	string	The name of the volume. This field cannot be specified in a PATCH method.
uuid	string	Unique identifier for the volume. This corresponds to the instance-uuid that is exposed in the CLI and ONTAPI. It does not change due to a volume move. • example: 028baa66-41bd-11e9-81d5-00a0986138f7 • Introduced in: 9.6 • x-nullable: true

location

The location of the NVMe namespace within the ONTAP cluster. NVMe namespaces do not support rename, or movement between volumes. Valid in POST.

Name	Type	Description
namespace	string	The base name component of the NVMe namespace. Valid in POST. If properties `name` and `location.namespace` are specified in the same request, they must refer to the base name. NVMe namespaces do not support rename.
node	node	The cluster node that hosts the NVMe namespace.
qtree	qtree	The qtree in which the NVMe namespace is optionally located. Valid in POST. If properties `name` and `location.qtree.name` and/or `location.qtree.uuid` are specified in the same request, they must refer to the same qtree. NVMe namespaces do not support rename.

Name	Type	Description
volume	volume	The volume in which the NVMe namespace is located. Valid in POST. If properties `name` and `location.volume.name` and/or `location.volume.uuid` are specified in the same request, they must refer to the same volume. NVMe namespaces do not support movement between volumes.

iops

The rate of I/O operations observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

latency

The round trip latency in microseconds observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

throughput

The rate of throughput bytes per second observed at the storage object.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

metric

Performance numbers, such as IOPS latency and throughput

Name	Type	Description
_links	_links	
duration	string	The duration over which this sample is calculated. The time durations are represented in the ISO-8601 standard format. Samples can be calculated over the following durations:
iops	iops	The rate of I/O operations observed at the storage object.
latency	latency	The round trip latency in microseconds observed at the storage object.

Name	Type	Description
status	string	Any errors associated with the sample. For example, if the aggregation of data over multiple nodes fails then any of the partial errors might be returned, "ok" on success, or "error" on any internal uncategorized failure. Whenever a sample collection is missed but done at a later time, it is back filled to the previous 15 second timestamp and tagged with "backfilled_data". "Inconsistent_ delta_time" is encountered when the time between two collections is not the same for all nodes. Therefore, the aggregated value might be over or under inflated. "Negative_delta" is returned when an expected monotonically increasing value has decreased in value. "Inconsistent_old_data" is returned when one or more nodes do not have the latest data.
throughput	throughput	The rate of throughput bytes per second observed at the storage object.
timestamp	string	The timestamp of the performance data.

self_link

Name	Type	Description
self	href	

qos_policy

The QoS policy for the volume provisioned to host the namespace. This property is only supported when the request provisions a new volume. If no `qos_policy` is provided, a policy is be set based on the `storage_service.name`, which defaults to the most performant service available.

Name	Type	Description
_links	self_link	

Name	Type	Description
name	string	The QoS policy group name. This is mutually exclusive with UUID and other QoS attributes during POST and PATCH.
uuid	string	The QoS policy group UUID. This is mutually exclusive with name and other QoS attributes during POST and PATCH.

snapshot_policy_reference

This is a reference to the snapshot policy.

Name	Type	Description
_links	_links	
name	string	
uuid	string	

storage_service

Determines the placement of the namespace based on the value specified. Valid in POST.

Name	Type	Description
name	string	Storage service name. If not specified, the default value is the most performant for the platform.

object_stores

tiering

The tiering placement and policy definitions for the volume provisioned to host the namespace. This property is only supported when the request provisions a new volume.

Name	Type	Description
object_stores	array[object_stores]	Object stores to use. Used for placement.

Name	Type	Description
policy	string	Policy that determines whether the user data blocks of a volume in a FabricPool will be tiered to the cloud store when they become cold. FabricPool combines flash (performance tier) with a cloud store into a single aggregate. Temperature of a volume block increases if it is accessed frequently and decreases when it is not. Valid in POST or PATCH. all ‐ Allows tiering of both snapshots and active file system user data to the cloud store as soon as possible by ignoring the temperature on the volume blocks. auto ‐ Allows tiering of both snapshot and active file system user data to the cloud store none ‐ Volume blocks are not be tiered to the cloud store. snapshot_only ‐ Allows tiering of only the volume snapshots not associated with the active file system. The default tiering policy is "snapshot-only" for a FlexVol volume and "none" for a FlexGroup volume. The default minimum cooling period for the "snapshot-only" tiering policy is 2 days and for the "auto" tiering policy it is 31 days.

provisioning_options

Options that are applied to the operation.

guarantee

Properties that request and report the space guarantee for the NVMe namespace.

Name	Type	Description
requested	boolean	The requested space reservation policy for the NVMe namespace. If *true*, a space reservation is requested for the namespace; if *false*, the namespace is thin provisioned. Guaranteeing a space reservation request for a namespace requires that the volume in which the namespace resides also be space reserved and that the fractional reserve for the volume be 100%. The space reservation policy for an NVMe namespace is determined by ONTAP. • readOnly: 1 • Introduced in: 9.6 • x-nullable: true
reserved	boolean	Reports if the NVMe namespace is space guaranteed. This property is *true* if a space guarantee is requested and the containing volume and aggregate support the request. This property is *false* if a space guarantee is not requested or if a space guarantee is requested and either the containing volume and aggregate do not support the request.

space

The storage space related properties of the NVMe namespace.

Name	Type	Description
block_size	integer	The size of blocks in the namespace in bytes. The default for namespaces with an `os_type` of *vmware* is *512*. All other namespaces default to *4096*. Valid in POST when creating an NVMe namespace that is not a clone of another. Disallowed in POST when creating a namespace clone. Valid in POST.
efficiency_ratio	number	The storage efficiency ratio of the namespace without snapshots. (Logical Used / Used) This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.
guarantee	guarantee	Properties that request and report the space guarantee for the NVMe namespace.
physical_used	integer	The number of bytes consumed on the disk by the namespace, excluding snapshots. This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.
physical_used_by_snapshots	integer	The number of bytes consumed on the disk by the namespace's snapshots. This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.

Name	Type	Description
size	integer	The total provisioned size of the NVMe namespace. Valid in POST and PATCH. The NVMe namespace size can be increased but not be made smaller using the REST interface. The maximum and minimum sizes listed here are the absolute maximum and absolute minimum sizes in bytes. The maximum size is variable with respect to large NVMe namespace support in ONTAP. If large namespaces are supported, the maximum size is 128 TB (140737488355328 bytes) and if not supported, the maximum size is just under 16 TB (17557557870592 bytes). The minimum size supported is always 4096 bytes. For more information, see *Size properties* in the *docs* section of the ONTAP REST API documentation. • example: 1073741824 • format: int64 • Max value: 140737488355328 • Min value: 4096 • Introduced in: 9.6 • x-nullable: true

Name	Type	Description
used	integer	The amount of space consumed by the main data stream of the NVMe namespace. This value is the total space consumed in the volume by the NVMe namespace, including filesystem overhead, but excluding prefix and suffix streams. Due to internal filesystem overhead and the many ways NVMe filesystems and applications utilize blocks within a namespace, this value does not necessarily reflect actual consumption/availability from the perspective of the filesystem or application. Without specific knowledge of how the namespace blocks are utilized outside of ONTAP, this property should not be used and an indicator for an out-of-space condition. For more information, see *Size properties* in the *docs* section of the ONTAP REST API documentation. • format: int64 • readOnly: 1 • Introduced in: 9.6 • x-nullable: true

iops_raw

The number of I/O operations observed at the storage object. This should be used along with delta time to calculate the rate of I/O operations per unit of time.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

latency_raw

The raw latency in microseconds observed at the storage object. This should be divided by the raw IOPS value to calculate the average latency per I/O operation.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

throughput_raw

Throughput bytes observed at the storage object. This should be used along with delta time to calculate the rate of throughput bytes per unit of time.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

statistics

These are raw performance numbers, such as IOPS latency and throughput. These numbers are aggregated across all nodes in the cluster and increase with the uptime of the cluster.

Name	Type	Description
iops_raw	iops_raw	The number of I/O operations observed at the storage object. This should be used along with delta time to calculate the rate of I/O operations per unit of time.
latency_raw	latency_raw	The raw latency in microseconds observed at the storage object. This should be divided by the raw IOPS value to calculate the average latency per I/O operation.
status	string	Any errors associated with the sample. For example, if the aggregation of data over multiple nodes fails then any of the partial errors might be returned, "ok" on success, or "error" on any internal uncategorized failure. Whenever a sample collection is missed but done at a later time, it is back filled to the previous 15 second timestamp and tagged with "backfilled_data". "Inconsistent_delta_time" is encountered when the time between two collections is not the same for all nodes. Therefore, the aggregated value might be over or under inflated. "Negative_delta" is returned when an expected monotonically increasing value has decreased in value. "Inconsistent_old_data" is returned when one or more nodes do not have the latest data.
throughput_raw	throughput_raw	Throughput bytes observed at the storage object. This should be used along with delta time to calculate the rate of throughput bytes per unit of time.

Name	Type	Description
timestamp	string	The timestamp of the performance data.

status

Status information about the NVMe namespace.

Name	Type	Description
container_state	string	The state of the volume and aggregate that contain the NVMe namespace. Namespaces are only available when their containers are available.
mapped	boolean	Reports if the NVMe namespace is mapped to an NVMe subsystem. There is an added computational cost to retrieving this property's value. It is not populated for a GET request unless it is explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.
read_only	boolean	Reports if the NVMe namespace allows only read access.
state	string	The state of the NVMe namespace. Normal states for a namespace are *online* and *offline*. Other states indicate errors.

consistency_group_nvme_host_dh_hmac_chap

A container for the configuration of NVMe in-band authentication using the DH-HMAC-CHAP protocol for a host.

Name	Type	Description
group_size	string	The Diffie-Hellman group size for NVMe in-band authentication. When property `host_secret_key` is provided, this property defaults to `2048_bit`. When supplied, the property `host_secret_key` must also be supplied. Optional in POST.
hash_function	string	The hash function for NVMe in-band authentication. When property `host_secret_key` is provided, this property defaults to `sha_256`. When supplied, the property `host_secret_key` must also be supplied. Optional in POST.
mode	string	The expected NVMe in-band authentication mode for the host. This property is an indication of which secrets are configured for the host. When set to: • none: The host has neither the host nor controller secret configured, and no authentication is performed. • unidirectional: The host has a host secret configured. The controller will authenticate the host. • bidirectional: The host has both a host and controller secret configured. The controller will authenticate the host and the host will authenticate the controller.

tls

A container for the configuration for NVMe/TCP-TLS transport session for the host.

Name	Type	Description
key_type	string	The method by which the TLS pre-shared key (PSK) is configured for the host. Optional in POST. The values for property `key_type` and property `configured_psk` must logically agree. Possible values: • `none` - TLS is not configured for the host connection. No value is allowed for property `configured_psk`. • `configured` - A user supplied PSK is configured for the NVMe/TCP-TLS transport connection between the host and the NVMe subsystem. A valid value for property `configured_psk` is required. This property defaults to `none` unless a value is supplied for `configured_psk` in which case it defaults to `configured`.

hosts

The NVMe host provisioned to access NVMe namespaces mapped to a subsystem.

Name	Type	Description
dh_hmac_chap	consistency_group_nvme_host_dh_hmac_chap	A container for the configuration of NVMe in-band authentication using the DH-HMAC-CHAP protocol for a host.
nqn	string	The NVMe qualified name (NQN) used to identify the NVMe storage target.

Name	Type	Description
priority	string	The host priority setting allocates appropriate NVMe I/O queues (count and depth) for the host to submit I/O commands. Absence of this property in GET implies io_queue count and I/O queue depth are being used.
tls	tls	A container for the configuration for NVMe/TCP-TLS transport session for the host.

subsystem

The NVMe subsystem to which the NVMe namespace is mapped.

Name	Type	Description
_links	_links	
comment	string	A configurable comment for the NVMe subsystem. Optional in POST.
hosts	array[hosts]	The NVMe hosts configured for access to the NVMe subsystem. Optional in POST.
name	string	The name of the NVMe subsystem. Once created, an NVMe subsystem cannot be renamed. Optional in POST.
os_type	string	The host operating system of the NVMe subsystem's hosts. Optional in POST.
uuid	string	The unique identifier of the NVMe subsystem.

subsystem_map

The NVMe subsystem with which the NVMe namespace is associated. A namespace can be mapped to zero (0) or one (1) subsystems.

There is an added computational cost to retrieving property values for `subsystem_map`. They are not populated for a GET request unless explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more. These properties are supported only for GET.

Name	Type	Description
_links	_links	
anagrpid	string	The Asymmetric Namespace Access Group ID (ANAGRPID) of the NVMe namespace. The format for an ANAGRPID is 8 hexadecimal digits (zero-filled) followed by a lower case "h".
nsid	string	The NVMe namespace identifier. This is an identifier used by an NVMe controller to provide access to the NVMe namespace. The format for an NVMe namespace identifier is 8 hexadecimal digits (zero-filled) followed by a lower case "h".
subsystem	subsystem	The NVMe subsystem to which the NVMe namespace is mapped.

svm

SVM, applies only to SVM-scoped objects.

Name	Type	Description
_links	_links	
name	string	The name of the SVM. This field cannot be specified in a PATCH method.
uuid	string	The unique identifier of the SVM. This field cannot be specified in a PATCH method.

error_arguments

Name	Type	Description
code	string	Argument code
message	string	Message argument

returned_error

Name	Type	Description
arguments	array[error_arguments]	Message arguments
code	string	Error code
message	string	Error message
target	string	The target parameter that caused the error.

Update an NVMe namespace

PATCH `/storage/namespaces/{uuid}`

Introduced In: 9.6

Updates an NVMe namespace.

Related ONTAP commands

- `volume file clone autodelete`
- `vserver nvme namespace modify`

Learn more

- DOC /storage/namespaces

Parameters

Name	Type	In	Required	Description
uuid	string	path	True	The unique identifier of the NVMe namespace to update.

Name	Type	In	Required	Description
return_timeout	integer	query	False	The number of seconds to allow the call to execute before returning. When doing a POST, PATCH, or DELETE operation on a single record, the default is 0 seconds. This means that if an asynchronous operation is started, the server immediately returns HTTP code 202 (Accepted) along with a link to the job. If a non-zero value is specified for POST, PATCH, or DELETE operations, ONTAP waits that length of time to see if the job completes so it can return something other than 202. • Default value: 1 • Max value: 120 • Min value: 0

Request Body

Name	Type	Description
auto_delete	boolean	This property marks the NVMe namespace for auto deletion when the volume containing the namespace runs out of space. This is most commonly set on namespace clones. When set to *true*, the NVMe namespace becomes eligible for automatic deletion when the volume runs out of space. Auto deletion only occurs when the volume containing the namespace is also configured for auto deletion and free space in the volume decreases below a particular threshold. This property is optional in POST and PATCH. The default value for a new NVMe namespace is *false*. There is an added computational cost to retrieving this property's value. It is not populated for a GET request unless it is explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.
clone	clone	This sub-object is used in POST to create a new NVMe namespace as a clone of an existing namespace, or PATCH to overwrite an existing namespace as a clone of another. Setting a property in this sub-object indicates that a namespace clone is desired. When used in a PATCH, the patched NVMe namespace's data is over-written as a clone of the source and the following properties are preserved from the patched namespace unless otherwise specified as part of the PATCH: `auto_delete` (unless specified in the request), `subsystem_map`, `status.state`, and `uuid`.

Name	Type	Description
comment	string	A configurable comment available for use by the administrator. Valid in POST and PATCH.
consistency_group	consistency_group	The namespace's consistency group. This property is populated for namespaces that are members of a consistency group. If the namespace is a member of a child consistency group, the parent consistency group is reported. A namespace's consistency group is the consistency group of its containing volume.
create_time	string	The time the NVMe namespace was created.
enabled	boolean	The enabled state of the NVMe namespace. Certain error conditions cause the namespace to become disabled. If the namespace is disabled, check the `status.state` property to determine what error disabled the namespace. An NVMe namespace is enabled automatically when it is created.
name	string	The name of the NVMe namespace. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume. NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional. Renaming an NVMe namespace is not supported. Valid in POST.
space	space	The storage space related properties of the NVMe namespace.

Name	Type	Description
status	status	Status information about the NVMe namespace.
uuid	string	The unique identifier of the NVMe namespace.

Example request


```
{
  "clone": {
    "source": {
      "name": "/vol/volume1/namespace1",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    }
  },
  "comment": "string",
  "consistency_group": {
    "name": "cg1",
    "uuid": "4abc2317-4332-9d37-93a0-20bd29c22df0"
  },
  "create_time": "2018-06-04 15:00:00 -0400",
  "name": "/vol/volume1/qtree1/namespace1",
  "space": {
    "efficiency_ratio": 2.5,
    "physical_used": 1073741824,
    "physical_used_by_snapshots": 1073741824,
    "size": 1073741824,
    "used": 0
  },
  "status": {
    "container_state": "string",
    "state": "online"
  },
  "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
}
```


Response

```
Status: 200, Ok
```

Response

```
Status: 202, Accepted
```

Error

```
Status: Default
```

ONTAP Error Response Codes

Error Code	Description
5374127	The specified namespace name is invalid.
5376461	The specified namespace name is invalid.
5376462	The specified namespace name is too long.
5376463	The snapshot portion of the specified namespace name is too long.
5376466	An attempt was made to rename an NVMe namespace to a snapshot name.
5376467	An attempt was made to rename a primary NVMe namespace to a secondary name.
5376468	An attempt was made to rename an NVMe namespace to a reserved name.
13565952	The namespace clone request failed.
72089724	The specified namespace size is too large.
72089730	The specified namespace cannot be updated as it resides in a snapshot.
72090005	The specified `clone.source.uuid` and `clone.source.name` do not refer to the same LUN.
72090006	The specified namespace was not found. This can apply to `clone.source` or the target namespace. The `target` property of the error object identifies the property.
72090007	The specified namespace was not found. This can apply to `clone.source` or the target namespace. The `target` property of the error object identifies the property.
72090010	An error occurred after successfully overwriting data for the namespace as a clone. Some properties were not modified.

Error Code	Description
72090011	An error occurred after successfully modifying some of the properties of the namespace. Some properties were not modified.
72090016	The namespace's aggregate is offline. The aggregate must be online to modify or remove the namespace.
72090017	The namespace's volume is offline. The volume must be online to modify or remove the namespace.
72090038	An attempt was made to reduce the size of the specified namespace.

Also see the table of common errors in the Response body overview section of this documentation.

Definitions

See Definitions

href

Name	Type	Description
href	string	

_links

source

The source NVMe namespace for a namespace clone operation. This can be specified using property `clone.source.uuid` or `clone.source.name`. If both properties are supplied, they must refer to the same namespace.

Valid in POST to create a new NVMe namespace as a clone of the source.

Valid in PATCH to overwrite an existing NVMe namespace's data as a clone of another.

Name	Type	Description
name	string	The name of the clone source NVMe namespace. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume. NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional. Valid in POST and PATCH.
uuid	string	The unique identifier of the clone source NVMe namespace. Valid in POST and PATCH.

clone

This sub-object is used in POST to create a new NVMe namespace as a clone of an existing namespace, or PATCH to overwrite an existing namespace as a clone of another. Setting a property in this sub-object indicates that a namespace clone is desired.

When used in a PATCH, the patched NVMe namespace's data is over-written as a clone of the source and the following properties are preserved from the patched namespace unless otherwise specified as part of the PATCH: `auto_delete` (unless specified in the request), `subsystem_map`, `status.state`, and `uuid`.

Name	Type	Description
source	source	The source NVMe namespace for a namespace clone operation. This can be specified using property `clone.source.uuid` or `clone.source.name`. If both properties are supplied, they must refer to the same namespace. Valid in POST to create a new NVMe namespace as a clone of the source. Valid in PATCH to overwrite an existing NVMe namespace's data as a clone of another.

consistency_group

The namespace's consistency group. This property is populated for namespaces that are members of a consistency group. If the namespace is a member of a child consistency group, the parent consistency group is reported. A namespace's consistency group is the consistency group of its containing volume.

Name	Type	Description
name	string	The name of the consistency group.
uuid	string	The unique identifier of the consistency group.

lun

The source LUN for convert operation. This can be specified using property `convert.lun.uuid` or `convert.lun.name`. If both properties are supplied, they must refer to the same LUN.

Valid in POST. A convert request from LUN to NVMe namespace cannot be combined with setting any other namespace properties. All other properties of the converted NVMe namespace come from the source LUN.

convert

This sub-object is used in POST to convert a valid in-place LUN to an NVMe namespace. Setting a property in this sub-object indicates that a conversion from the specified LUN to NVMe namespace is desired.

node

The cluster node that hosts the NVMe namespace.

Name	Type	Description
name	string	
uuid	string	

qtree

The qtree in which the NVMe namespace is optionally located. Valid in POST.

If properties `name` and `location.qtree.name` and/or `location.qtree.uuid` are specified in the same request, they must refer to the same qtree.

NVMe namespaces do not support rename.

Name	Type	Description
id	integer	The identifier for the qtree, unique within the qtree's volume.
name	string	The name of the qtree.

volume

The volume in which the NVMe namespace is located. Valid in POST.

If properties `name` and `location.volume.name` and/or `location.volume.uuid` are specified in the same request, they must refer to the same volume.

NVMe namespaces do not support movement between volumes.

Name	Type	Description
name	string	The name of the volume. This field cannot be specified in a PATCH method.
uuid	string	Unique identifier for the volume. This corresponds to the instance-uuid that is exposed in the CLI and ONTAPI. It does not change due to a volume move. • example: 028baa66-41bd-11e9-81d5-00a0986138f7 • Introduced in: 9.6 • x-nullable: true

location

The location of the NVMe namespace within the ONTAP cluster. NVMe namespaces do not support rename, or movement between volumes. Valid in POST.

Name	Type	Description
node	node	The cluster node that hosts the NVMe namespace.

iops

The rate of I/O operations observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

latency

The round trip latency in microseconds observed at the storage object.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

throughput

The rate of throughput bytes per second observed at the storage object.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

metric

Performance numbers, such as IOPS latency and throughput

Name	Type	Description
duration	string	The duration over which this sample is calculated. The time durations are represented in the ISO-8601 standard format. Samples can be calculated over the following durations:
iops	iops	The rate of I/O operations observed at the storage object.
latency	latency	The round trip latency in microseconds observed at the storage object.

Name	Type	Description
status	string	Any errors associated with the sample. For example, if the aggregation of data over multiple nodes fails then any of the partial errors might be returned, "ok" on success, or "error" on any internal uncategorized failure. Whenever a sample collection is missed but done at a later time, it is back filled to the previous 15 second timestamp and tagged with "backfilled_data". "Inconsistent_ delta_time" is encountered when the time between two collections is not the same for all nodes. Therefore, the aggregated value might be over or under inflated. "Negative_delta" is returned when an expected monotonically increasing value has decreased in value. "Inconsistent_old_data" is returned when one or more nodes do not have the latest data.
throughput	throughput	The rate of throughput bytes per second observed at the storage object.
timestamp	string	The timestamp of the performance data.

self_link

qos_policy

The QoS policy for the volume provisioned to host the namespace. This property is only supported when the request provisions a new volume. If no `qos_policy` is provided, a policy is be set based on the `storage_service.name`, which defaults to the most performant service available.

Name	Type	Description
name	string	The QoS policy group name. This is mutually exclusive with UUID and other QoS attributes during POST and PATCH.

Name	Type	Description
uuid	string	The QoS policy group UUID. This is mutually exclusive with name and other QoS attributes during POST and PATCH.

snapshot_policy_reference

This is a reference to the snapshot policy.

Name	Type	Description
name	string	
uuid	string	

storage_service

Determines the placement of the namespace based on the value specified. Valid in POST.

Name	Type	Description
name	string	Storage service name. If not specified, the default value is the most performant for the platform.

object_stores

Name	Type	Description
name	string	The name of the object store to use. Used for placement.

tiering

The tiering placement and policy definitions for the volume provisioned to host the namespace. This property is only supported when the request provisions a new volume.

Name	Type	Description
control	string	Storage tiering placement rules for the object.
object_stores	array[object_stores]	Object stores to use. Used for placement.

provisioning_options

Options that are applied to the operation.

guarantee

Properties that request and report the space guarantee for the NVMe namespace.

Name	Type	Description
requested	boolean	The requested space reservation policy for the NVMe namespace. If *true*, a space reservation is requested for the namespace; if *false*, the namespace is thin provisioned. Guaranteeing a space reservation request for a namespace requires that the volume in which the namespace resides also be space reserved and that the fractional reserve for the volume be 100%. The space reservation policy for an NVMe namespace is determined by ONTAP. • readOnly: 1 • Introduced in: 9.6 • x-nullable: true
reserved	boolean	Reports if the NVMe namespace is space guaranteed. This property is *true* if a space guarantee is requested and the containing volume and aggregate support the request. This property is *false* if a space guarantee is not requested or if a space guarantee is requested and either the containing volume and aggregate do not support the request.

space

The storage space related properties of the NVMe namespace.

Name	Type	Description
efficiency_ratio	number	The storage efficiency ratio of the namespace without snapshots. (Logical Used / Used) This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.
guarantee	guarantee	Properties that request and report the space guarantee for the NVMe namespace.
physical_used	integer	The number of bytes consumed on the disk by the namespace, excluding snapshots. This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.
physical_used_by_snapshots	integer	The number of bytes consumed on the disk by the namespace's snapshots. This property is not available on the namespace object in the REST API and is not reported for GET requests. See the containing volume object for this information.

Name	Type	Description
size	integer	The total provisioned size of the NVMe namespace. Valid in POST and PATCH. The NVMe namespace size can be increased but not be made smaller using the REST interface. The maximum and minimum sizes listed here are the absolute maximum and absolute minimum sizes in bytes. The maximum size is variable with respect to large NVMe namespace support in ONTAP. If large namespaces are supported, the maximum size is 128 TB (140737488355328 bytes) and if not supported, the maximum size is just under 16 TB (17557557870592 bytes). The minimum size supported is always 4096 bytes. For more information, see *Size properties* in the *docs* section of the ONTAP REST API documentation. • example: 1073741824 • format: int64 • Max value: 140737488355328 • Min value: 4096 • Introduced in: 9.6 • x-nullable: true

Name	Type	Description
used	integer	The amount of space consumed by the main data stream of the NVMe namespace. This value is the total space consumed in the volume by the NVMe namespace, including filesystem overhead, but excluding prefix and suffix streams. Due to internal filesystem overhead and the many ways NVMe filesystems and applications utilize blocks within a namespace, this value does not necessarily reflect actual consumption/availability from the perspective of the filesystem or application. Without specific knowledge of how the namespace blocks are utilized outside of ONTAP, this property should not be used and an indicator for an out-of-space condition. For more information, see *Size properties* in the *docs* section of the ONTAP REST API documentation. • format: int64 • readOnly: 1 • Introduced in: 9.6 • x-nullable: true

iops_raw

The number of I/O operations observed at the storage object. This should be used along with delta time to calculate the rate of I/O operations per unit of time.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

latency_raw

The raw latency in microseconds observed at the storage object. This should be divided by the raw IOPS value to calculate the average latency per I/O operation.

Name	Type	Description
other	integer	Performance metric for other I/O operations. Other I/O operations can be metadata operations, such as directory lookups and so on.
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

throughput_raw

Throughput bytes observed at the storage object. This should be used along with delta time to calculate the rate of throughput bytes per unit of time.

Name	Type	Description
read	integer	Performance metric for read I/O operations.
total	integer	Performance metric aggregated over all types of I/O operations.
write	integer	Performance metric for write I/O operations.

statistics

These are raw performance numbers, such as IOPS latency and throughput. These numbers are aggregated across all nodes in the cluster and increase with the uptime of the cluster.

Name	Type	Description
iops_raw	iops_raw	The number of I/O operations observed at the storage object. This should be used along with delta time to calculate the rate of I/O operations per unit of time.
latency_raw	latency_raw	The raw latency in microseconds observed at the storage object. This should be divided by the raw IOPS value to calculate the average latency per I/O operation.
status	string	Any errors associated with the sample. For example, if the aggregation of data over multiple nodes fails then any of the partial errors might be returned, "ok" on success, or "error" on any internal uncategorized failure. Whenever a sample collection is missed but done at a later time, it is back filled to the previous 15 second timestamp and tagged with "backfilled_data". "Inconsistent_delta_time" is encountered when the time between two collections is not the same for all nodes. Therefore, the aggregated value might be over or under inflated. "Negative_delta" is returned when an expected monotonically increasing value has decreased in value. "Inconsistent_old_data" is returned when one or more nodes do not have the latest data.
throughput_raw	throughput_raw	Throughput bytes observed at the storage object. This should be used along with delta time to calculate the rate of throughput bytes per unit of time.

Name	Type	Description
timestamp	string	The timestamp of the performance data.

status

Status information about the NVMe namespace.

Name	Type	Description
container_state	string	The state of the volume and aggregate that contain the NVMe namespace. Namespaces are only available when their containers are available.
mapped	boolean	Reports if the NVMe namespace is mapped to an NVMe subsystem. There is an added computational cost to retrieving this property's value. It is not populated for a GET request unless it is explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.
read_only	boolean	Reports if the NVMe namespace allows only read access.
state	string	The state of the NVMe namespace. Normal states for a namespace are *online* and *offline*. Other states indicate errors.

consistency_group_nvme_host_dh_hmac_chap

A container for the configuration of NVMe in-band authentication using the DH-HMAC-CHAP protocol for a host.

Name	Type	Description
mode	string	The expected NVMe in-band authentication mode for the host. This property is an indication of which secrets are configured for the host. When set to: • none: The host has neither the host nor controller secret configured, and no authentication is performed. • unidirectional: The host has a host secret configured. The controller will authenticate the host. • bidirectional: The host has both a host and controller secret configured. The controller will authenticate the host and the host will authenticate the controller.

tls

A container for the configuration for NVMe/TCP-TLS transport session for the host.

hosts

The NVMe host provisioned to access NVMe namespaces mapped to a subsystem.

subsystem

The NVMe subsystem to which the NVMe namespace is mapped.

subsystem_map

The NVMe subsystem with which the NVMe namespace is associated. A namespace can be mapped to zero (0) or one (1) subsystems.

There is an added computational cost to retrieving property values for `subsystem_map`. They are not populated for a GET request unless explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more. These properties are supported only for GET.

Name	Type	Description
anagrpid	string	The Asymmetric Namespace Access Group ID (ANAGRPID) of the NVMe namespace. The format for an ANAGRPID is 8 hexadecimal digits (zero-filled) followed by a lower case "h".
nsid	string	The NVMe namespace identifier. This is an identifier used by an NVMe controller to provide access to the NVMe namespace. The format for an NVMe namespace identifier is 8 hexadecimal digits (zero-filled) followed by a lower case "h".

svm

SVM, applies only to SVM-scoped objects.

Name	Type	Description
name	string	The name of the SVM. This field cannot be specified in a PATCH method.
uuid	string	The unique identifier of the SVM. This field cannot be specified in a PATCH method.

nvme_namespace

An NVMe namespace is a collection of addressable logical blocks presented to hosts connected to the storage virtual machine using the NVMe over Fabrics protocol.

An NVMe namespace must be mapped to an NVMe subsystem to grant access to the subsystem’s hosts. Hosts can then access the NVMe namespace and perform I/O using the NVMe over Fabrics protocol.

See the NVMe namespace object model to learn more about each of the properties supported by the NVMe namespace REST API. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume.

NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional.

An NVMe namespace is created to a specified size using thin or thick provisioning as determined by the volume on which it is created. An NVMe namespace can then be resized or cloned. An NVMe namespace cannot be renamed, or moved to a different volume. NVMe namespaces do not support the assignment of

a QoS policy for performance management, but a QoS policy can be assigned to the volume containing the namespace.

</namespace></qtree></volume>

Name	Type	Description
auto_delete	boolean	This property marks the NVMe namespace for auto deletion when the volume containing the namespace runs out of space. This is most commonly set on namespace clones. When set to *true*, the NVMe namespace becomes eligible for automatic deletion when the volume runs out of space. Auto deletion only occurs when the volume containing the namespace is also configured for auto deletion and free space in the volume decreases below a particular threshold. This property is optional in POST and PATCH. The default value for a new NVMe namespace is *false*. There is an added computational cost to retrieving this property's value. It is not populated for a GET request unless it is explicitly requested using the `fields` query parameter. See Requesting specific fields to learn more.

Name	Type	Description
clone	clone	This sub-object is used in POST to create a new NVMe namespace as a clone of an existing namespace, or PATCH to overwrite an existing namespace as a clone of another. Setting a property in this sub-object indicates that a namespace clone is desired. When used in a PATCH, the patched NVMe namespace's data is over-written as a clone of the source and the following properties are preserved from the patched namespace unless otherwise specified as part of the PATCH: `auto_delete` (unless specified in the request), `subsystem_map`, `status.state`, and `uuid`.
comment	string	A configurable comment available for use by the administrator. Valid in POST and PATCH.
consistency_group	consistency_group	The namespace's consistency group. This property is populated for namespaces that are members of a consistency group. If the namespace is a member of a child consistency group, the parent consistency group is reported. A namespace's consistency group is the consistency group of its containing volume.
create_time	string	The time the NVMe namespace was created.

Name	Type	Description
enabled	boolean	The enabled state of the NVMe namespace. Certain error conditions cause the namespace to become disabled. If the namespace is disabled, check the `status.state` property to determine what error disabled the namespace. An NVMe namespace is enabled automatically when it is created.
name	string	The name of the NVMe namespace. An NVMe namespace is located within a volume. Optionally, it can be located within a qtree in a volume. NVMe namespace names are paths of the form "/vol/<volume>[/<qtree>]/<namespace>" where the qtree name is optional. Renaming an NVMe namespace is not supported. Valid in POST.
space	space	The storage space related properties of the NVMe namespace.
status	status	Status information about the NVMe namespace.
uuid	string	The unique identifier of the NVMe namespace.

error_arguments

Name	Type	Description
code	string	Argument code
message	string	Message argument

returned_error

Name	Type	Description
arguments	array[error_arguments]	Message arguments
code	string	Error code
message	string	Error message
target	string	The target parameter that caused the error.

Copyright information

Trademark information