# Manage FC network interfaces

## REST API reference

NetApp
September 12, 2025

# Table of Contents

# Manage FC network interfaces

## Network FC interfaces endpoint overview

### Overview

Fibre Channel (FC) interfaces are the logical endpoints for FC network connections to an SVM. An FC interface provides FC access to storage within the interface SVM using either Fibre Channel Protocol or NVMe over FC (NVMe/FC).

The Fibre Channel interface REST API allows you to create, delete, update, and discover FC interfaces, and obtain status information for FC interfaces.

An FC interface is created on an FC port which is located on a cluster node. The FC port must be specified to identify the location of the interface for a POST or PATCH request that relocates an interface. You can identify the port by supplying either the node and port names or the port UUID.

### Performance monitoring

Performance of an FC interface can be monitored by observing the `metric.*` and `statistics.*` properties. These properties show the performance of an FC interface 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 FC interface using the port node and name to identify the location

This example uses the `return_records` query parameter to retrieve the newly created FC interface in the POST response.

```
# The API:
POST /api/network/fc/interfaces

# The call:
curl -X POST 'https://<mgmt-
ip>/api/network/fc/interfaces?return_records=true' -H 'accept:
application/hal+json' -d '{ "svm": { "name": "svm1" }, "name": "lif1",
"location": { "home_port": { "name": "0a", "home_node": { "name": "node1"
} } }, "data_protocol": "fcp" }'

# The response:
{
"num_records": 1,
"records": [
  {
    "svm": {
      "uuid": "cf300f5c-db83-11e8-bd46-005056bba0e0",
      "name": "svm1",
      "_links": {
        "self": {
          "href": "/api/svm/svms/cf300f5c-db83-11e8-bd46-005056bba0e0"
        }
      }
    },
    "uuid": "f6045b92-dec7-11e8-a733-005056bba0e0",
    "name": "lif1",
    "location": {
      "home_node": {
        "uuid": "bafe9b9f-db81-11e8-bd46-005056bba0e0",
        "name": "node1",
        "_links": {
          "self": {
            "href": "/api/cluster/nodes/bafe9b9f-db81-11e8-bd46-
005056bba0e0"
          }
        }
      },
      "home_port": {
        "uuid": "300c1ae3-db82-11e8-bd46-005056bba0e0",
        "name": "0a",
        "node": {
          "name": "node1"
        },
        "_links": {
          "self": {
            "href": "/api/network/fc/ports/300c1ae3-db82-11e8-bd46-
005056bba0e0"
          }
        }
      },
      "node": {
        "uuid": "bafe9b9f-db81-11e8-bd46-005056bba0e0",
        "name": "node1",
        "_links": {
          "self": {
            "href": "/api/cluster/nodes/bafe9b9f-db81-11e8-bd46-
005056bba0e0"
          }
        }
      },
      "port": {
        "uuid": "300c1ae3-db82-11e8-bd46-005056bba0e0",
        "name": "0a",
        "node": {
          "name": "node1"
        },
        "_links": {
          "self": {
            "href": "/api/network/fc/ports/300c1ae3-db82-11e8-bd46-
005056bba0e0"
          }
        }
      }
    },
    "enabled": true,
    "state": "down",
    "data_protocol": "fcp",
    "wwpn": "20:04:00:50:56:bb:a0:e0",
    "wwnn": "20:00:00:50:56:bb:a0:e0",
    "port_address": "9da2cb1",
    "_links": {
      "self": {
        "href": "/api/network/fc/interfaces/f6045b92-dec7-11e8-a733-
005056bba0e0"
      }
    }
  }
]
}
```

---

### Creating an FC interface using the port UUID to identify the location

This example uses the `return_records` query parameter to retrieve the newly created FC interface in the POST response.

```
# The API:
POST /api/network/fc/interfaces

# The call:
curl -X POST 'https://<mgmt-
ip>/api/network/fc/interfaces?return_records=true' -H 'accept:
application/hal+json' -d '{ "svm": { "name": "svm3" }, "name": "lif2",
"location": { "home_port": { "uuid": "24bb636a-db83-11e8-9a49-
005056bb1ec6" } }, "data_protocol": "fc_nvme" }'

# The response:
{
```

```
"num_records": 1,
"records": [
  {
    "svm": {
      "uuid": "a5060466-dbab-11e8-bd46-005056bba0e0",
      "name": "svm3",
      "_links": {
        "self": {
          "href": "/api/svm/svms/a5060466-dbab-11e8-bd46-005056bba0e0"
        }
      }
    },
    "uuid": "cdeb5591-dec9-11e8-a733-005056bba0e0",
    "name": "lif2",
    "location": {
      "home_node": {
        "uuid": "e85aa147-db83-11e8-9a48-005056bb1ec6",
        "name": "node3",
        "_links": {
          "self": {
            "href": "/api/cluster/nodes/e85aa147-db83-11e8-9a48-
005056bb1ec6"
          }
        }
      },
      "home_port": {
        "uuid": "24bb636a-db83-11e8-9a49-005056bb1ec6",
        "name": "1b",
        "node": {
          "name": "node3"
        },
        "_links": {
          "self": {
            "href": "/api/network/fc/ports/24bb636a-db83-11e8-9a49-
005056bb1ec6"
          }
        }
      },
      "node": {
        "uuid": "e85aa147-db83-11e8-9a48-005056bb1ec6",
        "name": "node3",
        "_links": {
          "self": {
            "href": "/api/cluster/nodes/e85aa147-db83-11e8-9a48-
005056bb1ec6"
          }
        }
      },
      "port": {
        "uuid": "24bb636a-db83-11e8-9a49-005056bb1ec6",
        "name": "1b",
        "node": {
          "name": "node3"
        },
        "_links": {
          "self": {
            "href": "/api/network/fc/ports/24bb636a-db83-11e8-9a49-
005056bb1ec6"
          }
        }
      }
    },
    "enabled": true,
    "state": "down",
    "data_protocol": "fc_nvme",
    "wwpn": "20:05:00:50:56:bb:a0:e0",
    "wwnn": "20:02:00:50:56:bb:a0:e0",
    "port_address": "612e202b",
    "_links": {
      "self": {
        "href": "/api/network/fc/interfaces/cdeb5591-dec9-11e8-a733-
005056bba0e0"
      }
    }
  }
]
}
```

### Retrieving all properties for all FC interfaces

This example uses the `fields` query parameter to retrieve all properties.

```
# The API:
GET /api/network/fc/interfaces

# The call:
curl -X GET 'https://<mgmt-ip>/api/network/fc/interfaces?fields=*' -H
'accept: application/hal+json'

# The response:
```

```
{
"records": [
  {
    "svm": {
      "uuid": "a5060466-dbab-11e8-bd46-005056bba0e0",
      "name": "svm3",
      "_links": {
        "self": {
          "href": "/api/svm/svms/a5060466-dbab-11e8-bd46-005056bba0e0"
        }
      }
    },
    "uuid": "cdeb5591-dec9-11e8-a733-005056bba0e0",
    "name": "lif2",
    "location": {
      "home_node": {
        "uuid": "e85aa147-db83-11e8-9a48-005056bb1ec6",
        "name": "node3",
        "_links": {
          "self": {
            "href": "/api/cluster/nodes/e85aa147-db83-11e8-9a48-
005056bb1ec6"
          }
        }
      },
      "home_port": {
        "uuid": "24bb636a-db83-11e8-9a49-005056bb1ec6",
        "name": "1b",
        "node": {
          "name": "node3"
        },
        "_links": {
          "self": {
            "href": "/api/network/fc/ports/24bb636a-db83-11e8-9a49-
005056bb1ec6"
          }
        }
      },
      "node": {
        "uuid": "e85aa147-db83-11e8-9a48-005056bb1ec6",
        "name": "node3",
        "_links": {
          "self": {
            "href": "/api/cluster/nodes/e85aa147-db83-11e8-9a48-
005056bb1ec6"
          }
        }
      },
      "port": {
        "uuid": "24bb636a-db83-11e8-9a49-005056bb1ec6",
        "name": "1b",
        "node": {
          "name": "node3"
        },
        "_links": {
          "self": {
            "href": "/api/network/fc/ports/24bb636a-db83-11e8-9a49-
005056bb1ec6"
          }
        }
      }
    },
    "enabled": true,
    "state": "down",
    "data_protocol": "fc_nvme",
    "wwpn": "20:05:00:50:56:bb:a0:e0",
    "wwnn": "20:02:00:50:56:bb:a0:e0",
    "port_address": "612e202b",
    "_links": {
      "self": {
        "href": "/api/network/fc/interfaces/cdeb5591-dec9-11e8-a733-
005056bba0e0"
      }
    }
  },
  {
    "svm": {
      "uuid": "cf300f5c-db83-11e8-bd46-005056bba0e0",
      "name": "svm1",
      "_links": {
        "self": {
          "href": "/api/svm/svms/cf300f5c-db83-11e8-bd46-005056bba0e0"
        }
      }
    },
    "uuid": "f6045b92-dec7-11e8-a733-005056bba0e0",
    "name": "lif1",
    "location": {
      "home_node": {
        "uuid": "bafe9b9f-db81-11e8-bd46-005056bba0e0",
        "name": "node1",
        "_links": {
          "self": {
            "href": "/api/cluster/nodes/bafe9b9f-db81-11e8-bd46-
005056bba0e0"
          }
        }
      },
      "home_port": {
        "uuid": "300c1ae3-db82-11e8-bd46-005056bba0e0",
        "name": "0a",
        "node": {
          "name": "node1"
        },
        "_links": {
          "self": {
            "href": "/api/network/fc/ports/300c1ae3-db82-11e8-bd46-
005056bba0e0"
          }
        }
      },
      "node": {
        "uuid": "bafe9b9f-db81-11e8-bd46-005056bba0e0",
        "name": "node1",
        "_links": {
          "self": {
            "href": "/api/cluster/nodes/bafe9b9f-db81-11e8-bd46-
005056bba0e0"
          }
        }
      },
      "port": {
        "uuid": "300c1ae3-db82-11e8-bd46-005056bba0e0",
        "name": "0a",
        "node": {
          "name": "node1"
        },
        "_links": {
          "self": {
            "href": "/api/network/fc/ports/300c1ae3-db82-11e8-bd46-
005056bba0e0"
          }
        }
      }
    },
    "enabled": true,
    "state": "down",
    "data_protocol": "fcp",
  "wwpn": "20:04:00:50:56:bb:a0:e0",
  "wwnn": "20:00:00:50:56:bb:a0:e0",
  "port_address": "9da2cb1",
  "_links": {
   "self": {
    "href": "/api/network/fc/interfaces/f6045b92-dec7-11e8-a733-
005056bba0e0"
   }
  }
 }
],
"num_records": 2,
"_links": {
 "self": {
  "href": "/api/network/fc/interfaces?fields=*"
 }
}
}
```

### Retrieving a list of selected FC interfaces

This example uses property query parameters to retrieve FC interfaces configured for the FC Protocol that are set to *up*.

```
# The API:
GET /api/network/fc/interfaces

# The call:
curl -X GET 'https://<mgmt-
ip>/api/network/fc/interfaces?data_protocol=fcp&state=up' -H 'accept:
application/hal+json'

# The response:
{
"records": [
  {
    "svm": {
      "uuid": "cf300f5c-db83-11e8-bd46-005056bba0e0",
      "name": "svm1",
      "_links": {
        "self": {
          "href": "/api/svm/svms/cf300f5c-db83-11e8-bd46-005056bba0e0"
        }
      }
    },
    "uuid": "f6045b92-dec7-11e8-a733-005056bba0e0",
    "name": "lif1",
    "state": "up",
    "data_protocol": "fcp",
    "_links": {
      "self": {
        "href": "/api/network/fc/interfaces/f6045b92-dec7-11e8-a733-
005056bba0e0"
      }
    }
  }
],
"num_records": 1,
"_links": {
  "self": {
    "href": "/api/network/fc/interfaces?data_protocol=fcp&state=up"
  }
}
}
```

### Retrieving a specific FC interface

```
# The API:
GET /api/network/fc/interfaces/{uuid}

# The call:
curl -X GET 'https://<mgmt-ip>/api/network/fc/interfaces/cdeb5591-dec9-
11e8-a733-005056bba0e0' -H 'accept: application/hal+json'

# The response:
{
"svm": {
  "uuid": "a5060466-dbab-11e8-bd46-005056bba0e0",
  "name": "svm3",
  "_links": {
    "self": {
      "href": "/api/svm/svms/a5060466-dbab-11e8-bd46-005056bba0e0"
    }
  }
},
"uuid": "cdeb5591-dec9-11e8-a733-005056bba0e0",
"name": "lif2",
"location": {
  "home_node": {
    "uuid": "e85aa147-db83-11e8-9a48-005056bb1ec6",
    "name": "node3",
    "_links": {
      "self": {
        "href": "/api/cluster/nodes/e85aa147-db83-11e8-9a48-005056bb1ec6"
      }
    }
  },
  "home_port": {
    "uuid": "24bb636a-db83-11e8-9a49-005056bb1ec6",
    "name": "1b",
    "node": {
      "name": "node3"
    },
    "_links": {
      "self": {
        "href": "/api/network/fc/ports/24bb636a-db83-11e8-9a49-
005056bb1ec6"
      }
    }
  },
  "node": {
    "uuid": "e85aa147-db83-11e8-9a48-005056bb1ec6",
    "name": "node3",
    "_links": {
      "self": {
        "href": "/api/cluster/nodes/e85aa147-db83-11e8-9a48-005056bb1ec6"
      }
    }
  },
  "port": {
    "uuid": "24bb636a-db83-11e8-9a49-005056bb1ec6",
    "name": "1b",
    "node": {
      "name": "node3"
    },
    "_links": {
      "self": {
        "href": "/api/network/fc/ports/24bb636a-db83-11e8-9a49-
005056bb1ec6"
      }
    }
  }
},
"enabled": true,
"state": "down",
"data_protocol": "fc_nvme",
"wwpn": "20:05:00:50:56:bb:a0:e0",
"wwnn": "20:02:00:50:56:bb:a0:e0",
"port_address": "612e202b",
"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/network/fc/interfaces/cdeb5591-dec9-11e8-a733-
005056bba0e0"
  }
}
}
```

## Disabling an FC interface

When updating certain properties or deleting an FC interface, the interface must first be disabled using the following:

```
# The API:
PATCH /api/network/fc/interfaces/{uuid}

# The call:
curl -X PATCH 'https://<mgmt-ip>/api/network/fc/interfaces/f6045b92-dec7-
11e8-a733-005056bba0e0' -H 'accept: application/hal+json' -d '{ "enabled":
false }'
```

### Moving an FC interface to a new node and port

To move an FC interface to another node or port, the destination FC port must be specified in a PATCH request. Either the port UUID or node and port names can be used to identify the port.

Note that only FC interfaces configured for the FC Protocol can be moved. FC interfaces configured for NVMe/FC cannot be moved. The interface must also be set to the disabled state before being moved.

```
# The API:
PATCH /api/network/fc/interfaces/{uuid}

# The call:
curl -X PATCH 'https://<mgmt-ip>/api/network/fc/interfaces/f6045b92-dec7-
11e8-a733-005056bba0e0' -H 'accept: application/hal+json' -d '{
"location": { "home_port": { "uuid": "a1dc7aa5-db83-11e8-9ef7-
005056bbbbcc" } } }'
```

### Deleting an FC interface

The FC interface must be disabled before being deleted.

```
# The API:
DELETE /api/network/fc/interfaces/{uuid}

# The call:
curl -X DELETE 'https://<mgmt-ip>/api/network/fc/interfaces/f6045b92-dec7-
11e8-a733-005056bba0e0' -H 'accept: application/hal+json'
```

# Retrieve FC interfaces

GET `/network/fc/interfaces`

**Introduced In:** 9.6

Retrieves FC interfaces.

## Related ONTAP commands

- `network interface show`
- `vserver fcp interface show`

## Learn more

- DOC /network/fc/interfaces

## Parameters

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| location.port.name | string | query | False | Filter by location.port.name |
| location.port.node.name | string | query | False | Filter by location.port.node.name |
| location.port.uuid | string | query | False | Filter by location.port.uuid |
| location.home_node.name | string | query | False | Filter by location.home_node.name<br>• Introduced in: 9.8 |
| location.home_node.uuid | string | query | False | Filter by location.home_node.uuid<br>• Introduced in: 9.8 |
| location.home_port.name | string | query | False | Filter by location.home_port.name<br>• Introduced in: 9.8 |

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| location.home_port.node.name | string | query | False | Filter by location.home_port.node.name<br>• Introduced in: 9.8 |
| location.home_port.uuid | string | query | False | Filter by location.home_port.uuid<br>• Introduced in: 9.8 |
| location.node.name | string | query | False | Filter by location.node.name |
| location.node.uuid | string | query | False | Filter by location.node.uuid |
| location.is_home | boolean | query | False | Filter by location.is_home<br>• Introduced in: 9.8 |
| enabled | boolean | query | False | Filter by enabled |
| port_address | string | query | False | Filter by port_address |
| uuid | string | query | False | Filter by uuid |
| data_protocol | string | query | False | Filter by data_protocol |
| metric.latency.total | integer | query | False | Filter by metric.latency.total<br>• Introduced in: 9.8 |
| metric.latency.read | integer | query | False | Filter by metric.latency.read<br>• Introduced in: 9.8 |

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| metric.latency.other | integer | query | False | Filter by metric.latency.other<br>- Introduced in: 9.8 |
| metric.latency.write | integer | query | False | Filter by metric.latency.write<br>- Introduced in: 9.8 |
| metric.status | string | query | False | Filter by metric.status<br>- Introduced in: 9.8 |
| metric.iops.total | integer | query | False | Filter by metric.iops.total<br>- Introduced in: 9.8 |
| metric.iops.read | integer | query | False | Filter by metric.iops.read<br>- Introduced in: 9.8 |
| metric.iops.other | integer | query | False | Filter by metric.iops.other<br>- Introduced in: 9.8 |
| metric.iops.write | integer | query | False | Filter by metric.iops.write<br>- Introduced in: 9.8 |
| metric.duration | string | query | False | Filter by metric.duration<br>- Introduced in: 9.8 |

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| metric.timestamp | string | query | False | Filter by metric.timestamp<br>- Introduced in: 9.8 |
| metric.throughput.write | integer | query | False | Filter by metric.throughput.write<br>- Introduced in: 9.8 |
| metric.throughput.read | integer | query | False | Filter by metric.throughput.read<br>- Introduced in: 9.8 |
| metric.throughput.total | integer | query | False | Filter by metric.throughput.total<br>- Introduced in: 9.8 |
| name | string | query | False | Filter by name |
| comment | string | query | False | Filter by comment |
| statistics.iops_raw.total | integer | query | False | Filter by statistics.iops_raw.total<br>- Introduced in: 9.8 |
| statistics.iops_raw.read | integer | query | False | Filter by statistics.iops_raw.read<br>- Introduced in: 9.8 |

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| statistics.iops_raw.other | integer | query | False | Filter by statistics.iops_raw.other<br>• Introduced in: 9.8 |
| statistics.iops_raw.write | integer | query | False | Filter by statistics.iops_raw.write<br>• Introduced in: 9.8 |
| statistics.latency_raw.total | integer | query | False | Filter by statistics.latency_raw.total<br>• Introduced in: 9.8 |
| statistics.latency_raw.read | integer | query | False | Filter by statistics.latency_raw.read<br>• Introduced in: 9.8 |
| statistics.latency_raw.other | integer | query | False | Filter by statistics.latency_raw.other<br>• Introduced in: 9.8 |
| statistics.latency_raw.write | integer | query | False | Filter by statistics.latency_raw.write<br>• Introduced in: 9.8 |
| statistics.timestamp | string | query | False | Filter by statistics.timestamp<br>• Introduced in: 9.8 |

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| statistics.status | string | query | False | Filter by statistics.status<br>- Introduced in: 9.8 |
| statistics.throughput_raw.write | integer | query | False | Filter by statistics.throughput_raw.write<br>- Introduced in: 9.8 |
| statistics.throughput_raw.read | integer | query | False | Filter by statistics.throughput_raw.read<br>- Introduced in: 9.8 |
| statistics.throughput_raw.total | integer | query | False | Filter by statistics.throughput_raw.total<br>- Introduced in: 9.8 |
| wwpn | string | query | False | Filter by wwpn |
| wwnn | string | query | False | Filter by wwnn |
| state | string | query | False | Filter by state |
| svm.uuid | string | query | False | Filter by svm.uuid |
| svm.name | string | query | False | Filter by svm.name |
| fields | array[string] | query | False | Specify the fields to return. |
| max_records | integer | query | False | Limit the number of records returned. |

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| return_records | boolean | query | False | The default is true for GET calls. When set to false, only the number of records is returned.<br>- 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.<br>- Default value: 1<br>- Max value: 120<br>- Min value: 0 |
| 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 | Number of records. |
| records | array[fc_interface] | |

### Example response

```
{
  "_links": {
    "next": {
      "href": "/api/resourcelink"
    },
    "self": {
      "href": "/api/resourcelink"
    }
  },
  "records": [
    {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "comment": "string",
      "data_protocol": "string",
      "location": {
        "home_node": {
          "_links": {
            "self": {
              "href": "/api/resourcelink"
            }
          },
          "name": "node1",
          "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
        },
        "home_port": {
          "_links": {
            "self": {
              "href": "/api/resourcelink"
            }
          },
          "name": "0a",
          "node": {
            "name": "node1"
          },
          "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
        },
        "node": {
          "_links": {
            "self": {
              "href": "/api/resourcelink"
            }
          },
          "name": "node1",
          "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
        },
        "port": {
          "_links": {
            "self": {
              "href": "/api/resourcelink"
            }
          },
          "name": "0a",
          "node": {
            "name": "node1"
          },
          "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
        }
      },
      "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": "2017-01-25T11:20:13Z"
      },
      "name": "lif1",
      "port_address": "5060F",
      "state": "string",
      "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": "2017-01-25T11:20:13Z"
      },
      "svm": {
        "_links": {
          "self": {
            "href": "/api/resourcelink"
          }
        },
        "name": "svm1",
        "uuid": "02c9e252-41be-11e9-81d5-00a0986138f7"
      },
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412",
      "wwnn": "20:00:00:50:56:b4:13:01",
      "wwpn": "20:00:00:50:56:b4:13:a8"
    }
  ]
}
```

## Error

```
Status: Default, Error
```

| Name | Type | Description |
|---|---|---|
| error | 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 | |

home_node

| Name | Type | Description |
|---|---|---|
| _links | _links | |
| name | string | |
| uuid | string | |

node

The node on which the FC port is located.

| Name | Type | Description |
|---|---|---|
| name | string | The name of the node on which the FC port is located. |

fc_port_reference

An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network.

| Name | Type | Description |
|---|---|---|
| _links | _links | |
| name | string | The name of the FC port. |
| node | node | The node on which the FC port is located. |

| Name | Type | Description |
|---|---|---|
| uuid | string | The unique identifier of the FC port. |

node

| Name | Type | Description |
|---|---|---|
| _links | _links | |
| name | string | |
| uuid | string | |

location

The location of the FC interface is defined by the location of its port. An FC port is identified by its UUID, or a combination of its node name and port name. Either the UUID or the node name and port name are required for POST. To move an interface, supply either the UUID or the node name and port name in a PATCH.

The location of an FC interface can be set using "location.home_node" and "location.home_port" during a POST or PATCH. "location.node" and "location.port" refer to the current location of the FC interface. This may be different from "location.home_node" and "location.home_port" in instances where the FC interface failed over due to an offline node.

| Name | Type | Description |
|---|---|---|
| home_node | home_node | |
| home_port | fc_port_reference | An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network. |
| is_home | boolean | Indicates whether or not the FC interface currently resides on the home node. |
| node | node | |
| port | fc_port_reference | An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network. |

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 | Peformance 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 | Peformance 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 | Peformance 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. |
| 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. |

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. |
| read | integer | Performance metric for read I/O operations. |
| total | integer | Performance metric aggregated over all types of I/O operations. |
| write | integer | Peformance 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. |

| Name | Type | Description |
|---|---|---|
| write | integer | Peformance 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 | Peformance 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. |

| 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_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. |
| timestamp | string | The timestamp of the performance data. |

svm

| Name | Type | Description |
|---|---|---|
| _links | _links | |
| name | string | The name of the SVM. |
| uuid | string | The unique identifier of the SVM. |

fc_interface

A Fibre Channel (FC) interface is the logical endpoint for FC network connections to an SVM. An FC interface provides FC access to storage within the interface SVM using either Fibre Channel Protocol or NVMe over Fibre Channel (NVMe/FC).

An FC interface is created on an FC port which is located on a cluster node. The FC port must be specified to identify the location of the interface for a POST or PATCH operation that relocates an interface. You can identify the port by supplying either the node and port names or the port UUID.

| Name | Type | Description |
|---|---|---|
| _links | _links | |
| comment | string | A user configurable comment. Optional in POST; valid in PATCH. To clear a prior comment, set the property to an empty string in PATCH. |
| data_protocol | string | The data protocol for which the FC interface is configured. Required in POST. |
| enabled | boolean | The administrative state of the FC interface. The FC interface can be disabled to block all FC communication with the SVM through this interface. Optional in POST and PATCH; defaults to *true* (enabled) in POST. |
| location | location | The location of the FC interface is defined by the location of its port. An FC port is identified by its UUID, or a combination of its node name and port name. Either the UUID or the node name and port name are required for POST. To move an interface, supply either the UUID or the node name and port name in a PATCH.<br>The location of an FC interface can be set using "location.home_node" and "location.home_port" during a POST or PATCH. "location.node" and "location.port" refer to the current location of the FC interface. This may be different from "location.home_node" and "location.home_port" in instances where the FC interface failed over due to an offline node. |
| metric | metric | Performance numbers, such as IOPS latency and throughput |

| Name | Type | Description |
|---|---|---|
| name | string | The name of the FC interface. Required in POST; optional in PATCH. |
| port_address | string | The port address of the FC interface. Each FC port in an FC switched fabric has its own unique FC port address for routing purposes. The FC port address is assigned by a switch in the fabric when that port logs in to the fabric. This property refers to the address given by a switch to the FC interface when the SVM performs a port login (PLOGI).<br>This is useful for obtaining statistics and diagnostic information from FC switches.<br>This is a hexadecimal encoded numeric value. |
| state | string | The current operational state of the FC interface. The state is set to *down* if the interface is not enabled.<br>If the node hosting the port is down or unavailable, no state value is returned. |
| 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. |
| svm | svm | |
| uuid | string | The unique identifier of the FC interface. Required in the URL. |

| Name | Type | Description |
|---|---|---|
| wwnn | string | The world wide node name (WWNN) of the FC interface SVM. The WWNN is generated by ONTAP when Fibre Channel Protocol or the NVMe service is created for the FC interface SVM.<br>• example: 20:00:00:50:56:b4:13:01<br>• readOnly: 1<br>• Introduced in: 9.6 |
| wwpn | string | The world wide port name (WWPN) of the FC interface. The WWPN is generated by ONTAP when the FC interface is created.<br>• example: 20:00:00:50:56:b4:13:a8<br>• readOnly: 1<br>• Introduced in: 9.6 |

error_arguments

| Name | Type | Description |
|---|---|---|
| code | string | Argument code |
| message | string | Message argument |

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 FC interface

POST `/network/fc/interfaces`

**Introduced In:** 9.6

Creates an FC interface.

## Required properties

- `svm.uuid` or `svm.name` - Existing SVM in which to create the FC interface.
- `name` - Name of the FC interface.
- `location.port.uuid` or both `location.port.name` and `location.port.node.name` - FC port on which to create the FC interface.
- `data_protocol` - Data protocol for the FC interface.

## Default property values

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

- `enabled` - *true*

## Related ONTAP commands

- `network interface create`

## Learn more

- DOC /network/fc/interfaces

## Parameters

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| return_records | boolean | query | False | The default is false. If set to true, the records are returned.<br>• Default value: |

## Request Body

| Name | Type | Description |
|---|---|---|
| comment | string | A user configurable comment. Optional in POST; valid in PATCH. To clear a prior comment, set the property to an empty string in PATCH. |
| data_protocol | string | The data protocol for which the FC interface is configured. Required in POST. |
| enabled | boolean | The administrative state of the FC interface. The FC interface can be disabled to block all FC communication with the SVM through this interface. Optional in POST and PATCH; defaults to *true* (enabled) in POST. |
| location | location | The location of the FC interface is defined by the location of its port. An FC port is identified by its UUID, or a combination of its node name and port name. Either the UUID or the node name and port name are required for POST. To move an interface, supply either the UUID or the node name and port name in a PATCH.<br>The location of an FC interface can be set using "location.home_node" and "location.home_port" during a POST or PATCH. "location.node" and "location.port" refer to the current location of the FC interface. This may be different from "location.home_node" and "location.home_port" in instances where the FC interface failed over due to an offline node. |
| name | string | The name of the FC interface. Required in POST; optional in PATCH. |

| Name | Type | Description |
|---|---|---|
| port_address | string | The port address of the FC interface. Each FC port in an FC switched fabric has its own unique FC port address for routing purposes. The FC port address is assigned by a switch in the fabric when that port logs in to the fabric. This property refers to the address given by a switch to the FC interface when the SVM performs a port login (PLOGI).<br>This is useful for obtaining statistics and diagnostic information from FC switches.<br>This is a hexadecimal encoded numeric value. |
| state | string | The current operational state of the FC interface. The state is set to *down* if the interface is not enabled.<br>If the node hosting the port is down or unavailable, no state value is returned. |
| svm | svm | |
| uuid | string | The unique identifier of the FC interface. Required in the URL. |
| wwnn | string | The world wide node name (WWNN) of the FC interface SVM. The WWNN is generated by ONTAP when Fibre Channel Protocol or the NVMe service is created for the FC interface SVM.<br>• example: 20:00:00:50:56:b4:13:01<br>• readOnly: 1<br>• Introduced in: 9.6 |

| Name | Type | Description |
|---|---|---|
| wwpn | string | The world wide port name (WWPN) of the FC interface. The WWPN is generated by ONTAP when the FC interface is created.<br>• example: 20:00:00:50:56:b4:13:a8<br>• readOnly: 1<br>• Introduced in: 9.6 |

### Example request

```
{
  "comment": "string",
  "data_protocol": "string",
  "location": {
    "home_node": {
      "name": "node1",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "home_port": {
      "name": "0a",
      "node": {
        "name": "node1"
      },
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "node": {
      "name": "node1",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "port": {
      "name": "0a",
      "node": {
        "name": "node1"
      },
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    }
  },
  "name": "lif1",
  "port_address": "5060F",
  "state": "string",
  "svm": {
    "name": "svm1",
    "uuid": "02c9e252-41be-11e9-81d5-00a0986138f7"
  },
  "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412",
  "wwnn": "20:00:00:50:56:b4:13:01",
  "wwpn": "20:00:00:50:56:b4:13:a8"
}
```

## Response

```
Status: 201, Created
```

| Name | Type | Description |
|---|---|---|
| num_records | integer | Number of records. |
| records | array[fc_interface] | |

#### Example response

```
{
  "records": [
    {
      "comment": "string",
      "data_protocol": "string",
      "location": {
        "home_node": {
          "name": "node1",
          "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
        },
        "home_port": {
          "name": "0a",
          "node": {
            "name": "node1"
          },
          "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
        },
        "node": {
          "name": "node1",
          "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
        },
        "port": {
          "name": "0a",
          "node": {
            "name": "node1"
          },
          "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
        }
      },
      "name": "lif1",
      "port_address": "5060F",
      "state": "string",
      "svm": {
        "name": "svm1",
        "uuid": "02c9e252-41be-11e9-81d5-00a0986138f7"
      },
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412",
      "wwnn": "20:00:00:50:56:b4:13:01",
      "wwpn": "20:00:00:50:56:b4:13:a8"
    }
  ]
}
```

## Error

```
Status: Default
```

ONTAP Error Response Codes

| Error Code | Description |
|---|---|
| 1966140 | An interface with the same name already exists. |
| 1966217 | The specified port is not valid on the node provided. |
| 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. |
| 5373966 | A Fibre Channel interface with the *fcp* protocol cannot be created in an SVM that is configured for NVMe. |
| 5374102 | The specified Fibre Channel interface cannot be created because the Fibre Channel adapter is down. Bring the adapter up and try again. |
| 5374871 | The Fibre Channel port identified by the specified UUID does not refer to the same port as that identified by the specified node name and/or port name. |
| 5374872 | If either `location.port.node.name` or `location.port.name` is supplied, both properties must be supplied. |
| 5374873 | The Fibre Channel port must be specified using either `location.port.uuid` or `location.port.node.name` and `location.port.name`. |
| 72089652 | An NVMe service must be created before creating a Fibre Channel interface using the NVMe over FC data protocol. |
| 72089672 | The specified Fibre Channel port does not support the NVMe over FC data protocol. |
| 72089900 | A Fibre Channel interface with the *fc_nvme* protocol cannot be created in an SVM that is configured for a SAN protocol. |

## Definitions

**See Definitions**

href

| Name | Type | Description |
|---|---|---|
| href | string | |

_links

home_node

| Name | Type | Description |
|---|---|---|
| name | string | |
| uuid | string | |

node

The node on which the FC port is located.

| Name | Type | Description |
|---|---|---|
| name | string | The name of the node on which the FC port is located. |

fc_port_reference

An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network.

| Name | Type | Description |
|---|---|---|
| name | string | The name of the FC port. |
| node | node | The node on which the FC port is located. |
| uuid | string | The unique identifier of the FC port. |

node

| Name | Type | Description |
|---|---|---|
| name | string | |
| uuid | string | |

location

The location of the FC interface is defined by the location of its port. An FC port is identified by its UUID, or a combination of its node name and port name. Either the UUID or the node name and port name are required for POST. To move an interface, supply either the UUID or the node name and port name in a PATCH.

The location of an FC interface can be set using "location.home_node" and "location.home_port" during a POST or PATCH. "location.node" and "location.port" refer to the current location of the FC interface. This may be different from "location.home_node" and "location.home_port" in instances where the FC interface failed over due to an offline node.

| Name | Type | Description |
|---|---|---|
| home_node | home_node | |
| home_port | fc_port_reference | An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network. |
| is_home | boolean | Indicates whether or not the FC interface currently resides on the home node. |
| node | node | |
| port | fc_port_reference | An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network. |

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 | Peformance 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 | Peformance 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 | Peformance 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. |

| 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. |

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 | Peformance 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 | Peformance 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 | Peformance 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. |

svm

| Name | Type | Description |
|---|---|---|
| name | string | The name of the SVM. |
| uuid | string | The unique identifier of the SVM. |

fc_interface

A Fibre Channel (FC) interface is the logical endpoint for FC network connections to an SVM. An FC interface provides FC access to storage within the interface SVM using either Fibre Channel Protocol or NVMe over Fibre Channel (NVMe/FC).

An FC interface is created on an FC port which is located on a cluster node. The FC port must be specified to identify the location of the interface for a POST or PATCH operation that relocates an interface. You can identify the port by supplying either the node and port names or the port UUID.

| Name | Type | Description |
|---|---|---|
| comment | string | A user configurable comment. Optional in POST; valid in PATCH. To clear a prior comment, set the property to an empty string in PATCH. |
| data_protocol | string | The data protocol for which the FC interface is configured. Required in POST. |
| enabled | boolean | The administrative state of the FC interface. The FC interface can be disabled to block all FC communication with the SVM through this interface. Optional in POST and PATCH; defaults to *true* (enabled) in POST. |

| Name | Type | Description |
|---|---|---|
| location | location | The location of the FC interface is defined by the location of its port. An FC port is identified by its UUID, or a combination of its node name and port name. Either the UUID or the node name and port name are required for POST. To move an interface, supply either the UUID or the node name and port name in a PATCH.<br>The location of an FC interface can be set using "location.home_node" and "location.home_port" during a POST or PATCH. "location.node" and "location.port" refer to the current location of the FC interface. This may be different from "location.home_node" and "location.home_port" in instances where the FC interface failed over due to an offline node. |
| name | string | The name of the FC interface. Required in POST; optional in PATCH. |
| port_address | string | The port address of the FC interface. Each FC port in an FC switched fabric has its own unique FC port address for routing purposes. The FC port address is assigned by a switch in the fabric when that port logs in to the fabric. This property refers to the address given by a switch to the FC interface when the SVM performs a port login (PLOGI).<br>This is useful for obtaining statistics and diagnostic information from FC switches.<br>This is a hexadecimal encoded numeric value. |

| Name | Type | Description |
|---|---|---|
| state | string | The current operational state of the FC interface. The state is set to *down* if the interface is not enabled.<br>If the node hosting the port is down or unavailable, no state value is returned. |
| svm | svm | |
| uuid | string | The unique identifier of the FC interface. Required in the URL. |
| wwnn | string | The world wide node name (WWNN) of the FC interface SVM. The WWNN is generated by ONTAP when Fibre Channel Protocol or the NVMe service is created for the FC interface SVM.<br>• example: 20:00:00:50:56:b4:13:01<br>• readOnly: 1<br>• Introduced in: 9.6 |
| wwpn | string | The world wide port name (WWPN) of the FC interface. The WWPN is generated by ONTAP when the FC interface is created.<br>• example: 20:00:00:50:56:b4:13:a8<br>• readOnly: 1<br>• Introduced in: 9.6 |

error_arguments

| Name | Type | Description |
|---|---|---|
| code | string | Argument code |
| message | string | Message argument |

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 FC interface

DELETE `/network/fc/interfaces/{uuid}`

**Introduced In:** 9.6

Deletes an FC interface.

## Related ONTAP commands

- `network interface delete`

## Learn more

- DOC /network/fc/interfaces

## Parameters

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| uuid | string | path | True | The unique identifier for the FC interface. |

## Response

```
Status: 200, Ok
```

## Error

```
Status: Default
```

ONTAP Error Response Codes

| Error Code | Description |
|---|---|
| 53280992 | The FC interface could not be deleted because it is enabled. |

| Name | Type | Description |
|---|---|---|
| error | 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 |

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 FC interface

GET `/network/fc/interfaces/{uuid}`

**Introduced In:** 9.6

Retrieves an FC interface.

## Expensive properties

There is an added 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.

- `statistics.*`
- `metric.*`

## Related ONTAP commands

- `network interface show`
- `vserver fcp interface show`

## Learn more

- DOC /network/fc/interfaces

## Parameters

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| uuid | string | path | True | The unique identifier for the FC interface. |
| fields | array[string] | query | False | Specify the fields to return. |

## Response

```
Status: 200, Ok
```

| Name | Type | Description |
|---|---|---|
| _links | _links | |
| comment | string | A user configurable comment. Optional in POST; valid in PATCH. To clear a prior comment, set the property to an empty string in PATCH. |
| data_protocol | string | The data protocol for which the FC interface is configured. Required in POST. |
| enabled | boolean | The administrative state of the FC interface. The FC interface can be disabled to block all FC communication with the SVM through this interface. Optional in POST and PATCH; defaults to *true* (enabled) in POST. |

| Name | Type | Description |
|---|---|---|
| location | location | The location of the FC interface is defined by the location of its port. An FC port is identified by its UUID, or a combination of its node name and port name. Either the UUID or the node name and port name are required for POST. To move an interface, supply either the UUID or the node name and port name in a PATCH.<br>The location of an FC interface can be set using "location.home_node" and "location.home_port" during a POST or PATCH. "location.node" and "location.port" refer to the current location of the FC interface. This may be different from "location.home_node" and "location.home_port" in instances where the FC interface failed over due to an offline node. |
| metric | metric | Performance numbers, such as IOPS latency and throughput |
| name | string | The name of the FC interface. Required in POST; optional in PATCH. |
| port_address | string | The port address of the FC interface. Each FC port in an FC switched fabric has its own unique FC port address for routing purposes. The FC port address is assigned by a switch in the fabric when that port logs in to the fabric. This property refers to the address given by a switch to the FC interface when the SVM performs a port login (PLOGI).<br>This is useful for obtaining statistics and diagnostic information from FC switches.<br>This is a hexadecimal encoded numeric value. |

| Name | Type | Description |
|---|---|---|
| state | string | The current operational state of the FC interface. The state is set to *down* if the interface is not enabled.<br>If the node hosting the port is down or unavailable, no state value is returned. |
| 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. |
| svm | svm | |
| uuid | string | The unique identifier of the FC interface. Required in the URL. |
| wwnn | string | The world wide node name (WWNN) of the FC interface SVM. The WWNN is generated by ONTAP when Fibre Channel Protocol or the NVMe service is created for the FC interface SVM.<br>• example: 20:00:00:50:56:b4:13:01<br>• readOnly: 1<br>• Introduced in: 9.6 |
| wwpn | string | The world wide port name (WWPN) of the FC interface. The WWPN is generated by ONTAP when the FC interface is created.<br>• example: 20:00:00:50:56:b4:13:a8<br>• readOnly: 1<br>• Introduced in: 9.6 |

### Example response

```
{
  "_links": {
    "self": {
      "href": "/api/resourcelink"
    }
  },
  "comment": "string",
  "data_protocol": "string",
  "location": {
    "home_node": {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "name": "node1",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "home_port": {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "name": "0a",
      "node": {
        "name": "node1"
      },
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "node": {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "name": "node1",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "port": {
      "_links": {
        "self": {
          "href": "/api/resourcelink"
        }
      },
      "name": "0a",
      "node": {
        "name": "node1"
      },
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    }
  },
  "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": "2017-01-25T11:20:13Z"
  },
  "name": "lif1",
  "port_address": "5060F",
  "state": "string",
  "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": "2017-01-25T11:20:13Z"
  },
  "svm": {
    "_links": {
      "self": {
        "href": "/api/resourcelink"
      }
    },
    "name": "svm1",
    "uuid": "02c9e252-41be-11e9-81d5-00a0986138f7"
  },
  "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412",
  "wwnn": "20:00:00:50:56:b4:13:01",
  "wwpn": "20:00:00:50:56:b4:13:a8"
}
```

## Error

```
Status: Default, Error
```

| Name | Type | Description |
|---|---|---|
| error | 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 | |

home_node

| Name | Type | Description |
|---|---|---|
| _links | _links | |
| name | string | |
| uuid | string | |

node

The node on which the FC port is located.

| Name | Type | Description |
|---|---|---|
| name | string | The name of the node on which the FC port is located. |

fc_port_reference

An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network.

| Name | Type | Description |
|---|---|---|
| _links | _links | |
| name | string | The name of the FC port. |
| node | node | The node on which the FC port is located. |
| uuid | string | The unique identifier of the FC port. |

node

| Name | Type | Description |
|---|---|---|
| _links | _links | |
| name | string | |
| uuid | string | |

location

The location of the FC interface is defined by the location of its port. An FC port is identified by its UUID, or a combination of its node name and port name. Either the UUID or the node name and port name are required for POST. To move an interface, supply either the UUID or the node name and port name in a PATCH.

The location of an FC interface can be set using "location.home_node" and "location.home_port" during a POST or PATCH. "location.node" and "location.port" refer to the current location of the FC interface. This may be different from "location.home_node" and "location.home_port" in instances where the FC interface failed over due to an offline node.

| Name | Type | Description |
|---|---|---|
| home_node | home_node | |
| home_port | fc_port_reference | An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network. |
| is_home | boolean | Indicates whether or not the FC interface currently resides on the home node. |
| node | node | |
| port | fc_port_reference | An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network. |

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. |

| 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 | Peformance 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 | Peformance 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 | Peformance 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. |
| 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. |

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. |
| read | integer | Performance metric for read I/O operations. |
| total | integer | Performance metric aggregated over all types of I/O operations. |
| write | integer | Peformance 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 | Peformance 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 | Peformance 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. |

| 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_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. |
| timestamp | string | The timestamp of the performance data. |

svm

| Name | Type | Description |
|---|---|---|
| _links | _links | |
| name | string | The name of the SVM. |
| uuid | string | The unique identifier of the SVM. |

error_arguments

| Name | Type | Description |
|---|---|---|
| code | string | Argument code |
| message | string | Message argument |

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 FC interface

PATCH `/network/fc/interfaces/{uuid}`

**Introduced In:** 9.6

Updates an FC interface.

## Related ONTAP commands

- `network interface modify`

## Learn more

- DOC /network/fc/interfaces

## Parameters

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| uuid | string | path | True | The unique identifier for the FC interface. |

## Request Body

| Name | Type | Description |
|---|---|---|
| comment | string | A user configurable comment. Optional in POST; valid in PATCH. To clear a prior comment, set the property to an empty string in PATCH. |
| enabled | boolean | The administrative state of the FC interface. The FC interface can be disabled to block all FC communication with the SVM through this interface. Optional in POST and PATCH; defaults to *true* (enabled) in POST. |
| location | location | The location of the FC interface is defined by the location of its port. An FC port is identified by its UUID, or a combination of its node name and port name. Either the UUID or the node name and port name are required for POST. To move an interface, supply either the UUID or the node name and port name in a PATCH.<br>The location of an FC interface can be set using "location.home_node" and "location.home_port" during a POST or PATCH. "location.node" and "location.port" refer to the current location of the FC interface. This may be different from "location.home_node" and "location.home_port" in instances where the FC interface failed over due to an offline node. |
| name | string | The name of the FC interface. Required in POST; optional in PATCH. |

| Name | Type | Description |
|---|---|---|
| port_address | string | The port address of the FC interface. Each FC port in an FC switched fabric has its own unique FC port address for routing purposes. The FC port address is assigned by a switch in the fabric when that port logs in to the fabric. This property refers to the address given by a switch to the FC interface when the SVM performs a port login (PLOGI).<br>This is useful for obtaining statistics and diagnostic information from FC switches.<br>This is a hexadecimal encoded numeric value. |
| state | string | The current operational state of the FC interface. The state is set to *down* if the interface is not enabled.<br>If the node hosting the port is down or unavailable, no state value is returned. |
| svm | svm | |
| uuid | string | The unique identifier of the FC interface. Required in the URL. |
| wwnn | string | The world wide node name (WWNN) of the FC interface SVM. The WWNN is generated by ONTAP when Fibre Channel Protocol or the NVMe service is created for the FC interface SVM.<br>• example: 20:00:00:50:56:b4:13:01<br>• readOnly: 1<br>• Introduced in: 9.6 |

| Name | Type | Description |
|---|---|---|
| wwpn | string | The world wide port name (WWPN) of the FC interface. The WWPN is generated by ONTAP when the FC interface is created.<br>• example: 20:00:00:50:56:b4:13:a8<br>• readOnly: 1<br>• Introduced in: 9.6 |

### Example request

```
{
  "comment": "string",
  "location": {
    "home_node": {
      "name": "node1",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "home_port": {
      "name": "0a",
      "node": {
        "name": "node1"
      },
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "node": {
      "name": "node1",
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    },
    "port": {
      "name": "0a",
      "node": {
        "name": "node1"
      },
      "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412"
    }
  },
  "name": "lif1",
  "port_address": "5060F",
  "state": "string",
  "svm": {
    "name": "svm1",
    "uuid": "02c9e252-41be-11e9-81d5-00a0986138f7"
  },
  "uuid": "1cd8a442-86d1-11e0-ae1c-123478563412",
  "wwnn": "20:00:00:50:56:b4:13:01",
  "wwpn": "20:00:00:50:56:b4:13:a8"
}
```

## Response

```
Status: 200, Ok
```

## Error

```
Status: Default
```

ONTAP Error Response Codes

| Error Code | Description |
|---|---|
| 1966140 | An interface with the same name already exists. |
| 1966217 | The specified port is not valid on the node provided. |
| 1966238 | The node or port of an active SAN data interface cannot be changed. |
| 1966702 | The destination node is not healthy. |
| 5374579 | The SAN Kernel Agent on the node is unavailable. |
| 5374870 | A partial failure occurred; renaming the interface failed. Correct the error and resubmit the request. |
| 5374871 | The Fibre Channel port identified by the specified UUID does not refer to the same port as that identified by the specified node name and/or port name. |
| 5374872 | If either `location.port.node.name` or `location.port.name` is supplied, both properties must be supplied. |
| 72089674 | You cannot move a Fibre Channel interface configured for the NVMe over FC data protocol. |

## Definitions

**See Definitions**

href

| Name | Type | Description |
|---|---|---|
| href | string | |

_links

home_node

| Name | Type | Description |
|---|---|---|
| name | string | |
| uuid | string | |

node

The node on which the FC port is located.

| Name | Type | Description |
|---|---|---|
| name | string | The name of the node on which the FC port is located. |

fc_port_reference

An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network.

| Name | Type | Description |
|---|---|---|
| name | string | The name of the FC port. |
| node | node | The node on which the FC port is located. |
| uuid | string | The unique identifier of the FC port. |

node

| Name | Type | Description |
|---|---|---|
| name | string | |
| uuid | string | |

location

The location of the FC interface is defined by the location of its port. An FC port is identified by its UUID, or a combination of its node name and port name. Either the UUID or the node name and port name are required for POST. To move an interface, supply either the UUID or the node name and port name in a PATCH.

The location of an FC interface can be set using "location.home_node" and "location.home_port" during a POST or PATCH. "location.node" and "location.port" refer to the current location of the FC interface. This may be different from "location.home_node" and "location.home_port" in instances where the FC interface failed over due to an offline node.

| Name | Type | Description |
|---|---|---|
| home_node | home_node | |
| home_port | fc_port_reference | An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network. |
| is_home | boolean | Indicates whether or not the FC interface currently resides on the home node. |
| node | node | |
| port | fc_port_reference | An FC port is the physical port of an FC adapter on a cluster node that can be connected to an FC network. |

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 | Peformance 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 | Peformance 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 | Peformance 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. |

| 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. |

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 | Peformance 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 | Peformance 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 | Peformance 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. |

svm

| Name | Type | Description |
|---|---|---|
| name | string | The name of the SVM. |
| uuid | string | The unique identifier of the SVM. |

fc_interface

A Fibre Channel (FC) interface is the logical endpoint for FC network connections to an SVM. An FC interface provides FC access to storage within the interface SVM using either Fibre Channel Protocol or NVMe over Fibre Channel (NVMe/FC).

An FC interface is created on an FC port which is located on a cluster node. The FC port must be specified to identify the location of the interface for a POST or PATCH operation that relocates an interface. You can identify the port by supplying either the node and port names or the port UUID.

| Name | Type | Description |
|---|---|---|
| comment | string | A user configurable comment. Optional in POST; valid in PATCH. To clear a prior comment, set the property to an empty string in PATCH. |
| enabled | boolean | The administrative state of the FC interface. The FC interface can be disabled to block all FC communication with the SVM through this interface. Optional in POST and PATCH; defaults to *true* (enabled) in POST. |

| Name | Type | Description |
|---|---|---|
| location | location | The location of the FC interface is defined by the location of its port. An FC port is identified by its UUID, or a combination of its node name and port name. Either the UUID or the node name and port name are required for POST. To move an interface, supply either the UUID or the node name and port name in a PATCH.<br>The location of an FC interface can be set using "location.home_node" and "location.home_port" during a POST or PATCH. "location.node" and "location.port" refer to the current location of the FC interface. This may be different from "location.home_node" and "location.home_port" in instances where the FC interface failed over due to an offline node. |
| name | string | The name of the FC interface. Required in POST; optional in PATCH. |
| port_address | string | The port address of the FC interface. Each FC port in an FC switched fabric has its own unique FC port address for routing purposes. The FC port address is assigned by a switch in the fabric when that port logs in to the fabric. This property refers to the address given by a switch to the FC interface when the SVM performs a port login (PLOGI).<br>This is useful for obtaining statistics and diagnostic information from FC switches.<br>This is a hexadecimal encoded numeric value. |

| Name | Type | Description |
|---|---|---|
| state | string | The current operational state of the FC interface. The state is set to *down* if the interface is not enabled.<br>If the node hosting the port is down or unavailable, no state value is returned. |
| svm | svm | |
| uuid | string | The unique identifier of the FC interface. Required in the URL. |
| wwnn | string | The world wide node name (WWNN) of the FC interface SVM. The WWNN is generated by ONTAP when Fibre Channel Protocol or the NVMe service is created for the FC interface SVM.<br>• example: 20:00:00:50:56:b4:13:01<br>• readOnly: 1<br>• Introduced in: 9.6 |
| wwpn | string | The world wide port name (WWPN) of the FC interface. The WWPN is generated by ONTAP when the FC interface is created.<br>• example: 20:00:00:50:56:b4:13:a8<br>• readOnly: 1<br>• Introduced in: 9.6 |

error_arguments

| Name | Type | Description |
|---|---|---|
| code | string | Argument code |
| message | string | Message argument |

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 FC interface historical performance metrics

GET `/network/fc/interfaces/{uuid}/metrics`

**Introduced In:** 9.8

Retrieves historical performance metrics for an FC interface.

## Parameters

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| throughput.write | integer | query | False | Filter by throughput.write |
| throughput.read | integer | query | False | Filter by throughput.read |
| throughput.total | integer | query | False | Filter by throughput.total |
| duration | string | query | False | Filter by duration |
| timestamp | string | query | False | Filter by timestamp |
| status | string | query | False | Filter by status |
| iops.total | integer | query | False | Filter by iops.total |
| iops.read | integer | query | False | Filter by iops.read |
| iops.other | integer | query | False | Filter by iops.other |
| iops.write | integer | query | False | Filter by iops.write |

| Name | Type | In | Required | Description |
|---|---|---|---|---|
| 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 |
| latency.write | integer | query | False | Filter by latency.write |
| uuid | string | path | True | Unique identifier of the FC interface. |
| 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:<br>• 1h: Metrics over the most recent hour sampled over 15 seconds.<br>• 1d: Metrics over the most recent day sampled over 5 minutes.<br>• 1w: Metrics over the most recent week sampled over 30 minutes.<br>• 1m: Metrics over the most recent month sampled over 2 hours.<br>• 1y: Metrics over the most recent year sampled over a day.<br>• Default value: 1<br>• 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.<br>• Default value: 1<br>• Max value: 120<br>• 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"
    }
  },
  "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": "2017-01-25T11:20:13Z"
    }
  ]
}
```

## Error

```
Status: Default, Error
```

| Name | Type | Description |
|---|---|---|
| error | 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 | Peformance 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 | Peformance 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 | Peformance metric for write I/O operations. |

records

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. |

| 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. |

error_arguments

| Name | Type | Description |
|---|---|---|
| code | string | Argument code |
| message | string | Message argument |

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