



Cisco Crosswork Hierarchical Controller 7.0

NBI and SHQL Reference Guide

April 2023

Introducing Crosswork Hierarchical Controller APIs

Cisco Crosswork Hierarchical Controller offers an extensive set of APIs as a northbound interface to service orchestrators, OSSs, and Inventory systems. These APIs are essential tools to automate operational workflows, planning scenarios, and administrative tasks. These APIs can be used to retrieve information, provision services and resources, build the network and register for notifications.

Crosswork Hierarchical Controller makes use of multiple technologies for these APIs:

- RESTCONF – based on YANG models defined for NETCONF
- REST with JSON in body
- REST with SHQL in body

API Architecture and Technologies

RESTCONF/YANG Protocol Introduction

The RESTCONF protocol is an HTTP-based protocol that provides a programmatic interface for accessing data defined in YANG 1.0 [RFC 6020] or YANG 1.1 [RFC 7950] using the data store concepts defined in the Network Configuration Protocol (NETCONF) [RFC 6241] and further extended in NMDA [RFC 8342].

The main reasons for selecting RESTCONF include:

- The capability of mapping HTTP CRUD operations to provide a model-driven API.
- The ability to use a ubiquitous REST architecture used by many Web applications.
- The ability to use and mix various standard data models defined in the YANG modeling language produced by multiple SDOs. This potentially allows an implementor a choice of SDN architectures, namely IETF ACTN, ONF TAPI, and MEF TAPI.

Crosswork Hierarchical Controller’s RESTCONF implementation consists of the following resources:

- **+restconf}/data (Data API):** Create/Retrieve/Update/Delete (CRUD) based API for the entire configuration data tree defined in the YANG data models.
- **+restconf}/operations (Operations API):** RPC-based API entry to trigger RPC defined in YANG.
- **+restconf}/data/ietf-restconf-monitoring:restconf-state/streams (Notifications API):** RESTCONF API to discover possible Notification streams as defined in RESTCONF protocol. See [Device Management](#) for more information.
- **+restconf}/yang-library-version:** This mandatory leaf identifies the revision date of the "ietf-yang-library" YANG module that is implemented by this server.

As per RFC 8040, the RESTCONF implementation supports JSON encoding for YANG data [RFC 7951]. The server accepts “application/yang-data+json”.

Query Filtering

According to the RESTCONF specification, each operation allows zero or more query parameters to be present in the request URI. Specifically, query operations’ parameters are described in [Section 4.8 of \[RFC 8040\]](#). The following table describes the supported query parameters.

Table 1. Supported Query Parameters

| Name | Methods | Description |
|---------|---------|--|
| content | GET | Select config and/or non-config data resources |
| depth | GET | Request limited subtree depth in the reply content |
| fields | GET | Request a subset of the target resource content |

Transport Security

Data Integrity

Crosswork Hierarchical Controller serves all NBI-accessed data over TLS [RFC 5246] – a transport-layer providing data integrity and confidentiality. Any client accessing Crosswork Hierarchical Controller **MUST** support TLS 1.2 and above.

Server Authentication

Crosswork Hierarchical Controller NBI presents an X.509v3 certificate upon establishing a TLS connection with a client.

A client **MUST** support certificate validation as per RESTCONF [RFC 8040 Section 2]. The RESTCONF client **MUST** either (1) use X.509 certificate path validation [RFC5280] to verify the integrity of the RESTCONF server's TLS certificate or (2) match the server's TLS certificate with a certificate obtained by a trusted mechanism (e.g., a pinned certificate). If the X.509 certificate path validation fails and the presented X.509 certificate does not match a certificate obtained by a trusted mechanism, the connection **MUST** be terminated, as described in Section 7.2.1 of [RFC5246].

NOTE: For the purpose of testing, a client may disable certificate validation to check interface functionality.

Client Authentication

Crosswork Hierarchical Controller client authentication is based on HTTP Basic Authentication [RFC 7617]. A client must present a username and password over the established TLS connection. More specifically, a client must use an Authorization header using a Basic authentication scheme. For further information, see the HTTP Authentication Scheme Registry [RFC 7235].

API Endpoints

Crosswork Hierarchical Controller supports APIs for receiving notifications, information, and provisioning services.

- **Device manager:** Various APIs that allow you to manage adapters, devices and credentials
- **Get inventory:** Get optical devices and routers, their cards, ports, and their attributes.
- **Get performance counters:** Get L2 performance (throughput) counters per specific port or link.
- **Cross links:** Various APIs to administer cross links.
- **Shared risk:** Various APIs that allow you to administer shared risk policies and rules.
- **Generic queries:** Use SHQL commands to query any data from the Crosswork Hierarchical Controller model.

Object IDs

Any reference to specific objects in the APIs uses object IDs. IDs can be used for a network, site, node, TP (termination point), LSP, service, link, network access, or VLAN.

The implementation by Crosswork Hierarchical Controller for these IDs is the GUID. The GUID is a unique identifier of any object in the model and it can be retrieved as the object ID using the relevant GET API. Once retrieved, it can be used as the object reference further on.

Device Management

Crosswork Hierarchical Controller provides APIs to administer device management.

You can access the Device Manager API using Swagger:

- <https://<host>/api/v2/apps/device-manager-srv/rest/doc>

The APIs include:

- Get all adapters
- Get adapter configuration schema
- Get adapter device schema
- Get all devices
- Add device to Crosswork Hierarchical Controller
- Assign device to adapter
- Update device's adapter configuration
- Unassign device from adapter
- Update device name
- Assign device to Site
- Get device status
- Get a device
- Remove device from Crosswork Hierarchical Controller
- Get credentials keys' names and types
- Add new credentials
- Update credentials
- Delete credentials
- Get the parameters schema for the specified credentials type

Get Adapters

Use this API to get the list of all the adapters. The **guid** in the response is the **adapterGuid** used in several of the other device manager methods.

Request Method

GET

Request URL

<https://example-host/api/v2/apps/device-manager-srv/rest/adapter>

Request Parameters

None

Response Example

```
[
  {
    "guid": "cisco_ios_xr",
    "enabled": true,
    "config": {
      "polling": 300,
      "concurrency": 1,
      "ssh_config": {
        "enabled": false,
        "connect_timeout": 10,
        "command_timeout": 10
      },
      "file_bringing": {
        "enabled": false,
        "location": "",
        "file_type": "XX"
      },
      "collection_parameters": {
        "enable_igp_isis": true,
        "enable_stats": true,
        "enable_vrf": true,
        "enable_lldp": true,
        "enable_mpls": true,
        "enable_snmp": true,
        "igp_isis_priority": 1,
        "igp_seed_routers": true
      }
    }
  },
  {
    "guid": "juniper_os_1",
    "enabled": true,
    "config": {
      "polling": 300,
      "concurrency": 1,
      "ssh_config": {
        "enabled": false,
```

```

        "connect_timeout": 10,
        "command_timeout": 10
    },
    "file_bringing": {
        "enabled": false,
        "location": "",
        "file_type": "XX"
    },
    "collection_parameters": {
        "enable_igp_isis": true,
        "enable_igp_ospf": true,
        "enable_stats": true,
        "enable_vrf": true,
        "enable_lldp": true,
        "enable_mpls": true,
        "enable_snmp": true,
        "igp_isis_priority": 1,
        "igp_seed_routers": true
    }
}
]

```

Get Adapter Configuration Schema

Use this API to retrieve the adapter's configuration schema.

Request Method

GET

Request URL

`https://example-host/api/v2/apps/device-manager-srv/rest/adapter/{adapterGuid}/adapter-schema`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|---|
| adapterGuid | string | The adapter guid. Use the guid returned by the Get Adapters method. |

Response Example

```

{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "polling": {

```

```
"type": "integer",
"minimum": 0,
"default": 300,
"title": "Polling Cycle [sec]",
"description": "Poll every X seconds"
},
"concurrency": {
  "type": "integer",
  "default": 1,
  "title": "Number of concurrent routers collected"
},
"ssh_config": {
  "type": "object",
  "title": "SSH Configuration Parameters",
  "properties": {
    "enabled": {
      "type": "boolean",
      "default": false,
      "title": "Enable Tunnel"
    },
    "host": {
      "type": "string",
      "title": "Tunnel Host"
    },
    "port": {
      "type": "integer",
      "title": "Tunnel Port"
    },
    "tunnel_credentials_key": {
      "type": "string",
      "title": "Tunnel Credentials Key",
      "credentials-types": [
        "SSH_USER_PASSWORD"
      ]
    },
    "connect_timeout": {
      "type": "integer",
      "default": 10,
      "title": "Router Connect timeout"
    },
    "command_timeout": {
      "type": "integer",
```

```

        "default": 10,
        "title": "Router Command timeout"
    }
},
"additionalProperties": false
},
"file_bringer": {
    "type": "object",
    "title": "File Bringer Parameters",
    "properties": {
        "enabled": {
            "type": "boolean",
            "default": false,
            "title": "Enable File Bringer"
        },
        "location": {
            "type": "string",
            "title": "File Server Location",
            "description": "\"[sftp|file]://<server>:<port>/<absolute path>\"",
            "default": ""
        },
        "file_type": {
            "title": "File Type",
            "enum": [
                "XX",
                "XX"
            ],
            "default": "XX"
        },
        "authentication": {
            "type": "string",
            "credentials-types": [
                "SFTP"
            ]
        }
    },
    "additionalProperties": false
},
"collection_parameters": {
    "type": "object",
    "title": "Collection Parameters",
    "properties": {

```



```
"enable_igp_isis": {
  "type": "boolean",
  "default": false,
  "title": "Enable IGP IS-IS Collection"
},
"enable_stats": {
  "type": "boolean",
  "default": false,
  "title": "Enable Stats Collection"
},
"enable_vrf": {
  "type": "boolean",
  "default": false,
  "title": "Enable VRF Collection"
},
"enable_lldp": {
  "type": "boolean",
  "default": false,
  "title": "Enable LLDP Collection"
},
"enable_mpls": {
  "type": "boolean",
  "default": false,
  "title": "Enable MPLS Tunnels Collection"
},
"enable_snmp": {
  "type": "boolean",
  "default": false,
  "title": "Enable SNMP Collection"
},
"igp_isis_priority": {
  "type": "integer",
  "default": 1,
  "title": "IGP IS-IS Priority"
},
"igp_seed_routers": {
  "type": "boolean",
  "default": true,
  "title": "Collect only IGP IS-IS seed routers"
}
},
"additionalProperties": false
```

```

    }
  },
  "additionalProperties": false,
  "required": [
    "polling"
  ]
}

```

Get Adapter Device Schema

Use this API to retrieve the device configuration schema of a specific adapter.

Request Method

GET

Request URL

`https://example-host/api/v2/apps/device-manager-srv/rest/adapter/{adpaterGuid}/device-schema`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|---|
| adapterGuid | string | The adapter guid. Use the guid returned by the Get Adapters method. |

Response Example

```

{
  "type": "object",
  "properties": {
    "host": {
      "type": "string"
    },
    "port": {
      "type": "integer",
      "default": 22
    },
    "direct_connect": {
      "type": "boolean",
      "default": false,
      "title": "Direct Connect (avoid tunnel if configured)"
    },
    "authentication": {
      "type": "string",
      "credentials-types": [
        "SSH_USER_PASSWORD"
      ]
    }
  }
}

```

```

    },
    "enabled": {
      "type": "boolean",
      "default": false
    }
  },
  "additionalProperties": false,
  "required": [
    "enabled",
    "host",
    "port"
  ]
}

```

Get All Devices

Use this API to get all devices for a specific adapter. The **device_manager_guid** returned in the response is used as the **deviceGuid** in the relevant device manager APIs.

Request Method

GET

Request URL

`https://example-host/api/v2/apps/device-manager-srv/rest/device?adapter={adpaterGuid}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|---|
| adapterGuid | string | The adapter guid. Use the guid returned by the Get Adapters method. |

Response Example

```

[
  {
    "device_manager_guid": "0a9af2d2-324a-45dd-895e-9e6edb2f9d39",
    "inventory_type": "ROUTER",
    "device_model_guid": "IN/cisco_ios_xr/ROUTER-xrv-p1",
    "name": "cisco_1",
    "adapters": {
      "cisco_ios_xr": {
        "adapter_guid": "cisco_ios_xr",
        "config": {
          "port": 22,
          "direct_connect": true,
          "enabled": true,
          "host": "10.1.0.58",

```

```
        "authentication": "Cisco"
      },
      "enabled": false
    }
  },
  "identifiers": {
    "host_name": "10.1.0.58",
    "tid": null,
    "serial": null,
    "loopback_mac": null,
    "loopback_ip": null,
    "extra": {}
  },
  "site": null,
  "pending_delete": false,
  "enabled": false
},
{
  "device_manager_guid": "9988749383",
  "inventory_type": "ROUTER",
  "device_model_guid": "IN/cisco_ios_xr/ROUTER-xrv-p2",
  "name": "cisco_2",
  "adapters": {
    "cisco_ios_xr": {
      "adapter_guid": "cisco_ios_xr",
      "config": {
        "port": 22,
        "direct_connect": true,
        "enabled": true,
        "host": "10.1.0.71",
        "authentication": "Cisco"
      },
      "enabled": false
    }
  },
  "identifiers": {
    "host_name": "10.1.0.71",
    "tid": null,
    "serial": null,
    "loopback_mac": null,
    "loopback_ip": null,
    "extra": {}
  }
}
```

```
    },
    "site": null,
    "pending_delete": false,
    "enabled": false
  }
]
```

Add Device to Crosswork Hierarchical Controller

Use this API to add a device to Crosswork Hierarchical Controller.

Request Method

POST

Request URL

`https://example-host/api/v2/apps/device-manager-srv/rest/device`

Request Parameters

None

Request Body

| Parameter Name | Data Type | Description |
|----------------|-----------|---------------------------------------|
| description | string | The device name. |
| site | string | The site or specify null for no site. |

Request Body Example

```
{
  "name": "router-a",
  "site": null
}
```

Response Example

```
{
  "device_manager_guid": "a2087abe-5753-4387-b372-f8cfb571bb1e"
}
```

Assign Device to Adapter

Use this API to assign a device to an adapter.

Request Method

POST

Request URL

`https://example-host/api/v2/apps/device-manager-srv/rest/device/{deviceGuid}/adapters/{adapterGuid}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| deviceGuid | string | The device guid. Use the device_manager_guid returned by the Get All Devices method. |
| adapterGuid | string | The adapter guid. Use the guid returned by the Get Adapters method. |

Request Body

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| description | string | <p>The adapter's device parameters. The schema can be retrieved with the Get Adapter Device Schema API.</p> <p>Ensure that you add the parameters that are marked as required. In this schema example:</p> <pre> "required": ["enabled", "host", "port"] </pre> |

Request Body Example

```

{
  "host": "1.1.1.1",
  "port": 22,
  "enabled": false
}

```

Response Example

201 Successful

Update Device Adapter Configuration

Use this API to update the device adapter configuration.

Request Method

PUT

Request URL

`https://example-host/api/v2/apps/device-manager-srv/rest/device/{deviceGuid}/adapters/{adapterGuid}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| deviceGuid | string | The device guid. Use the device_manager_guid returned by the Get All Devices method. |
| adapterGuid | string | The adapter guid. Use the guid returned by the Get Adapters method. |

Request Body

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| description | string | <p>The adapter's device parameters. The schema can be retrieved with the Get Adapter Device Schema API.</p> <p>Ensure that you add the parameters that are marked as required. In this schema example:</p> <pre> "required": ["enabled", "host", "port"] </pre> |

Request Body Example

```

{
  "host": "1.1.1.1",
  "port": 22,
  "enabled": true
}

```

Response Example

200 Successful

Unassign Device from Adapter

Use this API to unassign a device from an adapter. The device is not deleted from the network.

Request Method

DELETE

Request URL

`https://example-host/api/v2/apps/device-manager-srv/rest/device/{deviceGuid}/adapters/{adapterGuid}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| deviceGuid | string | The device guid. Use the device_manager_guid returned by the Get All Devices method. |
| adapterGuid | string | The adapter guid. Use the guid returned by the Get Adapters method. |

Response Example

200 Successful

Update Device Name

Use this API to update the device name.

Request Method

PUT

Request URL

```
https://example-host/api/v2/apps/device-manager-  
srv/rest/device/{deviceGuid}/name/{name}
```

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| deviceGuid | string | The device guid. Use the device_manager_guid returned by the Get All Devices method. |
| adapterGuid | string | The adapter guid. Use the guid returned by the Get Adapters method. |

Response Example

```
200 Successful
```

Assign Device to Site

Use this API to assign a device to site.

Request Method

```
PUT
```

Request URL

```
https://example-host/api/v2/apps/device-manager-srv/rest/device/{deviceGuid}/site
```

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| deviceGuid | string | The device guid. Use the device_manager_guid returned by the Get All Devices method. |

Request Body

| Parameter Name | Data Type | Description |
|----------------|-----------|----------------|
| guid | string | The site GUID. |

Request Body Example

```
{  
  "guid": "ST/e7df76d7a9cb"  
}
```

Response Example

```
200 Successful operation.
```

Get Device Status

Use this API to get device status.

Request Method

```
GET
```

Request URL

https://example-host/api/v2/apps/device-manager-srv/rest/device/{deviceGuid}/status

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| deviceGuid | string | The device guid. Use the device_manager_guid returned by the Get All Devices method. |

Response Example

```
{
  "adapte-name": {
    "last-successfull-discovery": "2020-05-04T17:50:15.530Z"
  }
}
```

Get a Device

Use this API to get a device.

Request Method

GET

Request URL

https://example-host/api/v2/apps/device-manager-srv/rest/ device/{deviceGuid}

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| deviceGuid | string | The device guid. Use the device_manager_guid returned by the Get All Devices method. |

Response Example

```
{
  "device_manager_guid": "83ce57e9-a82b-488a-994b-f8c4191f0158",
  "inventory_type": null,
  "device_model_guid": null,
  "name": "router-a",
  "adapters": {
    "cisco_ios_xr": {
      "adapter_guid": "cisco_ios_xr",
      "config": {
        "host": "1.1.1.2",
        "port": 22,
        "enabled": false
      },
      "enabled": false
    }
  }
},
```

```

"identifiers": {
  "host_name": null,
  "tid": null,
  "serial": null,
  "loopback_mac": null,
  "loopback_ip": null,
  "extra": {}
},
"site": null,
"pending_delete": false,
"enabled": false
}

```

Remove Device from Crosswork Hierarchical Controller

Use this API to remove the device from Crosswork Hierarchical Controller.

Request Method

DELETE

Request URL

```

https://example-host/api/v2/apps/device-manager-
srv/rest/device/{deviceGuid}?force=true

```

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| deviceGuid | string | The device guid. Use the device_manager_guid returned by the Get All Devices method. |
| force | string | Whether to remove the device from the system even if it is assigned to an adapter. |

Response Example

200 OK!

Get Credentials

Use this API to get all configured credentials keys' names and types.

Request Method

GET

Request URL

```

https://example-host/api/v2/apps/device-manager-srv/rest/credentials?type={type}

```

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|---------------|---|
| type | array[string] | SSH_USER_PASSWORD, SSH_PUBLIC_KEY, HTTP, SNMP_COMMUNITY, SFTP |

Response Example

```
[
  {
    "name": "A",
    "type": "SSH_USER_PASSWORD"
  },
  {
    "name": "Cisco",
    "type": "SSH_USER_PASSWORD"
  },
  {
    "name": "junos",
    "type": "SSH_USER_PASSWORD"
  }
]
```

Add New Credentials

Use this API to add new credentials.

Request Method

POST

Request URL

`https://example-host/api/v2/apps/device-manager-srv/rest/credentials/{name}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|----------------------|
| name | string | The credential name. |

Request Body

| Parameter Name | Data Type | Description |
|----------------|---------------|--|
| type | array[string] | SSH_USER_PASSWORD, SSH_PUBLIC_KEY, HTTP, SNMP_COMMUNITY, SFTP |
| config | description | The credentials parameters structure as described by the Get Credentials API. |

Request Body Example

```
{
  "type": "SSH_USER_PASSWORD",
  "config": {}
}
```

Response Example

201 Successful Operation

Update Credentials

Use this API to update credentials.

Request Method

PUT

Request URL

`https://example-host/api/v2/apps/device-manager-srv/rest/credentials/{name}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|----------------------|
| name | string | The credential name. |

Request Body

| Parameter Name | Data Type | Description |
|----------------|---------------|--|
| type | array[string] | SSH_USER_PASSWORD, SSH_PUBLIC_KEY, HTTP, SNMP_COMMUNITY, SFTP |
| config | description | The credentials parameters structure as described by the Get Credentials API. |

Request Body Example

```
{
  "type": "SSH_USER_PASSWORD",
  "config": {}
}
```

Response Example

200 Successful Operation

Delete Credentials

Use this API to delete credentials.

Request Method

DELETE

Request URL

`https://example-host/api/v2/apps/device-manager-srv/rest/credentials/{name}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|----------------------|
| name | string | The credential name. |

Response Example

200 Successful

Get the Parameters Schema for the Specified Credentials Type

Use this API to get the parameters schema for the specified credentials type.

Request Method

GET

Request URL

https://example-host/api/v2/apps/device-manager-srv/rest/XXX

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|---------------|---|
| type | array[string] | SSH_USER_PASSWORD, SSH_PUBLIC_KEY, HTTP, SNMP_COMMUNITY, SFTP |

Response Example

```
{
  "type": "object",
  "properties": {
    "username": {
      "title": "Username",
      "type": "string"
    },
    "password": {
      "title": "Password",
      "type": "string",
      "encrypted": true
    }
  },
  "required": [
    "username",
    "password"
  ]
}
```

Physical Inventory

The physical inventory API provides information on devices attributes, slots, cards, and ports.

This API supports optical devices only.

Get Physical Inventory

The API accepts two queries:

- Device ID - get full list of devices or specific device by its ID.
- Full - get detailed device inventory or only its ID (full=true/false).

Request Method

GET

Request URL

```
/api/v2/apps/network-inventory-app/rest/devices/
```

Request Header

```
Accept: application/yang-data+json
```

```
Content-Type: application/yang-data+json
```

Response Example

```
{
  "id": "XX-01011",
  "type": "ONE",
  "desc": "XX-01011",
  "vendor_model": "ciena_6500",
  "tid": "TID111",
  "serial_number": "[Shelf-1 [1-XX,2-XX]]",
  "SHELF": [
    {
      "id": "1",
      "type": "SHELF",
      "desc": "6500 14-Slot Converged Optical Shelf Assembly, OCP",
      "vendor_model": "ciena_6500",
      "tid": "XX-01011",
      "aid": "SHELF-1",
      "clei": "WWW",
      "serial_number": "SSS",
      "FAN": [
        {
          "id": "FAN-1-1",
          "type": "FAN",
          "desc": "XX-01011/MNG-EQPT/FAN-1-1 (FAN), ctype=\"Fan Rear Exhaust High Flow Cooling\"",
          "vendor_model": "ciena_6500",
          "tid": "XX-01011",
          "aid": "FAN-1-1",
          "clei": "WWW",
          "part_number": "NNN",
          "serial_number": "MMM"
        }
      ]
    }
  ],
  "CARD": [
    {
      "id": "1-1",
      "type": "CARD",
```

```

        "desc": "XX-01011/1-1 (OTSC), ctype=\"2x10G OTR 4x XFP\"",
        "vendor_model": "ciena_6500",
        "tid": "XX-01011",
        "aid": "000-1-1",
        "clei": "WWW",
        "part_number": "MM",
        "serial_number": "NNN",
        "PORT": [
            {
                "id": "1-1-1",
                "type": "PORT",
                "desc": "XX-01011/1-1-1 (P10GSOEL), ctype=\"OC192
SR1/I64.1 10GBASE-LR/LW OTU2 10GFC 1200SMLLL 1310 nm XFP\"",
                "vendor_model": "ciena_6500",
                "tid": "XX-01011",
                "aid": "UU-1-1-1",
                "clei": "WWW",
                "pluggable": "PLUGGABLE",
                "part_number": "NNNA",
                "serial_number": "MMM"
            },
        ],
    },
    "POWER_SUPPLY": [
        {
            "id": "PWR-1-17-1",
            "type": "POWER_SUPPLY",
            "desc": "XX-01011/MNG-EQPT/PWR-1-17-1 (PWR), ctype=\"Power Card
60A breaker\"",
            "vendor_model": "ciena_6500",
            "tid": "XX-01011",
            "aid": "PWR-1-17-1",
            "clei": "WWWD",
            "part_number": "NNNA",
            "serial_number": "MMM"
        },
    ]
}
]
}
}

```

Response Parameters

The response contains information on all device inventory modules:

- Device – all device level attributes
- Shelf – one or more shelves
- Fan – fan trays
- Card – all pluggable, line and aggregation cards, including transceivers
- Port – all physical ports
- Power supply – all power modules

Performance Counters

The Crosswork Hierarchical Controller performance API is based on a non-SDO data model authored by Cisco. The SDOs lack a network-level, performance data model allowing NBI access to raw performance counters of managed network objects.

Current performance-counter data models in IETF address the device level, such as ietf-interfaces, whereas network-level data models focus on derived values, e.g., bandwidth. Cisco will implement a standard data model once available from IETF.

The Crosswork Hierarchical Controller data model exposes raw performance counters collected from ports and IP/MPLS LSPs. It is implemented in the form of a YANG RPC where an NBI client may trigger an operation via the RESTCONF NBI to retrieve packet and/or octet counters to derive rate, average, etc.

URIs

You can retrieve the counters for the latest PM interval or for a specific period. The stats can be retrieved for a specific LSP/port/link, all objects, or for a group of objects (set by POST message).

| Interface Type | Interface Operation | Request Method | URI |
|----------------|---|----------------|---|
| Performance | RPC to retrieve raw performance counters per object type in a time range. | GET | /api/v2/timeseries/stats/ |
| Performance | RPC to retrieve raw performance counters per object type in the latest time range. | GET | /api/v2/timeseries/stats/latest |
| Performance | RPC to retrieve raw performance counters for specific ports (not one and not ALL the ports) | PUT | /api/v2/timeseries/stats/bulkStatsQuery |

Get Performance Latest Stats

Gets the performance stats for the latest period for an object or all objects. You can specify a specific PM type with a specific port/LSP or stream ID.

Request Method

GET

Request URL

api/v2/timeseries/stats/latest

Request Header

Accept: application/yang-data+json

Content-Type: application/yang-data+json

Request Parameters

| Parameter Name | Data Type | Value | Description |
|----------------|---------------|---------------------------------------|------------------|
| objGuid | string | | The object guid. |
| pmType | array[string] | DELAY_USEC, MIN_RTT_USEC OCTET_OUT | The PM type. |
| streamId | numeric | | The stream ID. |

Request Example

```
GET /api/v2/timeseries/stats/latest?pmType=DELAY_USEC&objGuid=PO/juniper-  
northstar/LGC-LabMX960-02:ae5.0
```

```
GET /api/v2/timeseries/stats/latest?pmType=DELAY_USEC&streamId=3
```

Request Parameters

| Parameter Name | Data Type | Value | Description |
|----------------|---------------|---------------------------------------|---------------------------------------|
| objGuid | string | | The object guid. |
| timestamp | datetime | | The time. |
| deviceType | string | | The device type, for example, ROUTER. |
| timePeriodSec | numeric | | The time in seconds. |
| pmType | array[string] | DELAY_USEC, MIN_RTT_USEC OCTET_OUT | The PM type. |
| value | numeric | | The performance statistic. |
| streamId | numeric | | The stream ID. |

Response Example

```
{  
  "objGuid":  
  "LI/lsp/f66fa3288d396e47/f66fa3288d396e47/f1e815107b715b67/f1e815107b715b67/lsp_161908  
  1766048",  
  "timeStamp": 1622602613000,  
  "deviceType": "ROUTER",  
  "timePeriodSec": 0,  
  "pmType": "OCTET_OUT",  
  "value": 273252081213835,  
  "streamId": 6  
}
```

Get Performance Stats for Period

Gets the performance stats for a specific period for an object or all objects. You can specify a specific PM type with a specific port/LSP or stream ID.

Request Method

GET

Request URL

api/v2/timeseries/stats

Request Header

Accept: application/yang-data+json

Content-Type: application/yang-data+json

Request Parameters

| Parameter Name | Data Type | Value | Description |
|----------------|---------------|---------------------------------------|--------------------------|
| startTimeStamp | datetime | | The start of the period. |
| endTimeStamp | datetime | | The end of the period. |
| objGuid | string | | The object guid. |
| pmType | array[string] | DELAY_USEC, MIN_RTT_USEC OCTET_OUT | The PM type. |
| streamId | numeric | | The stream ID. |

Request Example

```
GET /api/v2/timeseries/stats?startTimeStamp=1643793190000&
endTimeStamp=1643878292000&pmTypeDELAY_USEC&objGuid=
PO/r_logical/16f1596a46b13da5/7d82f458ff24bfa3
```

Request Parameters

| Parameter Name | Data Type | Value | Description |
|----------------|---------------|---------------------------------------|---------------------------------------|
| objGuid | string | | The object guid. |
| Timestamp | datetime | | The time. |
| deviceType | string | | The device type, for example, ROUTER. |
| timePeriodSec | numeric | | The time in seconds. |
| pmType | array[string] | DELAY_USEC, MIN_RTT_USEC OCTET_OUT | The PM type. |
| Value | numeric | | The performance statistic. |
| streamId | numeric | | The stream ID. |

Response Example

```
{
```

```
"objGuid": "PO/r_logical/16f1596a46b13da5/7d82f458ff24bfa3",
"timeStamp": 1643874592000,
"deviceType": "ROUTER",
"timePeriodSec": 859,
"pmType": "DELAY_USEC",
"value": 649907,
"streamId": 5
}
{
"objGuid": "PO/r_logical/16f1596a46b13da5/7d82f458ff24bfa3",
"timeStamp": 1643875520000,
"deviceType": "ROUTER",
"timePeriodSec": 928,
"pmType": "DELAY_USEC",
"value": 724091,
"streamId": 5
}
{
"objGuid": "PO/r_logical/16f1596a46b13da5/7d82f458ff24bfa3",
"timeStamp": 1643876484000,
"deviceType": "ROUTER",
"timePeriodSec": 927,
"pmType": "DELAY_USEC",
"value": 69079,
"streamId": 5
}
{
"objGuid": "PO/r_logical/16f1596a46b13da5/7d82f458ff24bfa3",
"timeStamp": 1643877388000,
"deviceType": "ROUTER",
"timePeriodSec": 904,
"pmType": "DELAY_USEC",
"value": 665969,
"streamId": 5
}
```

Send Specific Ports Performance Stats

Gets the performance stats for a specific period for specific ports (not one and not ALL the ports).

Request Method

PUT

Request URL

```
api/v2/timeseries/stats/bulkStatsQuery
```

Request Header

```
Accept: application/yang-data+json
```

```
Content-Type: application/yang-data+json
```

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-------------|--|
| startTimeStamp | datetime | The start of the period. |
| endTimeStamp | datetime | The end of the period. |
| d@ports.json | JSON string | Where ports.json is list of the guids. |

Request Example

```
POST
/api/v2/timeseries/stats/bulkStatsQuery?startTimeStamp=1659402468000&endTimeStamp=1659
802468000' -d@ports.json
```

Where request.json is list of the guids:

```
[
  "PO/r_logical/1d80163d31fa18ee/870a15983eafb41f",
  "PO/r_logical/7b4564a1ccb6e4cb/5a52cb94b8d7dbc4"
]
```

Response Parameters

| Parameter Name | Data Type | Value | Description |
|----------------|---------------|-------------------------------------|---------------------------------------|
| objGuid | string | | The object guid. |
| timeStamp | datetime | | The time. |
| deviceType | string | | The device type, for example, ROUTER. |
| timePeriodSec | numeric | | The time in seconds. |
| pmType | array[string] | DELAY_USEC, MIN_RTT_USEC, OCTET_OUT | The PM type. |
| value | numeric | | The performance statistic. |
| streamId | numeric | | The stream ID. |

Response Example

```
[
  {
    "objGuid": "PO/r_logical/7b4564a1ccb6e4cb/5a52cb94b8d7dbc4",
    "timeStamp": 1659608730000,
```

```
"deviceType": "ROUTER",
"timePeriodSec": 900,
"pmType": "MAX_RTT_USEC",
"value": 481463,
"streamId": 4
},
{
"objGuid": "PO/r_logical/7b4564a1ccb6e4cb/5a52cb94b8d7dbc4",
"timeStamp": 1659608730000,
"deviceType": "ROUTER",
"timePeriodSec": 900,
"pmType": "DELAY_USEC",
"value": 443345,
"streamId": 4
},
{
"objGuid": "PO/r_logical/7b4564a1ccb6e4cb/5a52cb94b8d7dbc4",
"timeStamp": 1659608730000,
"deviceType": "ROUTER",
"timePeriodSec": 900,
"pmType": "MIN_RTT_USEC",
"value": 441197,
"streamId": 4
},
{
"objGuid": "PO/r_logical/1d80163d31fa18ee/870a15983eafb41f",
"timeStamp": 1659608735000,
"deviceType": "ROUTER",
"timePeriodSec": 900,
"pmType": "OCTET_OUT",
"value": 17939855810587,
"streamId": 4
}
}
```

Cross Links

Crosswork Hierarchical Controller provides APIs to administer cross links. For more information, see the *Cisco Crosswork Hierarchical Controller Administration Guide*.

The APIs include:

- Get all adjacent cross links
- Validate all cross links
- Validate a cross link

-
- Get all manual cross links
 - Get all manual cross links
 - Get a cross link
 - Add manual cross link to Crosswork Hierarchical Controller
 - Delete a cross link

Note: If the user does not have permission to execute the API, 403 Forbidden is returned. If the wrong credentials are used, 401 Authorization Required is returned.

Get All Adjacent Cross Links

Use this API to return port adjacencies that were discovered using automatic discovery protocols such as LLDP, elements in this list MAY be adjacencies that were not configured and not part of the manual cross list.

Request Method

GET

Request URL

`https://example-host/api/v2/crosslinks/adjacencies`

Request Parameters

None

Response Example

The response contains information on all adjacent cross links. For cross link it returns the following:

- interfaceA – deviceName, interfaceName, type, and guid
- interfaceB – deviceName, interfaceName, type, and guid
- technology (ETH or NMC)
- relatedModelObjectGuid

Validate All Cross Links

Use this API to trigger a validation all cross links. This runs a process to update the model (links and paths) based on the latest state of cross links, router-physical links, eth links, etc. Validating all cross links (and adding a new manual cross link) may update the path of a r-phy link and may even create the r-phy link altogether if it did not exist before. As a result of the new manual cross link, this may find the entire optical circuit from end-to-end.

For an Ethernet link, if there is a conflict between a manually added cross link and a cross link detected from the network, the manually added link is removed from Cisco Crosswork Hierarchical Controller network model. The manual link remains in the manual-links db and is re-added to the network model if the network-discovered link disappears AND the manual link gets a positive validation after there are no more conflicts (this is in contrast to never-conflicted manual links that appear in the network model even without a positive validation).

This API method returns an empty result. Use the [Get All Manual Cross Links](#) method to view the updated state.

Request Method

GET

Request URL

`https://example-host/api/v2/crosslinks/validate`

Request Parameters

None

Response Example

None

Validate a Cross Link

Use this API to trigger a validation of a specific cross link. This API only updates the relevant validation fields of the validated manual cross link without touching the topological model.

Request Method

GET

Request URL

`https://example-host/api/v2/crosslinks/validate/{guid}`

Response Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|---|
| guid | string | The cross link guid. Use the guid returned by the Get All Manual Cross Links method. |

Response Example

The response returns the following for the cross link:

- **guid**: The link guid
- **interfaceA** - deviceName, interfaceName, type, and guid
- **interfaceB** - deviceName, interfaceName, type, and guid
- **technology** (ETH or NMC)
- **relatedModelObjectGuid**
- **additionTimeEpochMillis**
- **description**
- **validated**: Whether the link has been validated.
- **validationResult**: The validation result (VALIDATED_BY_PORT_ADJACENCY, VALIDATED_BY_CORRELATION, NOT_CORRELATED, INCONCLUSIVE, UNKNOWN, or CONFLICT).
- **lastValidationCheck**: The timestamp when the link was last validated.
- **lastConflictTime**: This timestamp is hidden if there was never a conflict. If this is the same as the **lastValidationCheck**, then the manual link is currently in conflict. If **lastValidationCheck** is later

than the **lastConflictTime**, then the manual link was in conflict before but is not in conflict currently. If in addition **validated** is **true**, this manual link appears in the network model.

- **conflicts**: If a link is or was conflicted, the **conflicts** list is populated with one or two conflicting pairs, one pair per each of the manual-link ports.

```
{
  "guid": "23f6e6c340268f068f817f8f9484aa54a2ba92d5d7072efd7940308f8ecc8fa5",
  "interfaceA": {
    "deviceName": "CR2.ATH",
    "interfaceName": "TenGigE0/0/1/13",
    "type": "R_PHYSICAL",
    "guid": "PO/r_physical/8ff611ed7ebcd772/2c4e6530608a213d"
  },
  "interfaceB": {
    "deviceName": "SD2ATH02",
    "interfaceName": "ETH-1-1-8",
    "type": "ETH",
    "guid": "PO/eth/ecf4e9a6bbb76cbc/86cdb5c6379e5c62"
  },
  "technology": "ETH",
  "relatedModelObjectGuid":
  "LI/eth/8ff611ed7ebcd772/2c4e6530608a213d/ecf4e9a6bbb76cbc/86cdb5c6379e5c62",
  "additionTimeEpochMillis": 1666695054475,
  "description": "cross link description",
  "validated": true,
  "validationResult": "VALIDATED_BY_PORT_ADJACENCY",
  "lastValidationCheck": 1668598747209,
  "conflicts": [
    {
      "a": { "deviceName": "dev-name-of-manual-link-interfaceA", "interfaceName":
"iface-name-of-manual-link-interfaceA" },
      "b": { "deviceName": "dev-name-of-different-than-manual-link-interfaceB",
"interfaceName": "iface-name-different-than-manual-link-interfaceB" }
    },
    {
      "a": { "deviceName": "dev-name-of-manual-link-interfaceB", "interfaceName":
"iface-name-of-manual-link-interfaceB" },
      "b": { "deviceName": "dev-name-of-different-than-manual-link-interfaceA",
"interfaceName": "iface-name-different-than-manual-link-interfaceA" }
    }
  ]
}
```

Get All Manual Cross Links

Use this API to return an array of the manual cross links. The guid returned can be used as a request parameter in the other cross link APIs.

Request Method

GET

Request URL

`https://example-host/api/v2/crosslinks/manual`

Request Parameters

None

Response Example

The response returns an array of cross links with the following:

- guid
- interfaceA - deviceName, interfaceName, type, and guid
- interfaceB - deviceName, interfaceName, type, and guid
- technology (ETH or NMC)
- relatedModelObjectGuid
- additionTimeEpochMillis
- description
- validated
- validationResult (VALIDATED_BY_PORT_ADJACENCY, VALIDATED_BY_CORRELATION, NOT_CORRELATED, INCONCLUSIVE, UNKNOWN, CONFLICT)
- lastValidationCheck
- conflicts

```
{
  "guid": "23f6e6c340268f068f817f8f9484aa54a2ba92d5d7072efd7940308f8ecc8fa5",
  "interfaceA": {
    "deviceName": "CR2.ATH",
    "interfaceName": "TenGigE0/0/1/13",
    "type": "R_PHYSICAL",
    "guid": "PO/r_physical/8ff611ed7ebcd772/2c4e6530608a213d"
  },
  "interfaceB": {
    "deviceName": "SD2ATH02",
    "interfaceName": "ETH-1-1-8",
    "type": "ETH",
    "guid": "PO/eth/ecf4e9a6bbb76cbc/86cdb5c6379e5c62"
  },
}
```

```

    "technology": "ETH",
    "relatedModelObjectGuid":
"LI/eth/8ff611ed7ebcd772/2c4e6530608a213d/ecf4e9a6bbb76cbc/86cdb5c6379e5c62",
    "additionTimeEpochMillis": 1666695054475,
    "description": "cross link description",
    "validated": true,
    "validationResult": "VALIDATED_BY_PORT_ADJACENCY",
    "lastValidationCheck": 1668598747209,
    "conflicts": []
}

```

Get a Cross Link

Use this API to get a manual cross link.

Request Method

GET

Request URL

`https://example-host/api/v2/crosslinks/manual/{guid}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|---|
| guid | string | The cross link guid. Use the guid returned by the Get All Manual Cross Links method. |

Response Example

The response returns the following for the cross link:

- guid
- interfaceA - deviceName, interfaceName, type, and guid
- interfaceB - deviceName, interfaceName, type, and guid
- technology (ETH or NMC)
- relatedModelObjectGuid
- additionTimeEpochMillis
- description
- validated: Whether the link has been validated.
- validationResult: The validation result (VALIDATED_BY_PORT_ADJACENCY, VALIDATED_BY_CORRELATION, NOT_CORRELATED, INCONCLUSIVE, UNKNOWN, or CONFLICT).
- lastValidationCheck: The timestamp when the link was last validated.
- lastConflictTime: This timestamp is hidden if there was never a conflict. If this is the same as the **lastValidationCheck**, then the manual link is currently in conflict. If **lastValidationCheck** is later than the **lastConflictTime**, then the manual link was in conflict before but is not in conflict currently. If in addition **validated** is **true**, this manual link appears in the network model.

- **conflicts**: If a link is or was conflicted, the **conflicts** list is populated with one or two conflicting pairs, one pair per each of the manual-link ports.

```
{
  "guid": "fa6262a2f1202b4bba1dd699cc78b2cf9cc45c826487de6daad69e3fa0cf0a90",
  "interfaceA": {
    "deviceName": "CR2.BCN",
    "interfaceName": "TenGigE0/0/1/8",
    "type": "R_PHYSICAL",
    "guid": "PO/r_physical/b876eefb0f288974/146956e90f8b5b6d"
  },
  "interfaceB": {
    "deviceName": "OTN1MIL01",
    "interfaceName": "1-2-3",
    "type": "ETH",
    "guid": "PO/eth/5979a210307b1e66/fba4016fb0ebde72"
  },
  "technology": "ETH",
  "relatedModelObjectGuid":
  "LI/CL/PO/r_physical/b876eefb0f288974/146956e90f8b5b6d/PO/eth/5979a210307b1e66/fba4016fb0ebde72",
  "additionTimeEpochMillis": 1668597718579,
  "description": "example",
  "validated": false,
  "validationResult": "UNKNOWN",
  "lastValidationCheck": 1668597744370,
  "conflicts": []
}
```

Add Manual Cross Link to Crosswork Hierarchical Controller

Use this API to add a manual cross link to Crosswork Hierarchical Controller.

Request Method

POST

Request URL

<https://example-host/api/v2/apps/crosslinks/manual>

Request Parameters

None

Request Body

| Parameter Name | Data Type | Description |
|----------------|-----------|-------------|
| interfaceA | | Port A |

| Parameter Name | Data Type | Description |
|----------------|-----------|---------------------|
| deviceName | string | The device name. |
| interfaceName | string | The interface name. |
| interfaceB | | Port B |
| deviceName | string | The device name. |
| interfaceName | string | The interface name. |
| technology | string | The technology. |
| Description | string | The description. |

Request Body Example

```
{
  "interfaceA": {
    "deviceName": "CR2.BCN",
    "interfaceName": "TenGigE0/0/1/8"
  },
  "interfaceB": {
    "deviceName": "OTN1MIL01",
    "interfaceName": "1-2-3"
  },
  "technology": "ETH",
  "description": "example"
}
```

Response Example

```
{"guid": "d33ac2ec12c237e3a53bef30aec690e8f1ecff1a9c600c98b406ea9be30e91b7"}
```

Delete a Cross Link

Use this API to delete a manual cross link. The cross link is marked as deleted and is removed when the next validation runs.

Request Method

```
DELETE
```

Request URL

```
https://example-host/api/v2/crosslinks/manual/{guid}
```

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|---|
| guid | string | The cross link guid. Use the guid returned by the Get All Manual Cross Links method. |

Response Example

200 OK

Shared Risk

Crosswork Hierarchical Controller provides APIs to administer shared risk policies and rules.

You can access the Shared Risk API using Swagger: <https://<host>/api/v2/apps/srlg-app/rest/doc>

The APIs include:

- Get a specific policy
- Get all policies
- Create a policy
- Delete a policy
- Change the shared risk type of the policy
- Change a policy type
- Add a new rule to a policy
- Update the rule resources
- Delete a rule from a policy

Get Policies

Use this API to get the list of all the policies. This returns a list of all the policies and their rules.

Request Method

GET

Request URL

`https://example-host/api/v2/apps/srlg-app/rest/policy`

Request Parameters

None

Response Example

```
{
  "name": "policy-1",
  "shared_risk_types": [
    "Link",
    "Port",
    "Card",
    "Shelf",
    "Device"
  ],
  "policy_type": "MULTIPLE-LINKS",
  "rules": [
    {
      "name": "rule-1",
```

```

    "resources": [
      "LI/eth/000fc44c94a1f2cd/51308dfd752c1574/df753d953c1e1c8f/f8e7b20537ce03b7"
    ]
  },
  {
    "name": "rule99",
    "resources": [
      "inventory[.name=\"CR1.PAR\"]|port|link[.layer=\"R_LOGICAL\"]"
    ]
  }
],
{
  "name": "test",
  "shared_risk_types": [
    "Link",
    "Device",
    "Shelf",
    "Port",
    "Card"
  ],
  "policy_type": "MULTIPLE-LINKS",
  "rules": [
    {
      "name": "rule001",
      "resources": [
        "inventory[.name=\"ILA-SD1EVO01-SD1SEV01-1\"]|port|link[.layer=\"R_LOGICAL\"]"
      ]
    }
  ]
},
{
  "name": "policy-3",
  "shared_risk_types": [
    "Link"
  ],
  "policy_type": "SINGLE-PROTECTED",
  "rules": [
    {
      "name": "rule-99",
      "resources": [

```

```

        "link[.layer=\"R_LOGICAL\"]"
    ]
}
]
}

```

Get a Policy

Use this API to retrieve a policy.

Request Method

GET

Request URL

`https:// example-host/api/v2/apps/srlg-app/rest/policy/{policyGuid}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| policyGuid | string | The policy guid. Use the guid returned by the Get Policies method. |

Response Example

```

{
  "name": "policy-1",
  "shared_risk_types": [
    "Link",
    "Port",
    "Card",
    "Shelf",
    "Device"
  ],
  "policy_type": "MULTIPLE-LINKS",
  "rules": [
    {
      "name": "rule-1",
      "resources": [
        "LI/eth/000fc44c94a1f2cd/51308dfd752c1574/df753d953c1e1c8f/f8e7b20537ce03b7"
      ]
    },
    {
      "name": "rule99",
      "resources": [
        "inventory[.name=\"CR1.PAR\"]|port|link[.layer=\"R_LOGICAL\"]"
      ]
    }
  ]
}

```

```
}
```

Create a Policy

Use this API to create a policy.

Request Method

POST

Request URL

`https://example-host/api/v2/apps/srlg-app/rest/policy/{policyGuid}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|---|
| policyGuid | string | The policy guid . Use the guid returned by the Get Policies method. |

Request Body

| Parameter Name | Data Type | Description |
|-------------------|-----------|-------------------------------------|
| shared_risk_types | string | Link, Port, Card, Shelf, Device |
| policy_type | string | SINGLE-PROTECTED or MULTIPLE-LINKS. |

Request Body Example

```
{
  "shared_risk_types": [
    "Link"
  ],
  "policy_type": "SINGLE-PROTECTED"
```

Response Example

```
201 Successful Operation
```

Delete Policy

Use this API to delete a policy.

Request Method

DELETE

Request URL

`https://example-host/api/v2/apps/srlg-app/rest/policy/{policyGuid}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| policyGuid | string | The policy guid. Use the guid returned by the Get Policies method. |

Response Example

```
200 Successful
```


Update Policy Shared Risk Types

Use this API to change the policy shared risk types.

Request Method

PUT

Request URL

`https://example-host/api/v2/apps/srlg-app/rest/policy/{policyGuid}/shared_risk_types`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| policyGuid | string | The policy guid. Use the guid returned by the Get Policies method. |

Request Body

| Parameter Name | Data Type | Description |
|-------------------|-----------|---------------------------------|
| shared_risk_types | string | Link, Port, Card, Shelf, Device |

Request Body Example

```
{
  "shared_risk_types": [
    "Link"
  ]
}
```

Response Example

200 Successful Operation

Update Policy Type

Use this API to update credentials.

Request Method

PUT

Request URL

`https://example-host/api/v2/apps/srlg-app/rest/policy/{policyGuid}/policy-type`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| policyGuid | string | The policy guid. Use the guid returned by the Get Policies method. |

Request Body

| Parameter Name | Data Type | Description |
|----------------|-----------|-------------------------------------|
| policy_type | string | SINGLE-PROTECTED or MULTIPLE-LINKS. |

Request Body Example

```
{
```

```
    "policy_type": "SINGLE-PROTECTED"
  }
```

Response Example

200 Successful Operation

Add a Rule to a Policy

Use this API to add a rule to a policy. You can use an array of GUIDs and/or an SHQL query to create the rule.

Request Method

POST

Request URL

`https://example-host/api/v2/apps/srlg-app/rest/policy/{policyGuid}/rules{ruleName}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| policyGuid | string | The policy guid. Use the guid returned by the Get Policies method. |
| ruleName | string | The rule name. Use one of the rule names returned by the Get Policies method. |

Request Body

| Parameter Name | Data Type | Description |
|----------------|---------------|--|
| resources | array(string) | A list of GUID links and/or an SHQL query. If you use an SQL query, make sure that the expression is valid and returns a result. See the SQL User Guide. When you pass an SQL query, ensure that you wrap "" with a pair of \s, for example: "link[.layer=\ "R_LOGICAL\ "]" |

Request Body Example

```
{
  "resources": [
    "link[.layer=\ "R_LOGICAL\ "]"
  ]
}
```

or

```
{
  "resources": [
    "LI/guid1",
    "LI/guid2"
  ]
}
```

or

```
{
```

```

    "resources": [
      "inventory[.name=\"CR1.PAR\"]|port|link[.layer=\"R_LOGICAL\"]"
    ]
  }
}

```

Response Example

201 Successful Operation

Update a Rule

Use this API to update the rule's resources. You can use an array of GUIDs and/or an SHQL query to create the rule.

Request Method

PUT

Request URL

`https://example-host/api/v2/apps/srlg-app/rest/policy/{policyGuid}/rules{ruleName}`

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| policyGuid | string | The policy guid. Use the guid returned by the Get Policies method. |
| ruleName | string | The rule name. Use one of the rule names returned by the Get Policies method. |

Request Body

| Parameter Name | Data Type | Description |
|----------------|---------------|--|
| resources | array(string) | <p>A list of GUID links and/or an SHQL query.</p> <p>If you use an SQL query, make sure that the expression is valid and returns a result. See the SQL User Guide.</p> <p>When you pass an SQL query, ensure that you wrap "...." with a pair of \s, for example: "link[.layer=\"R_LOGICAL\"]"</p> |

Request Body Example

```

{
  "resources": [
    "link[.layer=\"R_LOGICAL\"]"
  ]
}

```

or

```

{
  "resources": [
    "LI/guid1",
    "LI/guid2"
  ]
}

```

or

```
{
  "resources": [
    "inventory[.name=\"CR1.PAR\"]|port|link[.layer=\"R_LOGICAL\"]"
  ]
}
```

Response Example

```
201 Successful Operation
```

Delete a Rule from a Policy

Use this API to delete a rule from a policy.

Request Method

```
DELETE
```

Request URL

```
https://example-host/api/v2/apps/srlg-app/rest/policy/{policyGuid}
/policy/{policyGuid}/rules/{ruleName}
```

Request Parameters

| Parameter Name | Data Type | Description |
|----------------|-----------|--|
| policyGuid | string | The policy guid. Use the guid returned by the Get Policies method. |
| ruleName | string | The rule name. Use one of the rule names returned by the Get Policies method. |

Response Example

```
200 Successful
```

Generic Information Retrieval using SHQL

Crosswork Hierarchical Controller provides a specialized query language designed for network data, called Sedona Hierarchical Query Language (SHQL). SHQL allows users to specify any query and quickly see the results, as well as to save the query for the benefit of others.

SHQL is unique in the way that it allows users to query a multi-layer model using simple keywords. Keywords allow for navigating the model, transitioning from one object type to another, and eliminating the need for complex conditions.

Transitioning is when SHQL allows for retrieving a list of objects and using this to transition to related objects, for example, retrieve all the SITES in the system and then use a language operator “|” (pipe operator) to get all INVENTORY in those sites etc.

For example, all OMSs used by a specific LSP can be retrieved using just a ‘downward’ command that retrieves, for the specified LSP, all links in the specified lower layer.

```
link[.name="my_lsp"] | downward ("OMS")
```

SHQL commands and syntax are described in the *Crosswork Hierarchical Controller SHQL Guide*.

SHQL can be used in the UI application in Crosswork Hierarchical Controller or as REST POST commands.

Request Method

POST

Request URL

https://<server>/api/v2/shql

Request Example

In this example, the query gets the list of OMSs that are the underlay for all LSPs in operational state down.

```
link[.layer="LSP" and .operStatus="DOWN"] | downward | link[.layer="OMS" and .operStatus="DOWN"]
```

Response Example

```
{
  "activeProtectionPriority": null,
  "bidi": true,
  "desc": null,
  "distanceMeters": null,
  "extra": null,
  "guid":
  "LI/oms/af5e85ffc6049e8f/8c290fec341b62da/9bf4b791d3191519/3837d2f977f671bd",
  "inverseLinkId": null,
  "latencyMicros": null,
  "layer": "OMS",
  "name": "SD1BKL01/1-2-5&8 to SD1SLO01/1-3-5&8",
  "operStatus": "DOWN",
  "pathGroupType": "SINGLE_PATH",
  "paths": [
    {
      "guid":
      "PA/oms/af5e85ffc6049e8f/8c290fec341b62da/9bf4b791d3191519/3837d2f977f671bd"
    }
  ],
  "portA": {
    "guid": "PO/oms/af5e85ffc6049e8f/8c290fec341b62da",
    "type": "OMS"
  },
  "portB": {
    "guid": "PO/oms/9bf4b791d3191519/3837d2f977f671bd",
    "type": "OMS"
  },
  "protectionStatus": "N_A",
  "provider": "Topogen",
  "role": "REGULAR",
  "srlgs": null,
  "teMetric": null
}
```

Introducing SSQL

SSQL - The Need

Cisco Crosswork Hierarchical Controller provides valuable information on network structure and enables you to simulate various conditions using several dedicated applications.

You may, however, want to perform your own analytics and view the resulting reports. Querying a multi-dimensional data model is complex and requires deep understanding of the model and advanced dev skills. For example:

- Filter one layer by another
- Get data from different times
- Find anomalies
- Group links by attributes in multiple dimensions

The model data is multi-dimensional and includes:

- Vendors
- Topologies
- Layers
- Domains
- Status
- Time

The SSQL application solves this problem by implementing cross-queries between the layers and dimension tables. It hides data complexity and provides a simple, yet powerful query language that enables you to easily query the model across all these dimensions.

SSQL Overview

The SSQL application enables you to quickly create complex query commands and send them as requests to the Crosswork Hierarchical Controller SSQL REST API. Accessed through your browser, the SSQL application allows you to run queries and view an orderly distribution of results under their relevant column headings. The type and number of columns are based on the properties of the object type displayed.

The application covers every functionality of the query language. Auto-completion context menus allow you to easily select viable object types and their related properties and apply multiple conditions using a range of operands.

SSQL allows you to manipulate the displayed information by transforming the query from one object type to another within the same query command line. Alternatively, you can specify a collection of object types to which all the query items are related.

Note: The term object type used in this guide is defined as an item or group of items that share the same properties.

Crosswork Hierarchical Controller Object Model

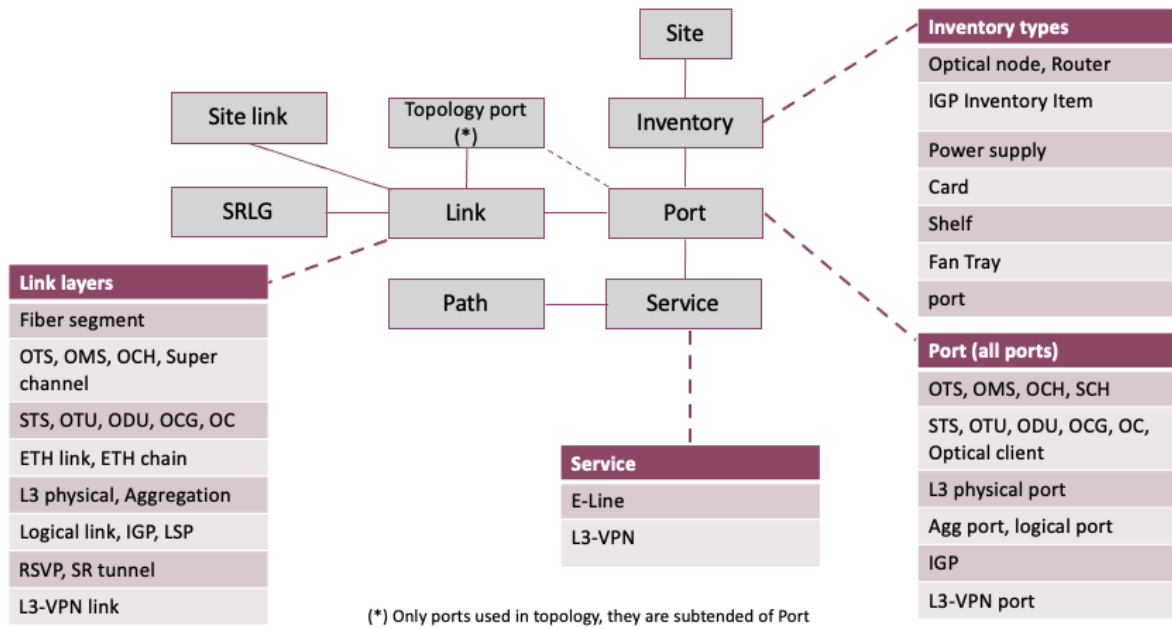


Figure 1.
SHQL Object Model

Inventory Types

Table 2. Inventory Types

| Inventory | Description |
|--------------------|--|
| Optical Node | An optical network element that is actively managed. This can be an amplifier, ROADM, transponder, or muxponder. |
| Router | A distinct instance of a router that has a loopback IP address and physical inventory. |
| IGP Inventory Item | A logical/virtual router as an instance in router. |
| Power Supply | A power supply module in an optical node or router. |
| Card | Any type of pluggable card. |
| Shelf | A chassis of a router or optical node. |
| Fan Tray | A pluggable or manageable fan tray unit in an optical node or router. |

Link Types

Table 3. Link Types

| Inventory | Description |
|---------------|---|
| Fiber Segment | A physical fiber line that spans from one DUCT to another and is used as a segment in a fiber link. |

| Inventory | Description |
|--------------|--|
| Fiber | A chain of fiber segments that spans from one optical device to another. |
| OTS | The logical layer used as an underlay for the OMS link. One OTS link can be created over a fiber link. |
| OMS | The logical layer used as an underlay for the OCH link. One OMS link can be created over a chain of OTS links. |
| NMC | A wavelength connection between two ROADMs over one or more OMS links, not connecting to client ports on muxponder. |
| OCH | A wavelength connection spanning the client port of one OEO device (transponder, muxponder, regenerator) and another. 40 or 80 OCH links can be created on top of an OMS link. An OCH link spans from one client port to another, where the client port is either a TDM or ETH port. |
| SCH | An OCH that spans from one ROADM device to another and terminates in a device and not in a client port. |
| STS, OCG, OC | The TDM links that span from one optical device to another, ride on top of OCH links and terminate in TDM client ports |
| OTU | The underlay link in the OTN layer, used for ODU links. It can ride on top of an OCH. |
| ODU | The sub signals in OTU links. Each OTU link can carry multiple ODU links, and ODU links can be divided into finer granularity ODU links recursively. |
| ETH Link | An ETH L2 link, that spans from one ETH UNI port to another, and rides on top of an ODU link. |
| ETH Chain | A chain of ETH links. |
| L3 Physical | An ethernet link (L2 connectivity) between two ETH ports. This is the underlay layer for an IP subnet link between two routers. |
| Agg | The Link Aggregation Group (LAG) where multiple ETH links are grouped to create higher BW and a resilient link. |
| Logical Link | A link where two IP subnets are connected over a physical link between two routers. |
| IGP | The path calculated by the router as the shortest path between two specific nodes. The link between two routers that carries IGP protocol messages. The link represents an IGP adjacency. |
| Tunnel | The MPLS RSVP, or Segment Routing tunnel created between two routers over IGP links, with or without TE options. |
| L3-VPN Link | The connection between two sites of a specific L3-VPN (can be a chain of LSP connections or an IGP path). |

Crosswork Hierarchical Controller Connection Model

For example: 10GE over 100G OTN link

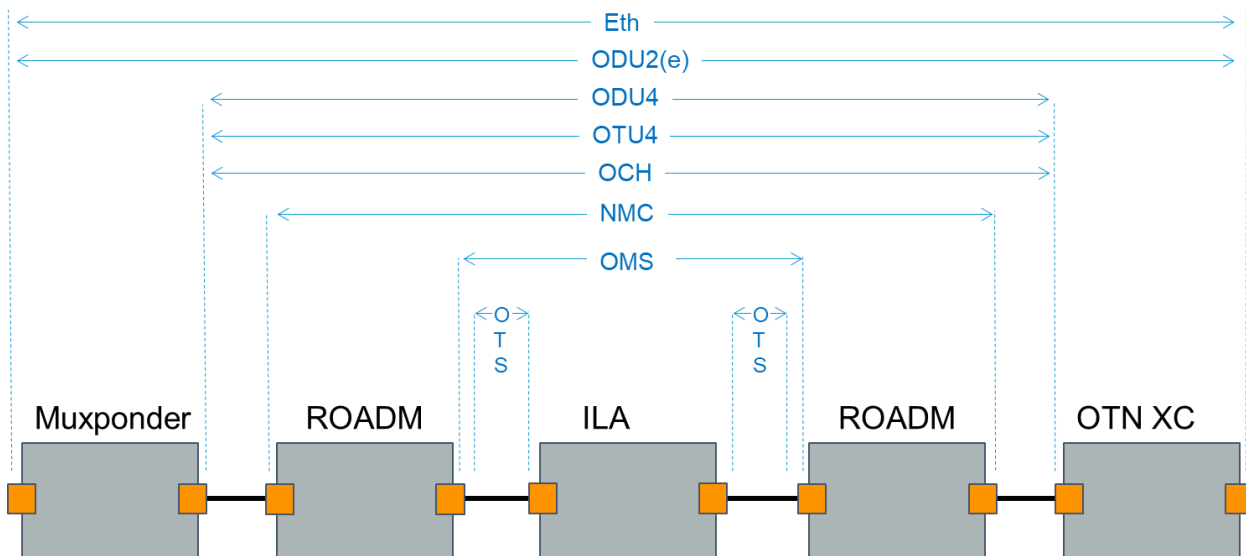


Figure 2.
Connection Model

SQL Example

This example finds if there are shared OMSs between two LSPs.

The SHQL interface shows a query designed to find shared OMSs between two LSPs. The query is as follows:

```
link[.name="CR2.DUS:CR2.BIL:lsp_0"] | downward | link[.layer="OMS"] as p; link[.name="CR2.DUS:CR1.BIL:lsp_0"] | downward | link[.layer="OMS"] as w; link[.guid in p and .guid in w]
```

The results table shows 9 OMS links:

| Guid | Layer | Name | OperStatus | PathGroupType | Paths | PortA |
|-------------------------------|-------|-------------------------------|------------|---------------|---|-------------------------------|
| Lj/oms/022fa130aca541bb/38... | OMS | SD1GIR01/1-3-5&8 to SD1PAM... | UP | SINGLE_PATH | [{"guid": "PA/oms/022fa130aca541bb/38..."}] | PO/oms/022fa130aca541bb/38... |
| Lj/oms/b0bd0180e1c79218/8... | OMS | SD1BCN01/1-2-5&8 to SD1MA... | UP | SINGLE_PATH | [{"guid": "PA/oms/b0bd0180e1c79218/8..."}] | PO/oms/b0bd0180e1c79218/... |
| Lj/oms/b0bd0180e1c79218/0... | OMS | SD1BCN01/1-5-5&8 to SD1GIR... | UP | SINGLE_PATH | [{"guid": "PA/oms/b0bd0180e1c79218/0..."}] | PO/oms/b0bd0180e1c79218/... |
| Lj/oms/990aae8f0c12944/38... | OMS | SD1STU01/1-3-5&8 to SD1ZU... | UP | SINGLE_PATH | [{"guid": "PA/oms/990aae8f0c12944/38..."}] | PO/oms/990aae8f0c12944/38... |
| Lj/oms/738790a2aca5de20/0... | OMS | SD1MAR01/1-5-5&8 to SD1MIL... | UP | SINGLE_PATH | [{"guid": "PA/oms/738790a2aca5de20/0..."}] | PO/oms/738790a2aca5de20/0... |

Figure 3.
SHQL Example

Using SHQL

About the SHQL Interface

In your browser, open the SHQL application by clicking its button on the sidebar.

Note: It is recommended that you use Chrome.

Figure 4.
SHQL Application Button

The SHQL application interface is displayed. The field in which you insert your query is empty.

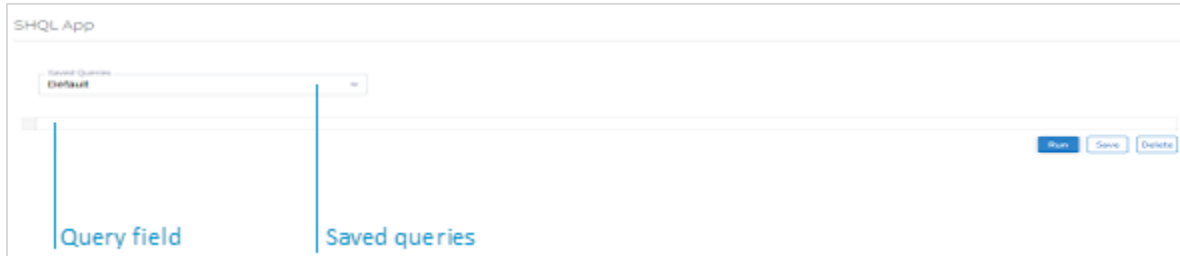


Figure 5.
SHQL Interface

Once you have run a query, you can save it for future use (see Saving Queries).

Running Queries

To run a query:

1. Enter a query command into the **Query** field (see Creating Queries).
2. Click **Run**.

The retrieved data is organized under its related tabs.

Figure 6 shows a list of results under two tabs after running the query, **inventory_item**.

The screenshot shows the SHQL interface with the query 'inventory_item' entered and the 'Run' button clicked. The results are displayed under the 'Optical Node (463)' tab. The table shows 463 items, with the first 10 rows visible. The columns are: Guid, Type, Name, Provider, DeviceFamily, DeviceType, ReachabilityStatus, Site, Tags, Vendor, and HasRoadm.

| Guid | Type | Name | Provider | DeviceFamily | DeviceType | ReachabilityStatus | Site | Tags | Vendor | HasRoadm |
|-------------------|------|-------------------|----------|--------------|------------|--------------------|-------------------|----------------------|--------|----------|
| IN/002d237f16b... | ONE | ILA-SD1EV001-S... | Topogen | ILA | ONE | REACHABLE | ST/002d237f16f... | ['Vendor': ['Cien... | Ciena | |
| IN/02539c320f9... | ONE | ILA-SD2BRA01-S... | Topogen | ILA | ONE | REACHABLE | ST/02539c320f9... | ['Vendor': ['Cien... | Ciena | |
| IN/027e3d88f5b... | ONE | ILA-SD1PRA01-S... | Topogen | ILA | ONE | REACHABLE | ST/027e3d88f5b... | ['Vendor': ['Cien... | Ciena | |
| IN/04346445503... | ONE | ILA-SD1BOR01-S... | Topogen | ILA | ONE | REACHABLE | ST/04346445503... | ['Vendor': ['Cien... | Ciena | |
| IN/05717daa222... | ONE | ILA-SD1FRA01-S... | Topogen | ILA | ONE | REACHABLE | ST/05717daa222... | ['Vendor': ['Cien... | Ciena | |
| IN/05b5b6c8c0a... | ONE | ILA-SD1PAM01-S... | Topogen | ILA | ONE | REACHABLE | ST/05b5b6c8c0a... | ['Vendor': ['Cien... | Ciena | |
| IN/0727496c71b... | ONE | ILA-SD2PRA01-S... | Topogen | ILA | ONE | REACHABLE | ST/0727496c71b... | ['Vendor': ['Cien... | Ciena | |
| IN/07dcb20e3f7... | ONE | ILA-SD2CAR01-S... | Topogen | ILA | ONE | REACHABLE | ST/07dcb20e3f7... | ['Vendor': ['Cien... | Ciena | |
| IN/082ba1fca29... | ONE | ILA-SD1EV001-S... | Topogen | ILA | ONE | REACHABLE | ST/082ba1fca29... | ['Vendor': ['Cien... | Ciena | |
| IN/0975b2257ba... | ONE | ILA-SD1BER01-S... | Topogen | ILA | ONE | REACHABLE | ST/0975b2257ba... | ['Vendor': ['Cien... | Ciena | |

Figure 6.
Query Screen with Inventory Item Results – Optical Node

SHQL Records fetched at: 12:11:12 06-07-2020

Saved Queries

inventory_item

Run Save Delete Query

RESULTS (546)

Optical Node (463) Router (83)

| Guid | Type | Name | Provider | Extra | ManagementIp | ReachabilityStatus | Site | Tags | Vendor |
|----------------------|--------|----------|----------|-------------------|--------------|--------------------|------------------|------------------------|--------|
| IN/Router/000fc44... | ROUTER | CR1.BEL | Topogen | ['is_core': True] | 10.40.0.7 | REACHABLE | ST/a9bd23d3ed70 | ['Vendor': ['Cisco']] | Cisco |
| IN/Router/03a767f... | ROUTER | ER1.LATH | Topogen | ['is_core': True] | 10.40.0.46 | REACHABLE | ST/7aac1877cd8b | ['Vendor': ['Huawei']] | Huawei |
| IN/Router/0529f67... | ROUTER | CR2.COR | Topogen | ['is_core': True] | 10.40.0.61 | REACHABLE | ST/26a779904306 | ['Vendor': ['Cisco']] | Cisco |
| IN/Router/0640f5f... | ROUTER | CR1.BUD | Topogen | ['is_core': True] | 10.40.0.44 | REACHABLE | ST/5960073736d5 | ['Vendor': ['Cisco']] | Cisco |
| IN/Router/09ac8d... | ROUTER | CR2.LIV | Topogen | ['is_core': True] | 10.40.0.4 | REACHABLE | ST/9354b4a138dd | ['Vendor': ['Cisco']] | Cisco |
| IN/Router/0a1e09e... | ROUTER | CR2.PRA | Topogen | ['is_core': True] | 10.40.0.27 | REACHABLE | ST/6bdd2d5799e | ['Vendor': ['Cisco']] | Cisco |
| IN/Router/0a22a0... | ROUTER | CR1.TLV | Topogen | ['is_core': True] | 10.40.0.67 | REACHABLE | ST/ec8037c4efe39 | ['Vendor': ['Cisco']] | Cisco |
| IN/Router/14bc8d... | ROUTER | CR2.MAD | Topogen | ['is_core': True] | 10.40.0.38 | REACHABLE | ST/d5e070604ed5 | ['Vendor': ['Cisco']] | Cisco |
| IN/Router/1877a9... | ROUTER | ER1.WAR | Topogen | ['is_core': True] | 10.40.0.83 | REACHABLE | ST/ec803a477b10 | ['Vendor': ['Huawei']] | Huawei |
| IN/Router/19694a... | ROUTER | CR1.WAR | Topogen | ['is_core': True] | 10.40.0.28 | REACHABLE | ST/ec803a477b10 | ['Vendor': ['Cisco']] | Cisco |

Figure 7.
Query Screen with Inventory Item Results – Routers

Clicking the ... adjacent to the object in the GUID column displays its properties in JSON format. The .JSON lists the object's properties and their values.

JSON

```
{
  "accessIdentifier": null,
  "children": null,
  "desc": null,
  "deviceFamily": null,
  "deviceType": null,
  "extra": {
    "is_core": true
  },
  "guid": "IN/Router/000fc44c94a1f2cd",
  "managementIp": "10.40.0.7",
  "name": "CR1.BEL",
  "parent": null,
  "provider": "Topogen",
  "reachabilityStatus": "REACHABLE",
  "serialNumber": null,
  "site": {
    "guid": "ST/a9bd23d3ed70"
  },
  "softwareVersion": null,
  "srlgs": null,
  "tags": {
    "Vendor": [
      "Cisco"
    ]
  },
  "topologyId": null,
  "type": "ROUTER",
  "vendor": "Cisco"
}
```

Figure 8.
Inventory Object in .JSON Format

Aborting Queries

When you run a query that takes a long time to return results you can abort the query.

To abort a query:

1. Enter a query command into the **Query** field (see Creating Queries).
2. Click **Run**. If the query will take a long time to return results, an abort option is offered.

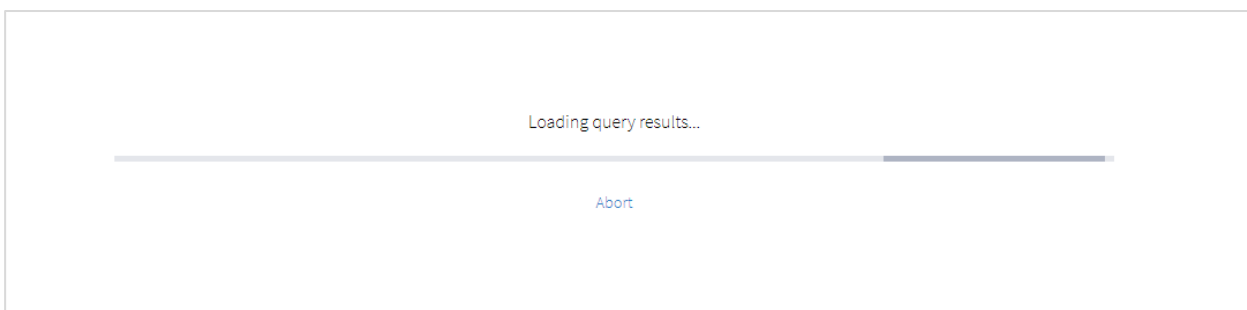


Figure 9.
Abort Query

3. If you want to terminate the query, click **Abort**.

Creating Queries

You can build queries using free text, selecting an item from the auto-completion context menu or using a combination of both. The first item you enter into the **Query** field is an object type.

You can use the auto-completion context menu at any location in the query syntax.

Note: Queries are not case sensitive.

To display an auto-completion context menu:

- Click inside the **Query** field and then do one of the following:
 - **Mac:** Press Command + spacebar
 - **PC:** Press Ctrl + spacebar

The menu appears.

Figure 10 shows the object type auto-completion context menu.

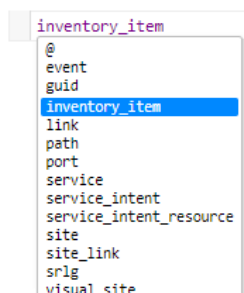



Figure 10.
Auto-completion Context Menu

Selecting an item from the menu displays it in the **Query** field. You can continue to build the query, adding conditions and functions.

Incorrect input - from either free text or context menu selection - is indicated by a red icon . Hovering over the icon indicates why the input is not viable or suggests the expected input.

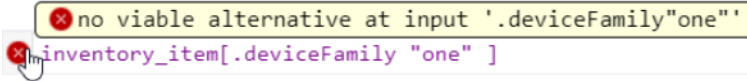


Figure 11.
Input Not Viable

Sometimes, the error is a simple one of syntax, as in Figure 12.

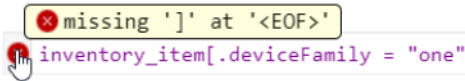


Figure 12.
Syntax Error

Note: It is useful to look at the .JSON file of the queried object type to ascertain what input is viable.

Saving Queries

After creating a query, you can store it in the Saved Queries dropdown menu for repeated use.

To save a query:

1. When you complete writing your query, click **Save As**.

The following dialog box appears.

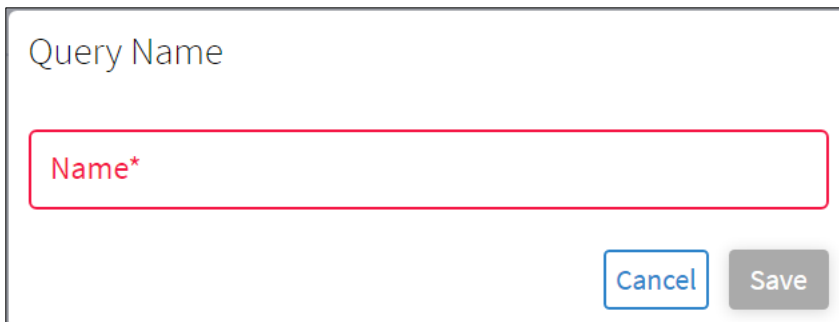


Figure 13.
Query Name Dialog Box

2. Type a name for the query and click **Save**.



Figure 14.
Saved Queries

3. Repeat steps 1 and 2 for every query you want to save.

Deleting Queries

To delete a query:

1. Select a query from the dropdown list of saved queries.

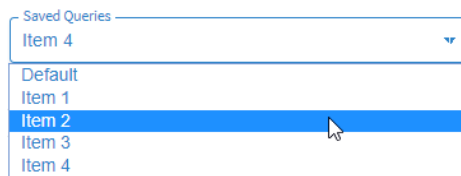


Figure 15.
Deleting Queries

2. Click **Delete**.

The query is removed from the list.

Sorting Columns

Retrieved data is displayed under the relevant column headings.

You can sort the items in ascending or descending order and apply filters to individual columns.

Note: Column management is handled in the same way as other Crosswork Hierarchical Controller tables.

To sort columns:

1. Click a column heading.
An upward pointing arrow appears.

| ↓ Name | ☰ |
|---------|---|
| ER1.WAR | |
| ER1.VAL | |
| ER1.TLV | |
| ER1.STO | |
| ER1.SQY | |
| ER1.PAR | |
| ER1.OVE | |
| ER1.MAN | |
| ER1.MAD | |
| ER1.LIS | |


Figure 16.
Column Contents in Descending Order

2. Click the arrow to sort the column items in ascending order. Click it again to sort the column items in descending order.

| ↓ Name | ☰ |
|---------|---|
| ER1.WAR | |
| ER1.VAL | |
| ER1.TLV | |
| ER1.STO | |
| ER1.SQY | |
| ER1.PAR | |
| ER1.OVE | |
| ER1.MAN | |
| ER1.MAD | |
| ER1.LIS | |

Figure 17.
Column Contents in Ascending Order

To filter column contents:

1. In the column heading, click the  Filter icon.
The column heading expands to display filtering options.

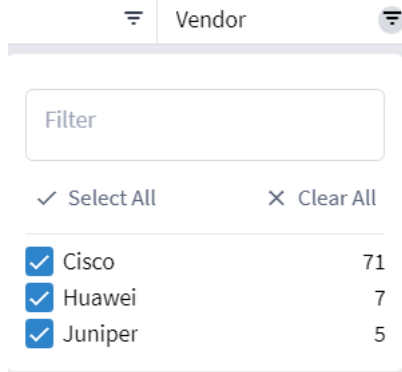



Figure 18.
Filter Column Contents

2. Select an option value or type a pattern into the text box to filter the column contents. The column Filter icon changes to , indicating that a filter is applied to the column.

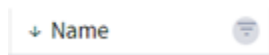


Figure 19.
Filter Applied to Column

Creating Conditions

You can apply conditions to an object type, using the operands described in the table below.

Conditions are generally not case sensitive. Case must be adhered to only when it's part of a predefined list of values.

Conditions are placed within square brackets([]).

Table 4. Operands

| Operand | Numerical | String | Description |
|----------|-----------|--------|--|
| != | ü | ü | Different than. |
| < | ü | | Less than. |
| <= | ü | | Less than or equal to. |
| = | ü | ü | Equal to. |
| > | ü | | Larger than. |
| >= | ü | | Larger than or equal to. |
| contains | | ü | Partial match. |
| endswith | | ü | Ending with a given pattern. |
| has | | | Item in an array. Use to look for an item when the field is a list. |

| Operand | Numerical | String | Description |
|------------|-----------|--------|--|
| in | ü | ü | Matched list of patterns. Use when the field is a single item and the filter contains multiple items. |
| intersect | ü | | Geographical intersection of regions at a specific longitude and latitude. For example: region[.geometry intersect (4.8945398, 52.3666)] |
| is | | ü | Boolean (true / false) and null. |
| not | | ü | Together with is, contains, endswith, startswith, to negate the condition. |
| startswith | | ü | Starting with a given pattern. |
| xin | ü | ü | Matched lists of patterns for multiple properties. |
| ~ | | ü | String with a regular expression. |

Example 1: Filtering with Operands

Figure 20. retrieves a list of routers with names that begin with CR1.

```
inventory_item [.type = "ROUTER" and .name startswith "CR1."]
```

Figure 20.
Filtering Using Operand

Figure 21 shows the results.

SHQL Records fetched at: 11:25:49 06-11-2020 UTC

Saved Queries

`inventory_item [.type = "ROUTER" and .name startswith "CR1."]`

Run Save As Delete Query

RESULTS (31)

Router (31)

| Guid | Type | Name | Provider | Extra | Managementip | ReachabilityStatus | Site | Tags | Vendor |
|--------------------|--------|---------|----------|-------------------|--------------|--------------------|-----------------|-----------------------|--------|
| IN/Router/000fc... | ROUTER | CR1.BEL | Topogen | {'is_core': True} | 10.40.0.7 | REACHABLE | ST/a9bd23d3ed70 | {'Vendor': ['Cisco']} | Cisco |
| IN/Router/0640f... | ROUTER | CR1.BUD | Topogen | {'is_core': True} | 10.40.0.44 | REACHABLE | ST/5960073736d5 | {'Vendor': ['Cisco']} | Cisco |
| IN/Router/0a22a... | ROUTER | CR1.TLV | Topogen | {'is_core': True} | 10.40.0.67 | REACHABLE | ST/c8037c4efe39 | {'Vendor': ['Cisco']} | Cisco |
| IN/Router/19694... | ROUTER | CR1.WAR | Topogen | {'is_core': True} | 10.40.0.28 | REACHABLE | ST/ec803a477b10 | {'Vendor': ['Cisco']} | Cisco |
| IN/Router/1df8a... | ROUTER | CR1.STO | Topogen | {'is_core': True} | 10.40.0.77 | REACHABLE | ST/3bafd5003f87 | {'Vendor': ['Cisco']} | Cisco |
| IN/Router/2023f... | ROUTER | CR1.ATH | Topogen | {'is_core': True} | 10.40.0.47 | REACHABLE | ST/7aac1877cd8b | {'Vendor': ['Cisco']} | Cisco |
| IN/Router/3052d... | ROUTER | CR1.HEL | Topogen | {'is_core': True} | 10.40.0.80 | REACHABLE | ST/2499da6f5e54 | {'Vendor': ['Cisco']} | Cisco |
| IN/Router/318d... | ROUTER | CR1.VAL | Topogen | {'is_core': True} | 10.40.0.63 | REACHABLE | ST/ac13481feb5 | {'Vendor': ['Cisco']} | Cisco |

Figure 21.
Query Results

Example 2: Filtering with Operands

Figure 22 retrieves the inventory where the vendor is Ciena.

```
inventory[.tags.Vendor has ("Ciena")]
```

Figure 22.
Filtering Using Operand

Figure 23 shows the results.

The screenshot shows the SHQL interface with a query and its results. The query is `inventory[.tags.Vendor has ("Ciena")]`. The results are 424 items, all of which are Optical Nodes. The table below shows the first six rows of the results.

| Guid | Type | Name | Provider | DeviceFamily | DeviceType | ReachabilityStat | Site | Tags | Vendor | HasRoadm |
|------------------|------|------------------|----------|--------------|------------|------------------|------------------|---------------------|--------|----------|
| IN/002d237f16... | ONE | ILA-SD1EVO01... | Topogen | ILA | ONE | REACHABLE | ST/002d237f1... | ['Vendor': ['Cie... | Ciena | |
| IN/02539c320f... | ONE | ILA-SD2BRA01... | Topogen | ILA | ONE | REACHABLE | ST/02539c320f... | ['Vendor': ['Cie... | Ciena | |
| IN/027e3d88f5... | ONE | ILA-SD1PRA01... | Topogen | ILA | ONE | REACHABLE | ST/027e3d88f... | ['Vendor': ['Cie... | Ciena | |
| IN/043464455... | ONE | ILA-SD1BOR01... | Topogen | ILA | ONE | REACHABLE | ST/043464455... | ['Vendor': ['Cie... | Ciena | |
| IN/05717daa2... | ONE | ILA-SD1FRA01... | Topogen | ILA | ONE | REACHABLE | ST/05717daa2... | ['Vendor': ['Cie... | Ciena | |
| IN/05b5b6c8c... | ONE | ILA-SD1PAM01... | Topogen | ILA | ONE | REACHABLE | ST/05b5b6c8c... | ['Vendor': ['Cie... | Ciena | |
| IN/0777496c7... | ONF | II A-SD2PRA01... | Tnnsen | II A | ONF | RFACHARI F | ST/0777496c7... | ['Vendor': ['Cie... | Ciena | |

Figure 23.
Query Results

Example 3: Filtering with Operands

Figure 24 looks for links where (port_a.guid = "PO/igp/06c0868cc5601e85/06c0868cc5601e85" and port_b.guid = "PO/igp/c3b0dc2cd3ad6406/c3b0dc2cd3ad6406") or (port_a.guid = "PO/igp/c3b0dc2cd3ad6406/c3b0dc2cd3ad6406" and port_b.guid = "PO/igp/06c0868cc5601e85/06c0868cc5601e85").

```
link[.layer="LSP" and (.portA, .portB) xin (port[.guid = "PO/igp/06c0868cc5601e85/06c0868cc5601e85"], port[.guid = "PO/igp/c3b0dc2cd3ad6406/c3b0dc2cd3ad6406"])]
```

Figure 24.
Filtering Using Operand

Figure 25 shows the results.

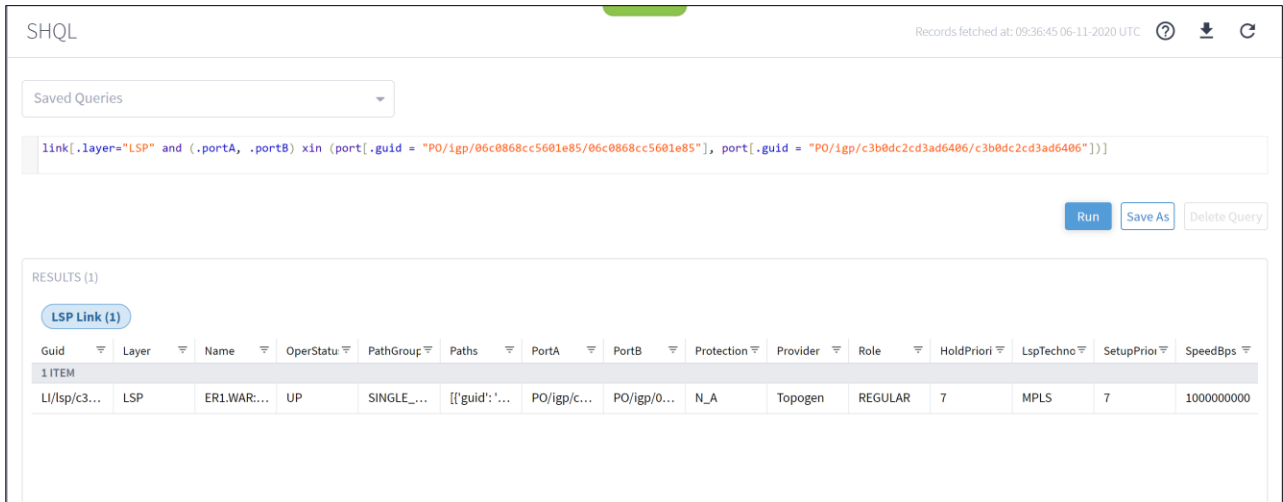


Figure 25.
Query Results

Example 4: Filtering with Operands

Figure 26 retrieves and adds up the events captured the day before yesterday (>-2d and <-1d). The .timeStamp property can be used with d (days), h (hours), m (months), M (minutes), S (seconds), y (years) or w (weeks).

```
event[.timeStamp > -5d and .timeStamp < -1d]
```

Figure 26.
Filtering Using Operand

Figure 27 shows the results.

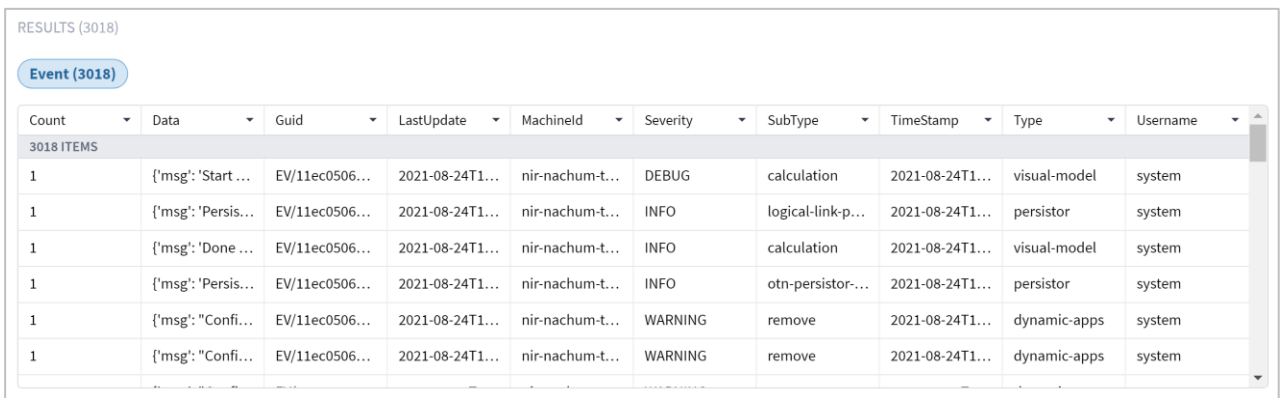


Figure 27.
Query Results

Transforming/Collecting Object Types

You can add an object type to the query command and determine if the data to be retrieved is transformed from one object type to another object type, or if the data reflects a collection of multiple object types and their related items.

- **Transformation:** Add a pipe (|) to the query command before adding the new object type. Transforms the results relating to the previous object type to output for the new object type.
- **Collection:** Add an ampersand (&) to the query command before adding the new object type. Retrieves all the output for all the preceding object types.
- **As:** Add a temporary variable. Enables you to create a query with an object type that is not related to the preceding object type.

The table describes object types and the object types to which they are directly related.

Table 5. Object Types and Properties

| Object type | Object type properties |
|----------------|--|
| event | N/A |
| inventory | inventory_item inventory port site srlg |
| inventory_item | inventory_item inventory port site srlg |
| link | link path port site link srlg region |
| path | path link |
| port | port link inventory_item inventory service srlg |
| region | site visual site link |

| Object type | Object type properties |
|-------------------------|--|
| risk_resource | srlg_risk_resource_mtm risk_resource port link inventory_item inventory |
| service | port service intent |
| service_intent | service_intent service service_intent_resource |
| service_intent_resource | service_intent_resource service_intent |
| site | site inventory item inventory visual site region |
| site_link | site_link visual_site link |
| srlg | srlg port link inventory_item inventory |
| srlg_risk_resource_mtm | srlg_risk_resource_mtm risk_resource port link inventory_item inventory |
| visual_site | visual_site site_link site region |

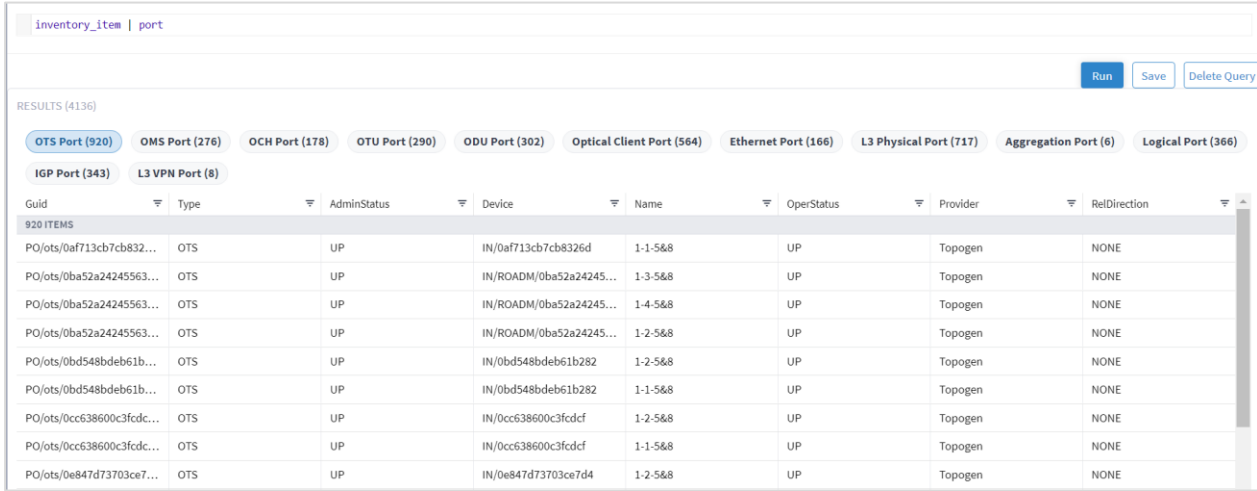
Example: Object Types Transformation

Figure 28. Figure 28 shows a query built to retrieve ports belonging to the object type inventory item. The object types are separated by a pipe (|).

```
inventory_item | port
```

Figure 28.
Query with Object Type Transformation

Figure 29 shows the results.



inventory_item | port

RESULTS (4136)

OTS Port (920) OMS Port (276) OCH Port (178) OTU Port (290) ODU Port (302) Optical Client Port (564) Ethernet Port (166) L3 Physical Port (717) Aggregation Port (6) Logical Port (366)

IGP Port (343) L3 VPN Port (8)

| Guid | Type | AdminStatus | Device | Name | OperStatus | Provider | RelDirection |
|----------------------------|------|-------------|-------------------------|---------|------------|----------|--------------|
| PO/ots/0af713cb7cb8326d... | OTS | UP | IN/0af713cb7cb8326d | 1-1-5&8 | UP | Topogen | NONE |
| PO/ots/0ba52a24245563... | OTS | UP | IN/ROADM/0ba52a24245... | 1-3-5&8 | UP | Topogen | NONE |
| PO/ots/0ba52a24245563... | OTS | UP | IN/ROADM/0ba52a24245... | 1-4-5&8 | UP | Topogen | NONE |
| PO/ots/0ba52a24245563... | OTS | UP | IN/ROADM/0ba52a24245... | 1-2-5&8 | UP | Topogen | NONE |
| PO/ots/0bd548bdeb61b... | OTS | UP | IN/0bd548bdeb61b282 | 1-2-5&8 | UP | Topogen | NONE |
| PO/ots/0bd548bdeb61b... | OTS | UP | IN/0bd548bdeb61b282 | 1-1-5&8 | UP | Topogen | NONE |
| PO/ots/0cc638600c3fcd... | OTS | UP | IN/0cc638600c3fcdcf | 1-2-5&8 | UP | Topogen | NONE |
| PO/ots/0cc638600c3fcd... | OTS | UP | IN/0cc638600c3fcdcf | 1-1-5&8 | UP | Topogen | NONE |
| PO/ots/0e847d73703ce7... | OTS | UP | IN/0e847d73703ce7d4 | 1-2-5&8 | UP | Topogen | NONE |

Figure 29.
Transformation Results

Example: Object Types Collection

Figure 30 shows a query built to retrieve sites and the inventory items the sites belong to. The object types are separated by an ampersand (&).

```
inventory_item & site
```

Figure 30.
Query with Object Type Collection

Figure 31 shows the results.

inventory_item & site

Run Save Delete

RESULTS (976)

Site (430) OneInventoryItem (463) RouterInventoryItem (83)

| Guid | Type | Name | Provider | DeviceFamily | DeviceType | Site | Vendor | HasRoadm |
|---------------------|------|---------------------|----------|--------------|------------|---------------------|--------|----------|
| 463 ITEMS | | | | | | | | |
| IN/002d237f16fb8... | ONE | ILA-SD1EV001-SD1... | Topogen | ILA | ONE | ST/002d237f16fb8... | Ciena | |
| IN/02539c320f9a3dff | ONE | ILA-SD2BRA01-SD2... | Topogen | ILA | ONE | ST/02539c320f9a3dff | Ciena | |
| IN/027e3d88f5b57... | ONE | ILA-SD1PRA01-SD1... | Topogen | ILA | ONE | ST/027e3d88f5b57... | Ciena | |
| IN/043464455031f... | ONE | ILA-SD1BOR01-SD... | Topogen | ILA | ONE | ST/043464455031f... | Ciena | |
| IN/05717daa222ea... | ONE | ILA-SD1FRA01-SD1... | Topogen | ILA | ONE | ST/05717daa222ea... | Ciena | |
| IN/05b5b6c8c0a02... | ONE | ILA-SD1PAM01-SD... | Topogen | ILA | ONE | ST/05b5b6c8c0a02... | Ciena | |
| IN/0727496c71b88... | ONE | ILA-SD2PRA01-SD2... | Topogen | ILA | ONE | ST/0727496c71b88... | Ciena | |
| IN/07dcb20e3f79b... | ONE | ILA-SD2CAR01-SD2... | Topogen | ILA | ONE | ST/07dcb20e3f79b... | Ciena | |
| IN/082ba1fca299661 | ONE | ILA-SD1EV001-SD1... | Topogen | ILA | ONE | ST/082ba1fca2996... | Ciena | |
| IN/0975b2257baf3... | ONE | ILA-SD1BER01-SD1... | Topogen | ILA | ONE | ST/0975b2257baf3... | Ciena | |

Figure 31.
Collection Results

Example: As Temporary Variable

Using **As** allows you to use the output of a query as the input in collections.

The following figures demonstrate how to create a query built to support non-related object types using a temporary variable.

Figure 32 shows a query command composed of an object type, link.

```
link[.layer = "R_LOGICAL"]
```

Figure 32.
Query

Figure 33 shows the results.

link[.layer = "R_LOGICAL"]

Run Save Delete

RESULTS (118)

RLogicalLink (118)

| Guid | Layer | Name | OperStatus | PathGroupType | Paths | PortA | PortB | ProtectionStatus | Provider | Role |
|-------------------|-----------|-------------------|------------|---------------|---------------------|------------------|-------------------|------------------|----------|---------|
| Li/v_logical/0... | R_LOGICAL | 10.40.0.45 to ... | UP | SINGLE_PATH | [{"guid": "PA/Li... | PO/v_logical/... | PO/v_logical/f... | N_A | Topogen | REGULAR |
| Li/v_logical/0... | R_LOGICAL | 10.40.0.41 to ... | UP | SINGLE_PATH | [{"guid": "PA/Li... | PO/v_logical/... | PO/v_logical/f... | N_A | Topogen | REGULAR |
| Li/v_logical/0... | R_LOGICAL | 10.40.0.34 to ... | UP | SINGLE_PATH | [{"guid": "PA/Li... | PO/v_logical/... | PO/v_logical/f... | N_A | Topogen | REGULAR |
| Li/v_logical/0... | R_LOGICAL | 10.40.1.118 to... | UP | SINGLE_PATH | [{"guid": "PA/Li... | PO/v_logical/... | PO/v_logical/f... | N_A | Topogen | REGULAR |
| Li/v_logical/0... | R_LOGICAL | 10.40.1.35 to ... | UP | SINGLE_PATH | [{"guid": "PA/Li... | PO/v_logical/... | PO/v_logical/f... | N_A | Topogen | REGULAR |
| Li/v_logical/0... | R_LOGICAL | 10.40.1.41 to ... | UP | SINGLE_PATH | [{"guid": "PA/Li... | PO/v_logical/... | PO/v_logical/f... | N_A | Topogen | REGULAR |
| Li/v_logical/0... | R_LOGICAL | 10.40.1.34 to ... | UP | SINGLE_PATH | [{"guid": "PA/Li... | PO/v_logical/... | PO/v_logical/f... | N_A | Topogen | REGULAR |
| Li/v_logical/0... | R_LOGICAL | 10.40.1.153 to... | UP | SINGLE_PATH | [{"guid": "PA/Li... | PO/v_logical/... | PO/v_logical/f... | N_A | Topogen | REGULAR |
| Li/v_logical/0... | R_LOGICAL | 10.40.1.146 to... | UP | SINGLE_PATH | [{"guid": "PA/Li... | PO/v_logical/... | PO/v_logical/f... | N_A | Topogen | REGULAR |
| Li/v_logical/0... | R_LOGICAL | 10.40.1.62 to ... | UP | SINGLE_PATH | [{"guid": "PA/Li... | PO/v_logical/... | PO/v_logical/f... | N_A | Topogen | REGULAR |

Figure 33.
Query Results

Figure 34 shows how to create a temporary variable using **As**. The variable L appears in the context menu as a valid object type.

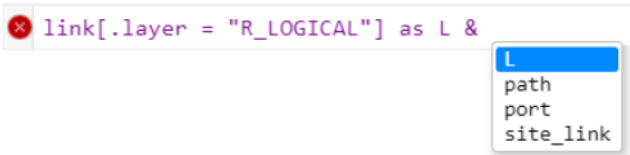


Figure 34.
Adding As

Figure 35 shows how to use the temporary variable L to transform a list of links to their ports, paths and site links.

```
link[.layer = "R_LOGICAL"] as L & L | port & L | path & L | site_link
```

Figure 35.
Query Using as

Figure 36 shows the results.


```
link[.layer = "R_LOGICAL"] as L & L | port & L | path & L |site_link
```

RESULTS (642)

Site Link (170) Logical Port (236) Logical Link (118) Path (118)

| Guid | Depth | Layer | Links | Name | SiteA | SiteB | Status | Tags |
|------------------------|-------|-----------|-----------------------------|-----------|------------------------|-----------------------|--------|-----------------------|
| 170 ITEMS | | | | | | | | |
| SL/R_LOGICAL/VS/596... | 3 | R_LOGICAL | [{"guid": "L/r_logical/0... | BUD - BUC | VS/5960073736d5_0_1... | VS/a1c737928f90_0_1_2 | NORMAL | ['Vendor': ['Cisco']] |
| SL/R_LOGICAL/VS/839... | 2 | R_LOGICAL | [{"guid": "L/r_logical/c... | OVE - MAD | VS/839c2694c73e_0_1 | VS/d5e070604ed5_0_1 | NORMAL | ['Vendor': ['Cisco']] |
| SL/R_LOGICAL/VS/ba7... | 1 | R_LOGICAL | [{"guid": "L/r_logical/c... | BCN - MAD | VS/ba7d24f4bc71_0 | VS/d5e070604ed5_0 | NORMAL | ['Vendor': ['Cisco']] |
| SL/R_LOGICAL/VS/3ba... | 1 | R_LOGICAL | [{"guid": "L/r_logical/c... | STO - COP | VS/3bafd5003f87 | VS/df8413a1d7a4 | NORMAL | ['Vendor': ['Cisco']] |
| SL/R_LOGICAL/VS/7aa... | 3 | R_LOGICAL | [{"guid": "L/r_logical/e... | ATH - BUC | VS/7aac1877cd8b_0_1_2 | VS/a1c737928f90_0_1_2 | NORMAL | ['Vendor': ['Cisco']] |
| SL/R_LOGICAL/VS/26a... | 1 | R_LOGICAL | [{"guid": "L/r_logical/6... | COR - VAL | VS/26a779904306_0 | VS/ac13481febe5_0 | NORMAL | ['Vendor': ['Cisco']] |
| SL/R_LOGICAL/VS/d5e... | 1 | R_LOGICAL | [{"guid": "L/r_logical/5... | MAD - PAR | VS/d5e070604ed5 | VS/e7df76d7a9cb | NORMAL | ['Vendor': ['Cisco']] |
| SL/R_LOGICAL/VS/56c... | 2 | R_LOGICAL | [{"guid": "L/r_logical/0... | MAN - BEL | VS/56ceb1686268_0_1 | VS/a9bd23d3ed70_0_1 | NORMAL | ['Vendor': ['Cisco']] |
| SL/R_LOGICAL/VS/297... | 1 | R_LOGICAL | [{"guid": "L/r_logical/8... | BKL - MAN | VS/2971737bd3ba_0 | VS/56ceb1686268_0 | DOWN | ['Vendor': ['Cisco']] |
| SL/R_LOGICAL/VS/d11... | 3 | R_LOGICAL | [{"guid": "L/r_logical/d... | LIS - MAD | VS/d11c3fd70e8a_0_1_2 | VS/d5e070604ed5_0_... | NORMAL | ['Vendor': ['Cisco']] |

Figure 36. Results using Temporary Variable

Functions

SHQL provides several functions that allow you to precisely define the quantity, type and network layer of the items to retrieve.

Functions are preceded by a pipe in the query command line.

You can retrieve an item and then specify whether to retrieve related items from either above or below the layer, or from both above and below. These recursive operations are valid for port, link, site, inventory, and visual site.

Using Retrospective plus a timestamp enables you to query a situation that existed before the present time. For example, you can check the status of a site when it was down earlier in the day but in the present is up and running.

In addition, SHQL functions enable you to retrieve data based on free text.

SHQL functions are described in the table below.

Table 6. SHQL Functions

| Function | Description |
|------------------|--|
| downward | Retrieves items from below the layer of the specified item |
| upward | Retrieves items from above the layer of the specified item |
| span | Retrieves items from below and above the layer of the specified item |
| fts | Free text search. Retrieves items according to the search string you enter. |
| group_by | Returns query results grouped by property, for example, how many ports have an OperStatus of Up. |
| retrospective(@) | Retrieves items from the past according to a given timestamp. |

Example: downward Function

The following figures demonstrate how to create a query to retrieve items from all layers below a link, according to its GUID.

Figure 37 shows a query command composed of an object type followed by its GUID.

```
link[ .guid = "LI/r_logical/000fc44c94a1f2cd/51308dfd752c1574/fe5f5fda14fc609e/a3b059003ccd1b4d" ]
```

Figure 37.

Query Command

Figure 38 shows the results.

The screenshot shows a query command in a text input field: `link[.guid = "LI/r_logical/000fc44c94a1f2cd/51308dfd752c1574/fe5f5fda14fc609e/a3b059003ccd1b4d"]`. Below the input are buttons for "Run", "Save As", and "Delete". The results section shows "RESULTS (1)" with a filter for "Logical Link (1)". A table displays the following data:

| Guid | Layer | Name | OperStatus | PathGroupType | Paths | PortA | PortB | ProtectionStat | Provider | Role |
|--|-----------|-------------------|------------|---------------|---------------------|------------------|------------------|----------------|----------|---------|
| LI/r_logical/000fc44c94a1f2cd/51308dfd752c1574/fe5f5fda14fc609e/a3b059003ccd1b4d | R_LOGICAL | 10.40.0.45 to ... | UP | SINGLE_PATH | [[{"guid": "PA/L... | PO/r_logical/... | PO/r_logical/... | N_A | Topogen | REGULAR |

Figure 38.

GUID Results

Figure 39 shows nine additional items below the link layer, retrieved after adding the function downward to the command.

The screenshot shows the same query command as Figure 37, but with the word "downward" added at the end: `link[.guid = "LI/r_logical/000fc44c94a1f2cd/51308dfd752c1574/fe5f5fda14fc609e/a3b059003ccd1b4d"] | downward`. The results section shows "RESULTS (79)" with filters for "Fiber Link (1)", "Fiber Segment Link (55)", "OTS Link (4)", "OMS Link (4)", "OCH Link (2)", "OTU Link (2)", "ODU Link (4)", "Ethernet Link (5)", "L3 Physical Link (1)", and "Logical Link (1)". A table displays the following data:

| Guid | Layer | DistanceMeters | Name | OperStatus | PathGroupType | Paths | ProtectionStat | Provider | Role |
|---------------------------|-------|----------------|-----------------|------------|---------------|----------------------|----------------|----------|---------|
| LI/fiber/9c095046015f020c | FIBER | 73673 | SD1BLA02/1-3... | UP | SINGLE_PATH | [[{"guid": "PA/LI... | N_A | Topogen | REGULAR |

Figure 39.

Results with the Function Downward

Example: retrospective Function

Figure 40 shows the query command built to retrieve the situation in the network at a specific time in the past.

You can use different time formats:

- Absolute time: @2019-05-10 10:00:00
- Relative time in the format '-'[0-9]+'[ymwdHMS]: @-10H
- Unix timestamp (ms): @1558610956000

```
@1560680169000 site
```

Figure 40.
@ and Epoch Timestamp

Figure 41 shows the results.

| GUID | Latitude | Longitude | Name | Parent |
|---------------------|------------|------------|--------------------------|----------------|
| ST/9a52e7b877293e29 | 51.7320209 | -1.2977263 | GDF | ST/4c37f0a8d98 |
| ST/9707dda29500ee09 | 44.3877201 | 0.0852114 | ILA-SD1BOR01-SD1MAR01-0 | |
| ST/e1c348d653e79458 | 52.5994966 | 21.4467033 | ILA-SD2MIN01-SD2WAR01-5 | |
| ST/aace62ec793f474a | 47.3768866 | 8.541994 | ZUR | ST/172e92f2e5f |
| ST/59becd53818200b0 | 45.8282135 | 5.0306429 | ILA-SD1LV001-SD1MAR01-2 | |
| ST/44352a2daefd7f53 | 55.2680552 | 14.8033629 | ILA-SD2MIN01-SD2MOS01-5 | |
| ST/9a52e7b877293e29 | 42.5047920 | 27.4628261 | [BUR] | |
| ST/9707dda29500ee09 | 41.9780289 | 2.157833 | ILA-SD1BCN01-SD1GIR01-0 | |
| ST/e1c348d653e79458 | 49.4824699 | 19.0547261 | ILA-SD2BRAT01-SD2KRA01-2 | |
| ST/aace62ec793f474a | 39.4149282 | 22.4384597 | ILA-SD2LAM01-SD2LAR02-0 | |

Figure 41.
Timestamp Results

Example: group_by Function

Figure 42 shows the query command to group the devices by site (and count them).

```
port |group_by(.device.site)|asc(.count)
```

Figure 42.
group_by

Figure 23 shows the results.

| Device.Site | Count |
|---------------------|-------|
| ST/9a52e7b877293e29 | 2 |
| ST/9707dda29500ee09 | 2 |
| ST/e1c348d653e79458 | 2 |
| ST/aace62ec793f474a | 2 |
| ST/59becd53818200b0 | 2 |
| ST/44352a2daefd7f53 | 2 |

Figure 43.
group_by Results

History Function

Crosswork Hierarchical Controller keeps records of all changes in the network inventory and topology. The changes are stored as a list of events in an event history table. An event is a record of any resource addition (ADD), deletion (DELETE) or attribute change (UPDATE). The events are linked to the model using the guid as an index to the objects in the model. The events table includes all the details of the changes including the time, resource ID, object type, event type, and the value before and after the change.

A unique implementation was selected in order to include all resources that were in the model at some point in the selected time range in the returned results. Thus, it guarantees that services or links that were subsequently deleted are still found. For example, if you query the table to get events for all services with specific tags for the past two weeks, the results will include the relevant services, including those that no longer exist at the end of the period. For more information, refer to the *Crosswork Hierarchical Controller Network History Guide*.

You can construct a query that uses a standard SHQL query to filter the model, then add the pipe (|) and filter the history table. The history table can be filtered by all the entry attributes:

- Changed object guid
- Changed object type
- Change type (ADD, UPDATE, DELETE)
- Value before
- Value after

The results returned are from the event history table (whereas regular SHQL queries return results from the objects table).

Example: history

Figure 44 shows the query command built to retrieve the events for links between two dates. You can narrow the query results by adding in the history table options, for example, only show UPDATE events, or search for links that went down.

```
@-18d:-17d link | history[.action="UPDATE"]
```

```
@-21d:-13d link | history[.to.operStatus = "DOWN"]
```

Figure 44.

@ and history

Figure 45 shows the results.

| Action | Changes | ObjGuid | ObjType | Timestamp |
|--------|---------|--|---------|---------------|
| INSERT | | LI/r_logical/f38377d02bd88162/69f91c6c31f4ce6e/5d41... | link | 1612994572113 |
| INSERT | | LI/fiber_segment/f158ea32d21c2f1d | link | 1612994572113 |
| INSERT | | LI/r_logical/5ed7163adf3bd5e/7d92f458f24bfa3/5d41... | link | 1612994572113 |
| INSERT | | LI/r_logical/318dds45085bb78/69f91c6c31f4ce6e/3460... | link | 1612994572113 |
| INSERT | | LI/fiber_segment/d5ce8140095d788f2 | link | 1612994572113 |
| INSERT | | LI/r_logical/3fe1f91b97e054c1/51308df752c1574/09ac... | link | 1612994572113 |
| INSERT | | LI/fiber_segment/4e58ae4e65eb6e26 | link | 1612994572113 |
| INSERT | | LI/fiber_segment/ebc10f48d59c843e | link | 1612994572113 |
| INSERT | | LI/fiber_segment/3f493e21cf7ceb90 | link | 1612994572113 |
| INSERT | | LI/fiber_segment/c06826a9b8bd8f5 | link | 1612994572113 |
| INSERT | | LI/fiber_segment/24c685e0fe2a140a | link | 1612994572113 |
| INSERT | | LI/fiber_segment/09678bf7b96050d7 | link | 1612994572113 |
| INSERT | | LI/fiber_segment/21dacb7f325008be | link | 1612994572113 |
| INSERT | | LI/fiber_segment/a7c4c8814fd443ef | link | 1612994572113 |
| INSERT | | LI/fiber_segment/3cd11fb650f541a | link | 1612994572113 |
| INSERT | | LI/fiber_segment/99a33043b4518d66 | link | 1612994572113 |
| INSERT | | LI/fiber_segment/cbb0a28a1b9c8e1 | link | 1612994572113 |
| INSERT | | LI/fiber_segment/f51hr7>R0A4ad770 | link | 1612994572113 |

Figure 45.
History Results

Additional Output Functions

You can add the functions described in the table below to retrieve results and display them in a specific order. Typically, these functions are added at the end of the query command. You can also view specific properties for the query results.

The function order is significant and must appear in the following order:

- fibre talk set > view > operands > asc/desc > limit > after

For example:

- This query is invalid: port | limit(10) | asc(.name)
- This query is valid: port | asc(.name) | limit(10)

Output functions are preceded by a pipe (|).

Table 7. Output Functions

| Function | Description |
|---------------|---|
| add_counters | Displays the total number per attribute value for the specified object type. |
| after (GUID) | Displays only the results that follow the item with the specified GUID. |
| asc (column) | Displays results in ascending natural order |
| desc (column) | Displays results in descending natural order |
| group_by | |
| limit(#) | Limits the number of displayed results |
| view | Displays the specified properties (with the labels provided) for the query results. |

Example: after

Figure 46 shows a query built to retrieve only items that follow the item with the specified GUID.

```
inventory_item | after("IN/Router/0529f67055b6efe0")
```

Figure 46.
After Output Function

Figure 47 shows the results.

| Guid | Type | Name | Provider | Extra | ManagementIp | ReachabilitySta | Site | Tags | Vendor |
|----------------------------|--------|---------|----------|--------------------|--------------|-----------------|-----------------|--------------------|--------|
| Router (80) | | | | | | | | | |
| 80 ITEMS | | | | | | | | | |
| IN/Router/0a22a0ec157648e1 | ROUTER | CR1.TLV | Topogen | {'is_core': True} | 10.40.0.67 | REACHABLE | ST/c8037c4ef... | ['Vendor': {'Ci... | Cisco |
| IN/Router/14bc8dcea3a57f18 | ROUTER | CR2.MAD | Topogen | {'is_core': True} | 10.40.0.38 | REACHABLE | ST/d5e070604... | ['Vendor': {'Ci... | Cisco |
| IN/Router/1877a9950429216f | ROUTER | ER1.WAR | Topogen | {'is_core': True} | 10.40.0.83 | REACHABLE | ST/ec803a477... | ['Vendor': {'H... | Huawei |
| IN/Router/19694a81a1d0ebed | ROUTER | CR1.WAR | Topogen | {'is_core': True} | 10.40.0.28 | REACHABLE | ST/ec803a477... | ['Vendor': {'Ci... | Cisco |
| IN/Router/1c8ae4011529613d | ROUTER | ER1.BCN | Topogen | {'is_core': False} | 10.40.0.51 | REACHABLE | ST/ba7d24f4b... | ['Vendor': {'Ci... | Cisco |
| IN/Router/1df8ab6998bf7fc8 | ROUTER | CR1.STO | Topogen | {'is_core': True} | 10.40.0.77 | REACHABLE | ST/3bafd5003... | ['Vendor': {'Ci... | Cisco |
| IN/Router/2023fd21fb812871 | ROUTER | CR1.ATH | Topogen | {'is_core': True} | 10.40.0.47 | REACHABLE | ST/7aac1877c... | ['Vendor': {'Ci... | Cisco |
| IN/Router/2f29e956db0f0cae | ROUTER | ER1.VAL | Topogen | {'is_core': False} | 10.40.0.62 | REACHABLE | ST/ac13481fe... | ['Vendor': {'Ci... | Cisco |
| IN/Router/3052d71050f39870 | ROUTER | CR1.HEL | Topogen | {'is_core': True} | 10.40.0.80 | REACHABLE | ST/2499da6f5... | ['Vendor': {'Ci... | Cisco |
| IN/Router/318dd5a50856bb78 | ROUTER | CR1.VAL | Topogen | {'is_core': True} | 10.40.0.63 | REACHABLE | ST/ac13481fe... | ['Vendor': {'Ci... | Cisco |

Figure 47.
After Output Results

Example 1: add_counters

Figure 48 shows a query built to retrieve the total number of each of the link layers. Adding `| limit (0)` to the query limits the response to the counters only.

```
link | add_counters (.layer) | limit (0)
```

Figure 48.
add_counters Function

Figure 49 shows the results.

| Attribute Name | Attribute Value | Counter |
|-------------------|-----------------|---------|
| RESULTS (0) | | |
| ShiqCounters (14) | | |
| 14 ITEMS | | |
| layer | ODU | 151 |
| layer | FIBER_SEGMENT | 19380 |
| layer | LSP | 1656 |
| layer | OTS | 460 |
| layer | FIBER | 432 |
| layer | OTU | 145 |
| layer | R_LOGICAL | 233 |
| layer | IGP | 118 |
| layer | OHS | 138 |
| layer | OCH | 89 |
| layer | L3_VPN | 8 |
| layer | ETH | 175 |
| layer | R_AGGREGATE | 3 |
| layer | R_PHYSICAL | 124 |

Figure 49.
add_counters Results

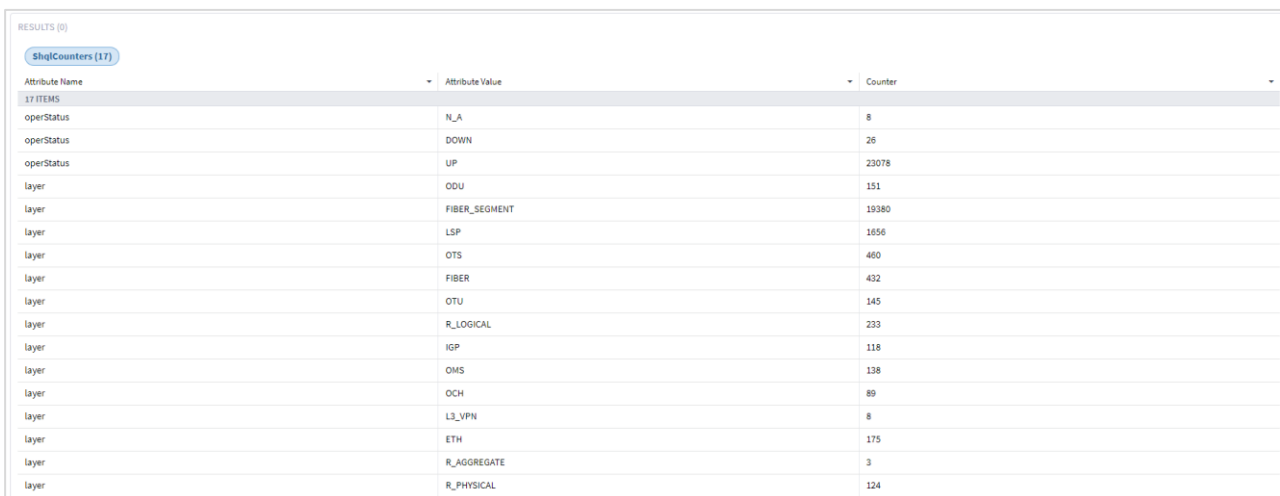
Example 2: add_counters

Figure 50 shows a query built to retrieve the total number of each of the link layers and the total number of links in each of the operational statuses (UP, DOWN or N_A). Adding **| limit (0)** to the query limits the response to the counters only.

```
link | add_counters (.operStatus, .layer) | limit (0)
```

Figure 50.
add_counters Function

Figure 51 shows the results.



| Attribute Name | Attribute Value | Counter |
|----------------|-----------------|---------|
| 17 ITEMS | | |
| operStatus | N_A | 8 |
| operStatus | DOWN | 26 |
| operStatus | UP | 23078 |
| layer | ODU | 151 |
| layer | FIBER_SEGMENT | 19380 |
| layer | LSP | 1656 |
| layer | OTS | 460 |
| layer | FIBER | 432 |
| layer | OTU | 145 |
| layer | R_LOGICAL | 233 |
| layer | IGP | 118 |
| layer | OMS | 138 |
| layer | OCH | 89 |
| layer | L3_VPN | 8 |
| layer | ETH | 175 |
| layer | R_AGGREGATE | 3 |
| layer | R_PHYSICAL | 124 |

Figure 51.
add_counters Results

Example 1: view

Figure 52 shows a query built to retrieve the port guid, name, device.name and teMetric for IGP ports.

```
port[.type="IGP"]|view ("guid":.guid, "name":.name, "device":.device.name, "te_metric":.teMetric)
```

Figure 52.
View Output Function

Figure 53 shows the results.

RESULTS (347)

[View \(347\)](#)

| Guid | Name | Device | Te_metric |
|---|-------------|---------|-----------|
| 347 ITEMS | | | |
| PO/igp/05e94d0bf33ab764/05e94d0bf33ab... | 10.40.0.30 | CR2.COP | 0 |
| PO/igp/05e94d0bf33ab764/f54e0329eb3d8... | 10.40.0.166 | CR2.COP | 40682 |
| PO/igp/05e94d0bf33ab764/fe3d98f432600bd1 | 10.40.0.189 | CR2.COP | 0 |
| PO/igp/06c0868cc5601e85/06c0868cc5601e85 | 10.40.0.21 | ER1.PAR | 0 |
| PO/igp/06c0868cc5601e85/2bba1a15c5d0800f | 10.40.0.106 | ER1.PAR | 0 |
| PO/igp/06c0868cc5601e85/e7909350ec27b7... | 10.40.0.137 | ER1.PAR | 0 |

Figure 53.
View Output Results

Example 2: view

Figure 54 shows a query built to retrieve the link guid and the number of paths in the link, when the link is not in a FIBER or FIBER_SEGMENT layer.

```
link[.layer not in ("FIBER", "FIBER_SEGMENT")] | view ("guid": .guid, "number": count(.paths))
```

Figure 54.
View Output Function

Figure 55 shows the results.

RESULTS (3185)

[View \(3185\)](#)

| Guid | Number |
|--|--------|
| 3185 ITEMS | |
| LI/eth/000fc44c94a1f2cd/51308dfd752c1574/df753d953c1e1c8f/f8e7b20537ce03b7 | 0 |
| LI/eth/018c725e75339c6c/1ab9838586d6af43/3ce85abb6ad75831/467d67a8221dfca2 | 1 |
| LI/eth/0640f5f2a4a4a5b2/51308dfd752c1574/3ce85abb6ad75831/467d67a8221dfca2 | 0 |
| LI/eth/0640f5f2a4a4a5b2/ad2095e495c117ce/3ce85abb6ad75831/45fbcf353eff8146 | 0 |
| LI/eth/08a4ce19ff3c89ab/efb8a463377896b8/ab34b1bcb5abd8cd/62cc89608ea94c16 | 0 |
| LI/eth/08a4ce19ff3c89ab/f8e7b20537ce03b7/ab34b1bcb5abd8cd/efb8a463377896b8 | 0 |

Figure 55.
View Output Results

Example 3: view

Figure 56 shows a query built to retrieve the link guid and port names when the link is not in a FIBER or FIBER_SEGMENT layer. The output is limited to 20 entries.

```
link[.layer not in ("FIBER", "FIBER_SEGMENT")] | view("g": .guid, "n-a": .portA.name, "n-b": .portB.name)|limit(20)
```

Figure 56.
View Output Function

Figure 57 shows the results.

RESULTS (20)

[View \(20\)](#)

| G | N-A | N-B |
|---|--------------------|--------------------|
| 20 ITEMS | | |
| LI/r_logical/f38377d02bd88162/69f91c6c31f4ce6e/5d4172c3965d428... | HundredGigE0/1/0/0 | HundredGigE0/1/0/0 |
| LI/r_logical/5ed7163adf3b8d5e/7d82f458ff24bfa3/5d4172c3965d428... | TenGigE0/0/0/8 | TenGigE0/0/0/4 |
| LI/r_logical/318dd5a50856bb78/69f91c6c31f4ce6e/3460e659c02aefc... | HundredGigE0/1/0/0 | HundredGigE0/1/0/0 |
| LI/r_logical/3fe1f91b97e054c1/51308dfd752c1574/09ac8d52042280e... | TenGigE0/0/0/5 | TenGigE0/0/0/4 |
| LI/r_logical/94b82c614c0402dc/51308dfd752c1574/f3b3b56ab45fd1... | TenGigE0/0/0/5 | TenGigE0/0/0/4 |
| LI/r_logical/050148266f11f3/c6ab5f4e45ef0e0/f602dfb1373f394c/1ab... | ODU100E | ODU100E |

Figure 57.
View Output Results

Example 4: view

Figure 58 shows a query built to count the hops for the specified path.

```
path[.guid="PA/LI/fiber/0d9fe6e5f9bad3c5"] | view("g": .guid, "hops": count(.hops))
```

Figure 58.
View Output Function

Figure 59 shows the results.

RESULTS (1)

[View \(1\)](#)

| G | Hops |
|------------------------------|------|
| 1 ITEM | |
| PA/LI/fiber/0d9fe6e5f9bad3c5 | 57 |

Figure 59.
View Output Results

Complex Examples

Complex Example 1

The following figures demonstrate how to get OMS links that causes LSPs to fail.

The query command includes the following:

```
site[.name="DUB" ] | inventory_item | port | link[.layer=" LSP" ] as w; site[.name="BEL" ] | inventory_item | port | link[.layer=" LSP" and .guid in w ] | downward(" OMS" ) | link[.layer=" OMS" and .operStatus=" DOWN" ]
```

Get all LSPs between sites DUB and BEL:

- Use | to transform from one object type to another.
- Use alias and in to reduce the list to only those between DUB to BEL.

Down to OMS links and filter by operational state:

- Use ‘downward’ function.

- Use layer name in function (downward (“OMS”)) to limit the downward to a specific level.

```
site[.name="DUB"] | inventory_item | port | link[.layer="LSP"] as w; site[.name="BEL"] | inventory_item | port | link[.layer="LSP" and .guid in w] | downward("OMS") | link[.layer="OMS" and .operStatus="DOWN"]
```

Figure 60.
Query Results

| Guid | Layer | Name | OperStatus | PathGroupType | Paths | PortA | PortB | ProtectionStatus | Provider | Role |
|-------------------|-------|-------------------|------------|---------------|---|-------|-------|------------------|----------|---------|
| LI/oms/af5e85f... | OMS | SD1BKL01/1-2-5... | DOWN | SINGLE_PATH | [{"guid": "PA/oms...", "port": "PO/oms/af5e85f...", "port_b": "PO/oms/9bf4b7..."} | | | N_A | Topogen | REGULAR |

Figure 61.
Query Results

Complex Example 2

The following figures demonstrate how to get the current list of OMSs.

The query command includes the following:

```
link[.name="CR1.DUB:CR1.BEL:lsp_0"] | downward | link[.layer="OMS"]
```

Get all OMSs used by specific LSP:

- Get LSP by its name.
- Downward to OMSs used by this LSP.

The query command that includes the following:

```
| link[.name="CR1.DUB:CR1.BEL:lsp_0"] | downward | link[.layer="OMS"]
```

Figure 62.
Query Command

SHQL Records fetched at: 19:54:47 06-12-2020 UTC

Saved Queries

```
link[.name="CR1.DUB:CR1.BEL:lsp_0"] | downward | link[.layer="OMS"]
```

Run Save As Delete Query

RESULTS (12)

OMS Link (12)

| Guid | Layer | Name | OperStatus | PathGroupType | Paths | PortA | PortB | ProtectionStatus | Provider | Role |
|--------------------|-------|-------------------|------------|---------------|---------------------|------------------|------------------|------------------|----------|---------|
| LI/oms/4afa911... | OMS | SD1DUB01/1-2-... | UP | SINGLE_PATH | [{"guid": "PA/om... | PO/oms/4afa91... | PO/oms/08a4ce... | N_A | Topogen | REGULAR |
| LI/oms/7c54f85... | OMS | SD1LIV01/1-2-5... | UP | SINGLE_PATH | [{"guid": "PA/om... | PO/oms/7c54f8... | PO/oms/ab34b... | N_A | Topogen | REGULAR |
| LI/oms/af5e85ff... | OMS | SD1BKL01/1-2-... | DOWN | SINGLE_PATH | [{"guid": "PA/om... | PO/oms/af5e85... | PO/oms/9bf4b7... | N_A | Topogen | REGULAR |
| LI/oms/aaba827... | OMS | SD1OXF01/1-3-... | UP | SINGLE_PATH | [{"guid": "PA/om... | PO/oms/aaba82... | PO/oms/9bf4b7... | N_A | Topogen | REGULAR |
| LI/oms/9c601df... | OMS | SD1BIR01/1-4-5... | UP | SINGLE_PATH | [{"guid": "PA/om... | PO/oms/9c601d... | PO/oms/aaba82... | N_A | Topogen | REGULAR |
| LI/oms/9c601df... | OMS | SD1BIR01/1-2-5... | UP | SINGLE_PATH | [{"guid": "PA/om... | PO/oms/9c601d... | PO/oms/7c54f8... | N_A | Topogen | REGULAR |
| LI/oms/9c601df... | OMS | SD1BIR01/1-3-5... | UP | SINGLE_PATH | [{"guid": "PA/om... | PO/oms/9c601d... | PO/oms/394fa7... | N_A | Topogen | REGULAR |
| LI/oms/644a90f... | OMS | SD1MAN01/1-3-... | UP | SINGLE_PATH | [{"guid": "PA/om... | PO/oms/644a90... | PO/oms/394fa7... | N_A | Topogen | REGULAR |
| LI/oms/df753d9... | OMS | SD1BEL01/1-2-5... | UP | SINGLE_PATH | [{"guid": "PA/om... | PO/oms/df753d... | PO/oms/ae10e0... | N_A | Topogen | REGULAR |
| LI/oms/9e79f8d... | OMS | SD1IOM101/1-3... | UP | SINGLE_PATH | [{"guid": "PA/om... | PO/oms/9e79f8... | PO/oms/ae10e0... | N_A | Topogen | REGULAR |

Figure 63.
Query Results

If you want to track the LSP path change over time, you can use for example, @-1d:

```
@-1d link[.name="CR1.DUB:CR1.BEL:lsp_0"] | downward | link[.layer="OMS"]
```

Figure 64.
Query Command

Complex Example 3

The following figures demonstrate how to find shared risk between two IP links.

The query command includes the following:

```
link[.name="10.40.0.157 to 10.40.0.158"] | downward | link[.layer="OMS"] as p;
link[.name="10.40.0.154 to 10.40.0.153"] | downward | link[.layer="OMS"] as w; link[.guid in p and .guid in w]
```

Get OMSs of the first IP logical link:

- Filter by link name, downward to its OMSs, and alias it as p.

Get OMSs of the second IP logical link:

- Filter by link name, downward to its OMSs, and alias it as w.

Get all shared OMS links:

- Get links that appear in both groups.

```
link[.name="10.40.0.157 to 10.40.0.158"] | downward | link[.layer="OMS"] as p; link[.name="10.40.0.154 to 10.40.0.153"] | downward | link[.layer="OMS"] as w; link[.guid in p and .guid in w]
```

Figure 65.
Query Command



Figure 66.
Query Results

Complex Example 4

The following figures demonstrate how to find free ports for services in a site.

The query command that includes the following:

site[.name contains "MIL"] | inventory_item | port[.type="ETH"] | link | port as p; site[.name contains "MIL"] | inventory_item | port[.type="ETH" and .guid not in p]

Get all ETH ports of specific site used by links:

- Get site ports, transform to links and then to ports, and alias it as p.

Get all ports that are not in use by links:

- Get site ports, transform to ports, filter out those that appear in p.

site[.name contains "MIL"] | inventory_item | port[.type="ETH"] | link | port as p; site[.name contains "MIL"] | inventory_item | port[.type="ETH" and .guid not in p]

Figure 67.
Query Command

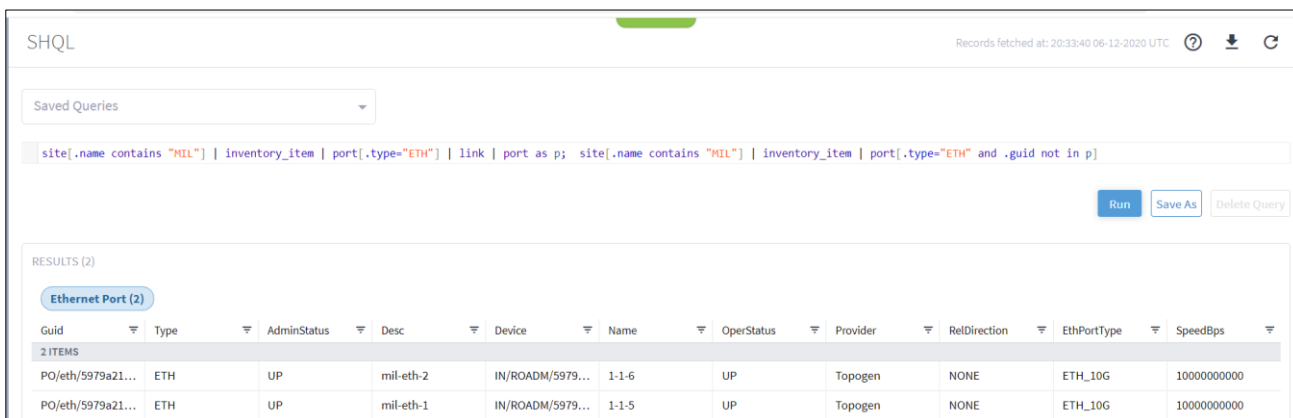


Figure 68.
Query Results

Try It Yourself

Try and create your own complex query.

Figure 69 shows an example that includes:

- Operands
- Transformations
- Collections
- Functions

```
site [.guid = "ST/9354b4a138dd_0_1_2"] | downward | inventory_item | port | link | upward | link[.layer in ("LSP", "E_LINE")] | limit(20) & port & inventory_item & site| upward
```

Figure 69.
Complex Query

Table 8. Breakdown of Complex Query

| Item | Type | See... |
|------------------------------------|---|--------------------------------------|
| site | Object type | Object Types and Properties |
| [.guid = "ST/31105c202abd"] | Condition (operand =) | Operands |
| downward | Function | SHQL Functions |
| inventory_item | Transformation object typ | Transforming/Collecting Object Types |
| port | Transformation object type | Transforming/Collecting Object Types |
| link | Transformation object type | Transforming/Collecting Object Types |
| upward | Function | SHQL Functions |
| link[.layer in ("LSP", "E_LINE")] | Transformation object type + condition (operand in) | Transforming/Collecting Object Types |
| limit(20) | Function | Output Functions |
| & port & inventory_item & site | Collections of object types | Transforming/Collecting Object Types |
| upward | Function | SHQL Functions |

Figure 70 shows the results.

```
site [.guid = "ST/9354b4a138dd_0_1_2"] | downward | inventory_item | port | link | upward | link[.layer in ("LSP", "E_LINE")] | limit(20) & port & inventory_item & site| upward
```

Run Save As Delete Query

RESULTS (58)

Site (20) Router (9) IGP Port (9) LSP Link (20)

| Guid | Latitude | Longitude | Name | Parent |
|-------------------|------------|------------|------|---------------------|
| 20 ITEMS | | | | |
| ST/a6da17cd217d | 51.2277411 | 6.7734556 | DUS | ST/a6da17cd217d_0 |
| ST/9354b4a138dd | 53.4083714 | -2.9915726 | LIV | ST/9354b4a138dd_0 |
| ST/839c2694c73e | 43.3619145 | -5.8493887 | OVE | ST/839c2694c73e_0 |
| ST/a9bd23d3ed70 | 54.597285 | -5.93012 | BEL | ST/a9bd23d3ed70_0 |
| ST/ea4a371e85b7 | 53.3498053 | -6.2603097 | DUB | ST/ea4a371e85b7_0 |
| ST/9354b4a138dd_0 | 53.4083714 | -2.9915726 | LIV | ST/9354b4a138dd_0_1 |
| ST/a6da17cd217d_0 | 51.2277411 | 6.7734556 | DUS | ST/a6da17cd217d_0_1 |
| ST/839c2694c73e_0 | 43.3619145 | -5.8493887 | OVE | ST/839c2694c73e_0_1 |
| ST/a9bd23d3ed70_0 | 54.597285 | -5.93012 | BEL | ST/a9bd23d3ed70_0_1 |
| ST/ea4a371e85b7_0 | 53.3498053 | -6.2603097 | DUB | ST/ea4a371e85b7_0_1 |

Figure 70.
Complex Query Results

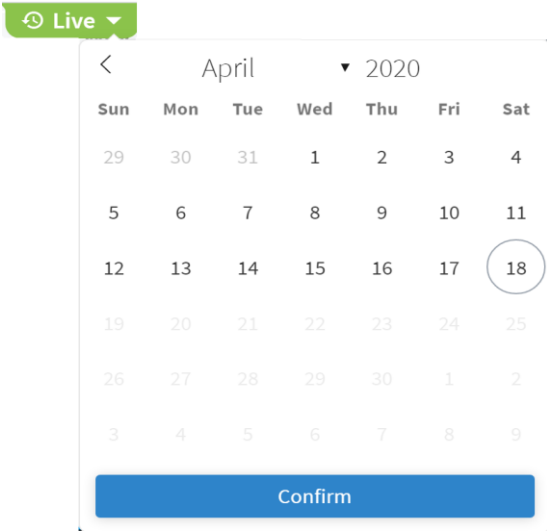
Use Time Machine

The time machine provides a snapshot of the state of the network as it was at a date in the past. In this mode, all applications reflect data and analysis that apply to this point in time.

You can use the time machine query the model as at a date in the past.

To change the model date:

1. In the applications bar in Crosswork Hierarchical Controller, select **SHQL**.
2. Click **Live**, select a date and click **Confirm**.



3. Run the required query.

Object Type Properties

The following sections list all the properties aligned with the different object types.

inventory_item Properties

Table 9. inventory_item properties

| inventory_item properties |
|---------------------------|
| .accessIdentifier |
| .cardStatus |
| .cardType |
| .children |
| .childrenPorts |
| .desc |
| .deviceFamily |
| .deviceType |
| domainName |
| .equipState |
| .extra |
| .formFactor |
| .guid |
| .hasRoadm |

inventory_item properties

.id

igpType

.isisInfo

.isisInfo.areaAddresses

.isisInfo.instanceIdentifier

.isisInfo.isDiscoveredL1

.isisInfo.isDiscoveredL2

.isisInfo.systemId

.isisInfo.topologyIdentifier

.managementIp

.modelName

.name

.oui

.parent

.partNumber

.pluggability

.provider

.reachabilityStatus

routerId

.serialNumber

.site

.softwareVersion

.supportedLambdas

.supportedPhysicalLayers

.supportedReachabilityMeters

.supportedSpeeds

.tags

.topologyId

.tunableType

.tunedLambda

.type

.vendor

port Properties

Table 10. port properties

| port properties |
|---|
| .adminGroups |
| .adminGroups.groupNumber |
| .adminGroups.name |
| .adminStatus |
| .aggRateBps |
| .ceInterfaceIp |
| .cePeRoutingProtocols |
| .channelNum |
| .childRole |
| .containingService |
| .desc |
| .device |
| .egressBandwidthProfile |
| .egressBandwidthProfile.cbsKb |
| .egressBandwidthProfile.cir |
| .egressBandwidthProfile.ebsKb |
| .egressBandwidthProfile.eir |
| .egressBandwidthProfile.qosPolicy |
| .ethPortType |
| .evcEgressBandwidthProfile |
| .evcEgressBandwidthProfile.cbsKb |
| .evcEgressBandwidthProfile.cir |
| .evcEgressBandwidthProfile.ebsKb |
| .evcEgressBandwidthProfile.eir |
| .evcEgressBandwidthProfile.qosPolicy |
| .evcIngressBandwidthProfilePerEvc |
| .evcIngressBandwidthProfilePerEvc.cbsKb |
| .evcIngressBandwidthProfilePerEvc.cir |
| .evcIngressBandwidthProfilePerEvc.ebsKb |

port properties

| |
|---|
| .evcIngressBandwidthProfilePerEvc.eir |
| .evcIngressBandwidthProfilePerEvc.qosPolicy |
| .exportRts |
| .extra |
| .guid |
| .hasServiceEndPointCapability |
| .id |
| .ifIndex |
| .igpInstanceName |
| .igpMetric |
| .importRts |
| .ingressBandwidthProfile |
| .ingressBandwidthProfile.cbsKb |
| .ingressBandwidthProfile.cir |
| .ingressBandwidthProfile.ebsKb |
| .ingressBandwidthProfile.eir |
| .ingressBandwidthProfile.qosPolicy |
| .ipAddress |
| .isLoopback |
| .isManagement |
| .isRouterId |
| .isStatsDummy |
| .isisInfo |
| .isisInfo.isisLevel |
| .isisInfo.netPrefixMetric |
| .isisInfo.portType |
| .lambdaNm |
| .links |
| .lowerPorts |
| .mappedCeTags |
| .mappedSvlan |
| mappingType |

port properties

.maxLinkBandwidth

.maxReservableLinkBandwidth

.mediaFrameType

.memberCount

.mtuSize

.name

.nonPrimaryIps

.ocType

.oduld

.oduPolarization

.oduType

.operStatus

ospfInfo

.otuType

.packetOpticalDetails

.packetOpticalDetails.availableBandwidth

.parent

.pdhType

.physicalAddress

.portCapabilities

.primaryIp

.protectionType

.provider

.range

.relDirection

.relPort

.rfParams

.routeDistinguisher

.rsvpReservedBandwidthBps

.rsvpStaticBandwidthBps

.spanLoss

.speedBps

| port properties |
|-----------------------------------|
| .srPrefixSids |
| .srlgs |
| .stsType |
| .tags |
| .teMetric |
| .type |
| .unreservedLinkBwForPriorityIndex |
| .upperPorts |
| .vcatsType |
| .vlan |
| .vpnRole |
| .vrfName |
| .vtType |
| .ycableType |

link Properties

Table 11. link properties

| link properties |
|---------------------------|
| .activeProtectionPriority |
| .adjacencySegmentInfo |
| .adminGroupConstraints |
| .bindingSid |
| .candidatePaths |
| .color |
| .deploymentType |
| .desc |
| .distanceMeters |
| .desc |
| .evcPerformanceProfile |
| .extra |
| .geometry |
| .guid |

link properties

.holdPriority

.id

.inverseLinkId

.isApproximated

.isBidi

.isCeVlanCosPreservation

.latencyMicros

.layer

.lineType

.lspTechnology

.mappingStatus

.name

.numberOfFibers

.operStatus

.owner

.pathGroupType

.paths

.portA

.portB

.prefixSegmentInfo

.protectedLayer

.protectionStatus

.provider

.role

.rxLabel

.segmentType

.setupPriority

.sidIndex

.sidLabel

.signalType

.speedBps

.srAdjacencySids

link properties

| |
|-------------|
| .srIgs |
| .tags |
| .teMetric |
| .txLabel |
| .usedByHops |
| .vclId |

site Properties

Table 12. site properties

site properties

| |
|------------|
| .extra |
| .guid |
| .id |
| .latitude |
| .longitude |
| .name |
| .parent |

service Properties

Table 13. service properties

service properties

| |
|---------------------------|
| .adminStatus |
| .anyToAnyOrHubRouteTarget |
| .containedPorts |
| .customerDetails |
| .customerName |
| .desc |
| .extra |
| .guid |
| .id |
| .name |
| .operStatus |

| service properties |
|------------------------|
| .provider |
| .serviceIntent |
| .serviceIntentRelation |
| .spokeRouteTarget |
| .tags |
| .type |
| .vpnTopology |

service_intent Properties

Table 14. service_intent properties

| service_intent properties |
|---|
| .adminState |
| .anyToAnyDetails |
| .anyToAnyDetails.minimumSites |
| .anyToAnyDetails.routeTarget |
| .baseTemplate |
| .customerName |
| .dataForNonMultiplexedService |
| .dataForNonMultiplexedService.ethPortType |
| .deploymentInfo.lastStatusChangeTimeStamp |
| .deploymentInfo.operation |
| .deploymentInfo.phase |
| .desc |
| .extra |
| .guid |
| .holdPriority |
| .hubAndSpokeDetails |
| .hubAndSpokeDetails.hubRouteTarget |
| .hubAndSpokeDetails.minimumHubSites |
| .hubAndSpokeDetails.minimumSpokeSites |
| .hubAndSpokeDetails.spokeRouteTarget |
| .id |

service_intent properties

| |
|---|
| .ipAddressAllocationPolicy |
| .isOamPmCollectionEnabled |
| .isRemoved |
| .isTemplate |
| .name |
| .oduType |
| .pathComputationSettings |
| .pathComputationSettings.backupPathOptimization |
| .pathComputationSettings.includeLinkStates |
| .pathComputationSettings.mainPathOptimization |
| .protectionSettings |
| .protectionSettings.protectionPolicy |
| .protectionSettings.resourceDiversityFor1Plus1 |
| .protectionSettings.resourceDiversityFor1Plus1.diversifiedResources |
| .protectionSettings.resourceDiversityFor1Plus1.diversityPolicy |
| .provider |
| .qosSettings |
| .qosSettings.cbsKb |
| .qosSettings.cir |
| .qosSettings.ebsKb |
| .qosSettings.eir |
| .qosSettings.qosPolicy |
| .resourceAllocationPolicy |
| .routeDistinguisher |
| .setupPriority |
| .srData |
| .tunnelType |
| .type |
| .underlayTech |
| .underlayTech.allowedNni |
| .underlayTech.ipMpls |
| .underlayTech.ipMpls.allowedNni |

| service_intent properties |
|---|
| .underlayTech.ipMpls.tunnelResiliency |
| .underlayTech.ipMpls.tunnelTypes |
| .underlayTech.mplsTp |
| .underlayTech.mplsTp.allowedNni |
| .underlayTech.mplsTp.tunnelResiliency |
| .underlayTech.mplsTp.tunnelUsageConstraints |
| .underlayTech.otn |
| .underlayTech.otn.allowedNni |
| .underlayTech.otn.serviceTunnelRate |
| .underlayTech.otn.serviceTunnelType |
| .underlayTech.selectedTech |
| .underlayTech.tunnelResiliency |
| .underlayTech.tunnelTypes |
| .underlayTech.wdm |
| .underlayTech.wdm.allowedNni |
| .vlanManipulation |
| .vpnTopology |

service_intent_resource Properties

Table 15. service_intent_resource Properties

| service_intent_resource properties |
|---|
| .constraintCompliance |
| .extra |
| .guid |
| .id |
| .includeType |
| .layer1Info |
| .layer1Info.tunnelRate |
| .layer2Info |
| .layer2Info.mappedCeTags |
| .layer2Info.qos |
| .layer2Info.qos.cbsKb |

| service_intent_resource properties |
|---|
| .layer2Info.qos.cir |
| .layer2Info.qos.ebsKb |
| .layer2Info.qos.eir |
| .layer2Info.qos.qosPolicy |
| .layer3Info |
| .layer3Info.cePeSettings |
| .layer3Info.cePeSettings.bgpRoutingDetails |
| .layer3Info.cePeSettings.bgpRoutingDetails.autonomousSystem |
| .layer3Info.cePeSettings.bgpRoutingDetails.peeringIp |
| .layer3Info.cePeSettings.ospfRoutingDetails |
| .layer3Info.cePeSettings.ospfRoutingDetails.metric |
| .layer3Info.cePeSettings.ospfRoutingDetails.ospfArea |
| .layer3Info.cePeSettings.routingMethod |
| .layer3Info.cePeSettings.staticRoutingDetails |
| .layer3Info.cePeSettings.staticRoutingDetails.entries.adminPreference |
| .layer3Info.cePeSettings.staticRoutingDetails.entries.ipPrefix |
| .layer3Info.cePeSettings.staticRoutingDetails.entries.nextHop |
| .layer3Info.ipAddress |
| .layer3Info.I3VpnRole |
| .order |
| .protectionRole |
| .resource |
| .resourceDiversity |
| .serviceIntent |
| .tag |
| .type |

event Properties

Table 16. event properties

| event properties |
|------------------|
| .count |
| .data |

| event properties |
|------------------|
| .guid |
| .lastUpdate |
| .machineId |
| .severity |
| .subType |
| .timeStamp |
| .type |
| .username |

srlg Resource Properties

Table 17. srlg properties

| srlg properties |
|-----------------|
| .externalId |
| .extra |
| .guid |
| .id |
| .name |
| .ordinal |
| .provider |
| .srlgs |

path Properties

Table 18. path properties

| path properties |
|-----------------|
| .ecmpWeight |
| .extra |
| .guid |
| .hops |
| .id |
| .link |
| .priority |
| .provider |

path properties

.srPathSegments

.tunPortA

.tunPortB

site_link Properties

Table 19. site_link properties

site_link properties

.depth

.guid

.id

.isCrossLink

.layer

.links

.name

.siteA

.siteB

.status

.tags

.utilization

visual_site Properties

Table 20. visual_site properties

visual_site properties

.depth

.guid

.id

.isOnlyAmplifiers

.latitude

.longitude

.name

.oneSize

.parent

visual_site properties

.reachabilityStatus

.routerSize

.site

.tags

inventory Properties

Table 21. inventory properties

inventory properties

.accessIdentifier

.cardStatus

.cardType

.children

.childrenPorts

.desc

.deviceFamily

.deviceType

.equipState

.extra

.formFactor

.guid

.hasRoadm

.isisInfo

.isisInfo.areaAddresses

.isisInfo.instanceIdentifier

.isisInfo.isDiscoveredL1

.isisInfo.isDiscoveredL2

.isisInfo.isisInstanceName

.isisInfo.systemId

.isisInfo.topologyIdentifier

.managementIp

.modelName

.name

| inventory properties |
|-----------------------------|
| .oui |
| .parent |
| .partNumber |
| .pluggability |
| .ports |
| .provider |
| .reachabilityStatus |
| .routerId |
| .serialNumber |
| .site |
| .softwareVersion |
| .srCapabilities |
| .srlgs |
| .supportedLambdas |
| .supportedPhysicalLayers |
| .supportedReacabilityMeters |
| .supportedSpeeds |
| .tags |
| .topologyId |
| .tunableType |
| .tunedLambda |
| .type |
| .unitType |
| .vendor |

srlg_risk_resource_mtm Properties

Table 22. srlg_risk_resource_mtm properties

| srlg_risk_resource_mtm properties |
|-----------------------------------|
| .guid |
| .id |
| .inventoryItem |
| .link |

srlg_risk_resource_mtm properties

.port

.provider

.srlg

risk_resource Properties

Table 23. risk_resource properties

risk_resource properties

.inventoryItem

.guid

.provider

.link

.srlg

.port

region Properties

Table 24. region properties

region properties

.geometry

.guid

.id

.name

.overlay

history Properties

Table 25. history properties

history properties

.action

.changes

.objGuid

.objName

.objSubtype

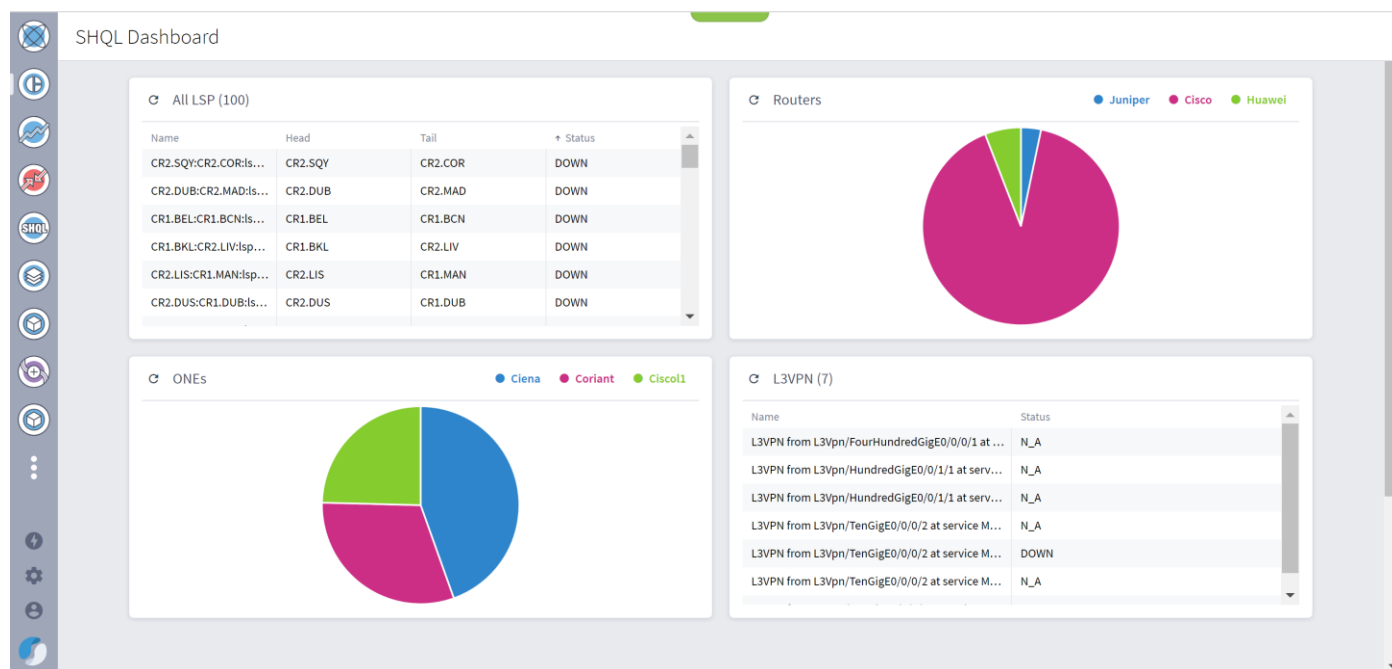
.objType

.timestamp

Creating SHQL Widgets

About SHQL Widgets

You can create customized widgets, rapidly with no development efforts and no software delivery. The widget query runs when opening the SHQL Dashboard application and the widgets are displayed. The widget also has a refresh button to run the query manually.



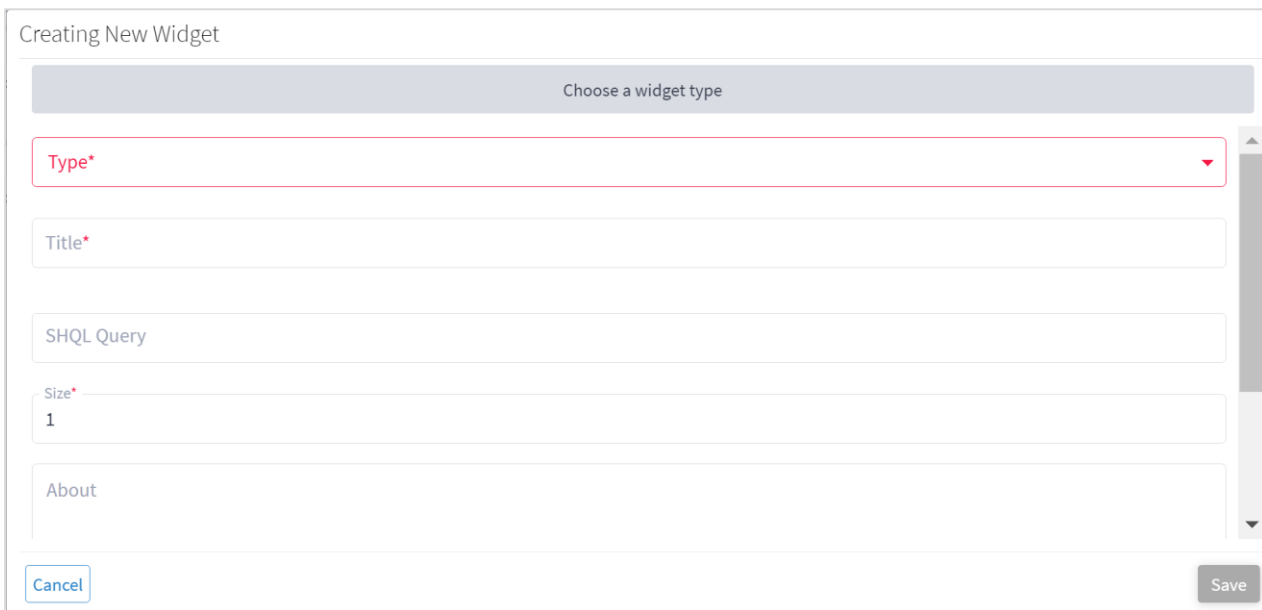
Creating SHQL Widgets

The widget attributes are:

- **Title:** The name of the SHQL widget as it appears in the SHQL Dashboard.
- **Query:** Must be up to 5 views, and limit(100).
- **Visual mode:**
 - **Pie/bar** – when the query contains counters only
 - **Graph** – when the query contains timestamp and counters
 - **Table** – when the query returns a list
- **'About'** text.
- **Optional:** Additional query to be used as a drill down into a widget (can be automatically generated (removing the counter and limit)).

To create an SHQL widget:

1. In the applications bar in Crosswork Hierarchical Controller, select **SHQL Widgets**.
2. Click + New SHQL Widget.



Creating New Widget

Choose a widget type

Type*

Title*

SHQL Query

Size*

1

About

Cancel Save

Figure 71.
Creating New Widget

3. Select the **Type**. This can be **Table**, **Graph**, or **Pie**.

Depending on the type of widget selected, guidance is provided on the SHQL permitted.

Appendix

Glossary

| Term | Description |
|----------|--|
| RESTCONF | A protocol based on HTTP for configuring data defined in YANG version 1 or YANG version 1.1 using the datastore concepts defined in the Network Configuration Protocol (NETCONF) |

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at <https://www.cisco.com/go/offices>.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <https://www.cisco.com/go/trademarks>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)