nisystemlink

Release 0.1.4

National Instruments

Apr 12, 2024

CONTENTS

CHAPTER

ONE

USER DOCUMENTATION

1.1 Getting Started

1.1.1 Tag API

Overview

The *TagManager* class is the primary entry point of the Tag API.

When constructing a *TagManager*, you can pass an *HttpConfiguration* (like one retrieved from the *HttpConfigurationManager*), or let *TagManager* use the default connection. The default connection depends on your SystemLink Client settings.

With a *TagManager* object, you can:

- Query, create, modify, and delete tags from the server
  - Use *open()* to get a tag's *TagData* from the server when you know the tag's `path`.
  - Use *query()* to get a *collection* of *TagData* objects based on the tags' `paths`, `keywords`, and/or `properties`.
  - Use *refresh()* to update a list of *TagData* objects with fresh metadata from the server.
  - Use *update()* to modify the server metadata for a list of tags, using either *TagData* objects to overwrite the server's tag data or *TagDataUpdate* objects to selectively update specific fields.
  - Use *delete()* to delete one or more tags from the server.
- Read and write tag values
  - Use *read()* to get a tag's current value. Via method parameters, you can also request the timestamp indicating when that value was last written and/or the aggregate data stored for the tag (if the tag's *collect_aggregates* attribute is enabled on the server).
  - Use *create_writer()* to get a *BufferedTagWriter* that will buffer a set of writes, and automatically send the writes when a given `buffer_size` is reached or when `max_buffer_time` has elapsed (or when *send_buffered_writes()* is called).
- Get a *TagSelection* that can help perform several of the above operations on several tags at once
  - Use *TagManager.create_selection()* if you already have a list of *TagData* objects that you want to perform a set of operations on.
  - Use *TagManager.open_selection()* if you just have a list of `paths` – optionally including glob-style wildcards! – with which to create the selection.

If you have a *TagSelection*, you can use it to *create* a *TagSubscription* that will trigger a *tag_changed* event any time one of the tags' values is changed.

Examples

Read and write individual tags

```
from datetime import timedelta

from nisystemlink.clients.tag import DataType, TagManager

mgr = TagManager()
tag = mgr.open("MyTags.Example Tag", DataType.DOUBLE, create=True)

with mgr.create_writer(buffer_size=10, max_buffer_time=timedelta(seconds=3)) as writer:
    writer.write(tag.path, tag.data_type, 3.5)
# Note: Exiting the "with" block automatically calls writer.send_buffered_writes()

read_result = mgr.read(tag.path)
assert read_result is not None
assert read_result.value == 3.5
```

Subscribe to tag changes

```
from contextlib import ExitStack
from time import sleep

from nisystemlink.clients.tag import DataType, TagData, TagManager, TagValueReader

SIMULATE_EXTERNAL_TAG_CHANGES = True


def on_tag_changed(tag: TagData, reader: TagValueReader) -> None:
    """Callback for tag_changed events."""
    path = tag.path
    data_type = tag.data_type.name

    if reader is not None:
        read_result = reader.read()
        # A read_result of None means that the tag has no value, but it *must*
        # have a value, because we got a tag_changed event!
        assert read_result is not None
        value = read_result.value
    else:
        value = "???"  # tag has unknown data type

    print(f'Tag changed: "{path}" = {value} ({data_type})')


mgr = TagManager()
if SIMULATE_EXTERNAL_TAG_CHANGES:
    mgr.open("MyTags.Example Tag", DataType.DOUBLE, create=True)
    writer = mgr.create_writer(buffer_size=1)
```

(continues on next page)

(continued from previous page)

```
with ExitStack() as stack:
    # Notes:
    # 1. The tags are assumed to already exist before this example is run, but
    #    setting SIMULATE_EXTERNAL_TAG_CHANGES to True will ensure there is one.
    # 2. Any tags that get added later will NOT automatically appear in the
    #    selection just because the path matches the wildcard used below; you
    #    must call one of the selection's refresh methods to update the tag list
    #    from the server. But even if you do that:
    # 3. The subscription will only contain the tags that were in the selection
    #    when the subscription was created. If you want the subscription to add
    #    new tags that were added to the selection, you must recreate it.
    paths = ["MyTags.*"]
    selection = stack.enter_context(mgr.open_selection(paths))
    if not selection.metadata:
        print(f"Found no tags that match {paths}")
    else:
        print("Matching tags:")
        for path in selection.metadata.keys():
            print(f" - {path}")
        print()

        subscription = stack.enter_context(selection.create_subscription())
        subscription.tag_changed += on_tag_changed

        # Wait forever, until a KeyboardInterrupt (Ctrl+C)
        print("Watching for tag changes; hit Ctrl+C to stop")
        try:
            i = 0
            while True:
                sleep(1)
                if SIMULATE_EXTERNAL_TAG_CHANGES:
                    writer.write("MyTags.Example Tag", DataType.DOUBLE, i)
                    i += 1
        except KeyboardInterrupt:
            pass
```

1.1.2 DataFrame API

Overview

The *DataFrameClient* class is the primary entry point of the DataFrame API.

When constructing a *DataFrameClient*, you can pass an *HttpConfiguration* (like one retrieved from the *HttpConfigurationManager*), or let *DataFrameClient* use the default connection. The default connection depends on your environment.

With a *DataFrameClient* object, you can:

- Create and delete data tables.
- Modify table metadata and query for tables by their metadata.
- Append rows of data to a table, query for rows of data from a table, and decimate table data.

- Export table data in a comma-separated values (CSV) format.

Examples

Create and write data to a table

```
import random
from datetime import datetime

from nisystemlink.clients.dataframe import DataFrameClient
from nisystemlink.clients.dataframe.models import (
    AppendTableDataRequest,
    Column,
    ColumnType,
    CreateTableRequest,
    DataFrame,
    DataType,
)

client = DataFrameClient()

# Create table
table_id = client.create_table(
    CreateTableRequest(
        name="Example Table",
        columns=[
            Column(name="ix", data_type=DataType.Int32, column_type=ColumnType.Index),
            Column(name="Float_Column", data_type=DataType.Float32),
            Column(name="Timestamp_Column", data_type=DataType.Timestamp),
        ],
    )
)

# Generate example data
frame = DataFrame(
    data=[[i, random.random(), datetime.now().isoformat()] for i in range(100)]
)

# Write example data to table
client.append_table_data(
    table_id, data=AppendTableDataRequest(frame=frame, endOfData=True)
)
```

Query and read data from a table

```
from nisystemlink.clients.dataframe import DataFrameClient
from nisystemlink.clients.dataframe.models import (
    DecimationMethod,
    DecimationOptions,
    QueryDecimatedDataRequest,
)

client = DataFrameClient()
```

(continues on next page)

(continued from previous page)

```python

# List a table
response = client.list_tables(take=1)
table = response.tables[0]

# Get table metadata by table id
client.get_table_metadata(table.id)

# Query decimated table data
request = QueryDecimatedDataRequest(
    decimation=DecimationOptions(
        x_column="index",
        y_columns=["col1"],
        intervals=1,
        method=DecimationMethod.MaxMin,
    )
)
client.query_decimated_data(table.id, request)
```

Export data from a table

```python
from shutil import copyfileobj

from nisystemlink.clients.dataframe import DataFrameClient
from nisystemlink.clients.dataframe.models import (
    ColumnFilter,
    ColumnOrderBy,
    ExportFormat,
    ExportTableDataRequest,
    FilterOperation,
)

client = DataFrameClient()

# List a table
response = client.list_tables(take=1)
table = response.tables[0]

# Export table data with query options
request = ExportTableDataRequest(
    columns=["col1"],
    order_by=[ColumnOrderBy(column="col2", descending=True)],
    filters=[
        ColumnFilter(column="col1", operation=FilterOperation.NotEquals, value="0")
    ],
    response_format=ExportFormat.CSV,
)

data = client.export_table_data(id=table.id, query=request)

# Write the export data to a file
with open(f"{table.name}.csv", "wb") as f:
```

(continues on next page)

(continued from previous page)

```
    copyfileobj(data, f)

# Alternatively, load the export data into a pandas dataframe
# import pandas as pd
# df = pd.read_csv(data)
```

1.1.3 Spec API

Overview

The *SpecClient* class is the primary entry point of the Specification Compliance API.

When constructing a *SpecClient*, you can pass an *HttpConfiguration* (like one retrieved from the *HttpConfigurationManager*), or let *SpecClient* use the default connection. The default connection depends on your environment.

With a *SpecClient* object, you can:

- Create and delete specifications under a product.
- Modify any fields of an existing specification
- Query for specifications on any fields using DynamicLinq syntax.

Examples

Create and Query Specifications

```
from nisystemlink.clients.core import HttpConfiguration
from nisystemlink.clients.spec import SpecClient
from nisystemlink.clients.spec.models import (
    Condition,
    ConditionRange,
    ConditionType,
    CreateSpecificationsRequest,
    NumericConditionValue,
    QuerySpecificationsRequest,
    SpecificationDefinition,
    SpecificationLimit,
    SpecificationType,
)

# Setup the server configuration to point to your instance of SystemLink Enterprise
server_configuration = HttpConfiguration(
    server_uri="https://yourserver.yourcompany.com",
    api_key="YourAPIKeyGeneratedFromSystemLink",
)
client = SpecClient(configuration=server_configuration)

# Create the spec requests
product = "Amplifier"
spec_requests = [
```

(continues on next page)

(continued from previous page)

```
    SpecificationDefinition(
        product_id=product,
        spec_id="spec1",
        type=SpecificationType.PARAMETRIC,
        category="Parametric Specs",
        name="output voltage",
        limit=SpecificationLimit(min=1.2, max=1.5),
        unit="mV",
    ),
    SpecificationDefinition(
        product_id=product,
        spec_id="spec2",
        type=SpecificationType.PARAMETRIC,
        category="Parametric Specs",
        name="input voltage",
        limit=SpecificationLimit(min=0.02, max=0.15),
        unit="mV",
        conditions=[
            Condition(
                name="Temperature",
                value=NumericConditionValue(
                    condition_type=ConditionType.NUMERIC,
                    range=[ConditionRange(min=-25, step=20, max=85)],
                    unit="C",
                ),
            ),
            Condition(
                name="Supply Voltage",
                value=NumericConditionValue(
                    condition_type=ConditionType.NUMERIC,
                    discrete=[1.3, 1.5, 1.7],
                    unit="mV",
                ),
            ),
        ],
    ),
    SpecificationDefinition(
        product_id=product,
        spec_id="spec3",
        type=SpecificationType.FUNCTIONAL,
        category="Noise Thresholds",
        name="noise",
    ),
]

# Create the specs on the server
client.create_specs(CreateSpecificationsRequest(specs=spec_requests))

# You can query specs based on any field using DynamicLinq syntax.
# These are just some representative examples.

response = client.query_specs(QuerySpecificationsRequest(product_ids=[product]))
```

(continues on next page)

(continued from previous page)

```
all_product_specs = response.specs

# Query based on spec id
response = client.query_specs(
    QuerySpecificationsRequest(product_ids=[product], filter='specId == "spec2"')
)
if response.specs:
    spec2 = response.specs[0]

# Query based on name
response = client.query_specs(
    QuerySpecificationsRequest(product_ids=[product], filter='name.Contains("voltage")')
)
voltage_specs = response.specs

# Query based on Category
response = client.query_specs(
    QuerySpecificationsRequest(
        product_ids=[product], filter='category == "Noise Thresholds"'
    )
)
noise_category = response.specs
print(noise_category)
```

Update and Delete Specifications

```
from nisystemlink.clients.core import HttpConfiguration
from nisystemlink.clients.spec import SpecClient
from nisystemlink.clients.spec.models import (
    QuerySpecificationsRequest,
    SpecificationDefinition,
    SpecificationType,
    UpdateSpecificationsRequest,
)

# Setup the server configuration to point to your instance of SystemLink Enterprise
server_configuration = HttpConfiguration(
    server_uri="https://yourserver.yourcompany.com",
    api_key="YourAPIKeyGeneratedFromSystemLink",
)
client = SpecClient(configuration=server_configuration)

# The query and delete examples assume you have created the specs from the query_specs␣
↪example
product = "Amplifier"

# update spec1 to change the block to "modifiedBlock"
# query the original spec
response = client.query_specs(
    QuerySpecificationsRequest(product_ids=[product], filter='specId == "spec1"')
)
if response.specs:
```

(continues on next page)

(continued from previous page)

```
    original_spec1 = response.specs[0]
    print(f"Original spec1 block: {original_spec1.block}")
    print(f"Original spec1 version: {original_spec1.version}")

# make the modifications
modified_spec = SpecificationDefinition(
    id=original_spec1.id,
    product_id=original_spec1.product_id,
    spec_id=original_spec1.spec_id,
    type=SpecificationType.FUNCTIONAL,
    keywords=["work", "reviewed"],
    block="modifiedBlock",
    version=original_spec1.version,
    workspace=original_spec1.workspace,
)
update_response = client.update_specs(
    specs=UpdateSpecificationsRequest(specs=[modified_spec])
)
if update_response and update_response.updated_specs:
    print(f"New spec1 version: {update_response.updated_specs[0].version}")

# query again to see new version
response = client.query_specs(
    QuerySpecificationsRequest(product_ids=[product], filter='specId == "spec1"')
)
if response.specs:
    original_spec1 = response.specs[0]
    print(f"Modified spec1 block: {original_spec1.block}")

# delete all the specs for the product
# query all specs
response = client.query_specs(QuerySpecificationsRequest(product_ids=[product]))
if response.specs:
    client.delete_specs(ids=[spec.id for spec in response.specs])
```

1.2 API Reference

1.2.1 nisystemlink.clients.core

pydantic model nisystemlink.clients.core.**ApiError**

Represents the standard error structure for SystemLink API responses.

```
{
   "$ref": "#/definitions/ApiError",
   "definitions": {
      "ApiError": {
         "title": "ApiError",
         "description": "Represents the standard error structure for SystemLink API␣
↪responses.",
```

(continues on next page)

(continued from previous page)

```
            "type": "object",
            "properties": {
               "name": {
                  "title": "Name",
                  "type": "string"
               },
               "code": {
                  "title": "Code",
                  "type": "integer"
               },
               "message": {
                  "title": "Message",
                  "type": "string"
               },
               "args": {
                  "title": "Args",
                  "default": [],
                  "type": "array",
                  "items": {
                     "type": "string"
                  }
               },
               "resourceType": {
                  "title": "Resourcetype",
                  "type": "string"
               },
               "resourceId": {
                  "title": "Resourceid",
                  "type": "string"
               },
               "innerErrors": {
                  "title": "Innererrors",
                  "default": [],
                  "type": "array",
                  "items": {
                     "$ref": "#/definitions/ApiError"
                  }
               }
            }
         }
      }
}
```

Fields

- *args*
- *code*
- *inner_errors*
- *message*
- *name*

- *resource_id*
- *resource_type*

field args: List[str] = []

Positional arguments for the error code.

field code: Optional[int] = None

Numeric error code.

field inner_errors: List[*ApiError*] = []

Inner errors when the top-level error represents more than one error.

field message: Optional[str] = None

Complete error message.

field name: Optional[str] = None

String error code.

field resource_id: Optional[str] = None

Identifier of the resource associated with the error.

field resource_type: Optional[str] = None

Type of resource associated with the error.

__init__(***data*)

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

classmethod construct(*_fields_set=None*, ***values*)

Creates a new model setting __dict__ and __fields_set__ from trusted or pre-validated data. Default values are respected, but no other validation is performed. Behaves as if *Config.extra = 'allow'* was set since it adds all passed values

Return type
Model

copy(*, *include=None*, *exclude=None*, *update=None*, *deep=False*)

Duplicate a model, optionally choose which fields to include, exclude and change.

Parameters

- **include** (Union[AbstractSetIntStr, MappingIntStrAny, None]) – fields to include in new model
- **exclude** (Union[AbstractSetIntStr, MappingIntStrAny, None]) – fields to exclude from new model, as with values this takes precedence over include
- **update** (Optional[DictStrAny]) – values to change/add in the new model. Note: the data is not validated before creating the new model: you should trust this data
- **deep** (bool) – set to *True* to make a deep copy of the model

Return type
Model

Returns
new model instance

dict(*, *include=None*, *exclude=None*, *by_alias=False*, *skip_defaults=None*, *exclude_unset=False*, *exclude_defaults=False*, *exclude_none=False*)

Generate a dictionary representation of the model, optionally specifying which fields to include or exclude.

Return type
DictStrAny

classmethod from_orm(*obj*)

Return type
Model

json(*, *include=None*, *exclude=None*, *by_alias=False*, *skip_defaults=None*, *exclude_unset=False*, *exclude_defaults=False*, *exclude_none=False*, *encoder=None*, *models_as_dict=True*, ***dumps_kwargs*)

Generate a JSON representation of the model, *include* and *exclude* arguments as per *dict()*.

encoder is an optional function to supply as *default* to json.dumps(), other arguments as per *json.dumps()*.

Return type
unicode

classmethod parse_file(*path*, *, *content_type=None*, *encoding='utf8'*, *proto=None*, *allow_pickle=False*)

Return type
Model

classmethod parse_obj(*obj*)

Return type
Model

classmethod parse_raw(*b*, *, *content_type=None*, *encoding='utf8'*, *proto=None*, *allow_pickle=False*)

Return type
Model

classmethod schema(*by_alias=True*, *ref_template='#/definitions/{model}'*)

Return type
DictStrAny

classmethod schema_json(*, *by_alias=True*, *ref_template='#/definitions/{model}'*, ***dumps_kwargs*)

Return type
unicode

classmethod update_forward_refs(***localns*)

Try to update ForwardRefs on fields based on this Model, globalns and localns.

Return type
None

classmethod validate(*value*)

Return type
Model

exception nisystemlink.clients.core.**ApiException**(*message=None*, *error=None*, *http_status_code=None*, *inner=None*)

Represents errors that occur when calling SystemLink APIs.

__init__(*message=None*, *error=None*, *http_status_code=None*, *inner=None*)

Initialize an exception.

Parameters

- **message** (Optional[str]) – The message describing the error.
- **error** (Optional[*nisystemlink.clients.core.ApiError*]) – The error returned by the API.
- **http_status_code** (Optional[int]) – The HTTP status code, if this exception was the result of an HTTP error.
- **inner** (Optional[Exception]) – The inner exception that caused the error.

property error: Optional[*ApiError*]

The error information returned by the SystemLink API, or None if the API did not return one or the error occurred trying to call the API.

property http_status_code: Optional[int]

The HTTP status code, if this exception was the result of an HTTP error.

property inner_exception: Optional[Exception]

The exception that caused this failure, if any.

property message: Optional[str]

The error message.

with_traceback()

Exception.with_traceback(tb) – set self.to tb and return self.

class nisystemlink.clients.core.**CloudHttpConfiguration**(*api_key*)

An *HttpConfiguration* specifically for use with SystemLink Cloud.

DEFAULT_TIMEOUT_MILLISECONDS = 60000

The default value of *timeout_milliseconds* to use when making API calls.

__init__(*api_key*)

Initialize a configuration for SystemLink Cloud using API key-based authentication.

Parameters

api_key (str) – The API key to send with requests.

Raises

ValueError – if api_key is empty.

property api_keys: Optional[Dict[str, str]]

The available API keys to use for authorization, or None if none were provided.

property cert_path: Optional[Path]

Local path to an SSL certificate file.

property password: Optional[str]

The password to use for HTTP authentication, or None if none was provided.

property server_uri: str

The urllib.parse.urlparse().scheme, urllib.parse.urlparse().hostname, and urllib.parse.urlparse().port of the web server hosting the SystemLink service to connect to.

Additional Uri properties such as urllib.parse.urlparse().path and urllib.parse.urlparse().query are ignored.

property timeout_milliseconds: int

The number of milliseconds before a request times out with an error.

Changing the timeout will not affect APIs that have already read the configuration.

property user_agent: Optional[str]

The string to pass the web server as the product name or names making the request, or None to use a library-specific default.

Changing the user-agent will not affect APIs that have already read the configuration.

property username: Optional[str]

The username to use for HTTP authentication, or None if none was provided.

class nisystemlink.clients.core.**HttpConfiguration**(*server_uri*, *api_key=None*, *username=None*, *password=None*, *cert_path=None*)

Represents the configuration for accessing a SystemLink service over HTTP.

DEFAULT_TIMEOUT_MILLISECONDS = 60000

The default value of *timeout_milliseconds* to use when making API calls.

__init__(*server_uri*, *api_key=None*, *username=None*, *password=None*, *cert_path=None*)

Initialize a configuration.

If neither `api_key` nor `username` and `password` are set, the configuration will use anonymous access, and any API calls that require authorization will fail.

Parameters

- **server_uri** (`str`) – The scheme, host, and port (if not default) of the web server hosting the SystemLink service to connect to. Additional Uri properties such as `urllib.parse.urlparse().path` and `urllib.parse.urlparse().query` are ignored.
- **api_key** (`Optional[str]`) – The API key to send with requests.
- **username** (`Optional[str]`) – The name of the user to use when authorization is required.
- **password** (`Optional[str]`) – The user's password to use when authorization is required.
- **cert_path** (`Optional[pathlib.Path]`) – Local path to an SSL certificate file.

Raises

- **ValueError** – if `server_uri` is missing scheme or host information.
- **ValueError** – if `username` or `password` is set, but not both.

property api_keys: Optional[Dict[str, str]]

The available API keys to use for authorization, or None if none were provided.

property cert_path: Optional[Path]

Local path to an SSL certificate file.

property password: Optional[str]

The password to use for HTTP authentication, or None if none was provided.

property server_uri: str

The `urllib.parse.urlparse().scheme`, `urllib.parse.urlparse().hostname`, and `urllib.parse.urlparse().port` of the web server hosting the SystemLink service to connect to.

Additional Uri properties such as `urllib.parse.urlparse().path` and `urllib.parse.urlparse().query` are ignored.

property timeout_milliseconds: int

The number of milliseconds before a request times out with an error.

Changing the timeout will not affect APIs that have already read the configuration.

property user_agent: Optional[str]

The string to pass the web server as the product name or names making the request, or None to use a library-specific default.

Changing the user-agent will not affect APIs that have already read the configuration.

property username: Optional[str]

The username to use for HTTP authentication, or None if none was provided.

class nisystemlink.clients.core.**HttpConfigurationManager**

Factory for *HttpConfiguration* objects.

HTTP_LOCALHOST_CONFIGURATION_ID = 'SYSTEMLINK_LOCALHOST'

The default ID of the SystemLink Server's configuration on the SystemLink Server itself.

HTTP_MASTER_CONFIGURATION_ID = 'SYSTEMLINK_MASTER'

The default ID of the SystemLink Server's configuration on systems registered using SystemLink Client.

classmethod get_configuration(*id=None*, *enable_fallbacks=True*)

Get the requested or default configuration.

Parameters

- **id** (Optional[str]) – The ID of the configuration to find.
- **enable_fallbacks** (Optional[bool]) – Whether or not to fallback to other known configurations, if id is unavailable.

Return type
nisystemlink.clients.core.HttpConfiguration

Returns
The configuration.

Raises

- **ValueError** – if id is None and enable_fallbacks is False.
- ***ApiException*** – if the specified (or default) configuration is not found.

class nisystemlink.clients.core.**JupyterHttpConfiguration**

An *HttpConfiguration* for Jupyter notebooks running in a SystemLink environment.

DEFAULT_TIMEOUT_MILLISECONDS = 60000

The default value of *timeout_milliseconds* to use when making API calls.

__init__()

Initialize a configuration for SystemLink using API key-based authentication provided through environment variables.

Raises
KeyError – if the expected environment variables are not set.

property api_keys: Optional[Dict[str, str]]

The available API keys to use for authorization, or None if none were provided.

property cert_path: Optional[Path]

Local path to an SSL certificate file.

property password: Optional[str]

The password to use for HTTP authentication, or None if none was provided.

property server_uri: str

The `urllib.parse.urlparse().scheme`, `urllib.parse.urlparse().hostname`, and `urllib.parse.urlparse().port` of the web server hosting the SystemLink service to connect to.

Additional Uri properties such as `urllib.parse.urlparse().path` and `urllib.parse.urlparse().query` are ignored.

property timeout_milliseconds: int

The number of milliseconds before a request times out with an error.

Changing the timeout will not affect APIs that have already read the configuration.

property user_agent: Optional[str]

The string to pass the web server as the product name or names making the request, or None to use a library-specific default.

Changing the user-agent will not affect APIs that have already read the configuration.

property username: Optional[str]

The username to use for HTTP authentication, or None if none was provided.

class nisystemlink.clients.core.helpers.**IteratorFileLike**(*iterator*)

A file-like object adapter that wraps a python iterator, providing a way to read from the iterator as if it was a file.

read(*size=-1*)

Read at most *size* bytes from the file-like object. If *size* is not specified or is negative, read until the iterator is exhausted and returns all bytes or characters read.

Return type
bytes

1.2.2 nisystemlink.clients.tag

class nisystemlink.clients.tag.**AsyncTagQueryResultCollection**(*first_page*, *total_count*, *skip*)

Represents a paginated list of tags returned by an asynchronous query.

property current_page: Optional[List[*TagData*]]

The current page of tag results that were last retrieved from the server, or None if there are no more results.

Use *move_next_page_async()* to get the next page of results.

async move_next_page_async()

Asynchronously retrieve the next page of query results from the server, returning them and updating *current_page*.

Does nothing if the last page has already been retrieved. Use *reset_async()* to start again from the first page.

Return type
Optional[List[*nisystemlink.clients.tag.TagData*]]

Returns
A task representing the asynchronous operation. On success, contains the next page of results, or None if there are no more results.

Raises
ApiException – if the API call fails.

`async reset_async()`

Asynchronously query the server for a fresh set of results, returning the first page and updating *current_page* and *total_count*.

Return type
List[*nisystemlink.clients.tag.TagData*]

Returns
A task representing the asynchronous operation. On success, contains the first page of results, or None if there are no results.

Raises
ApiException – if the API call fails.

`property total_count: int`

The total number of tags matched by the query at the time the query was made.

`class nisystemlink.clients.tag.BufferedTagWriter(stamper, buffer_size, flush_timer)`

Represents an *ITagWriter* that buffers tag writes instead of sending them immediately.

Writes that utilize automatic timestamps are based on the system time when buffered. Implementations may provide automatic sending of buffered writes based on different conditions. Unsent writes are discarded when the instance is deleted.

Note that *BufferedTagWriter* objects support using the `with` statement (or the `async with` statement), to automatically *send* any remaining buffered writes on exit.

`clear_buffered_writes()`

Clear any pending writes from *write()*.

Raises
ReferenceError – if the writer has been closed.

Return type
None

`get_tag_writer(path, data_type)`

Get a *TagValueWriter* for this path.

Parameters

- **path** (str) – The path of the tag to write.
- **data_type** (*nisystemlink.clients.tag.DataType*) – The data type of the tag to write.

Return type
nisystemlink.clients.tag.TagValueWriter

`send_buffered_writes()`

Write all of the pending writes from *write()* to the server.

Does nothing if there are no pending writes.

Raises

- **ReferenceError** – if the writer has been closed.
- *ApiException* – if the API call fails.

Return type
None

async send_buffered_writes_async()

Asynchronously write all of the pending writes from *write()* to the server.

Does nothing if there are no pending writes.

Raises

- **ReferenceError** – if the writer has been closed.
- *ApiException* – if the API call fails.

Return type
None

write(*path*, *data_type*, *value*, *, *timestamp=None*)

Write a tag's value.

Parameters

- **path** (str) – The path of the tag to write.
- **data_type** (*nisystemlink.clients.tag.DataType*) – The data type of the value to write.
- **value** (Union[bool, int, float, str, datetime.datetime]) – The tag value to write.
- **timestamp** (Optional[datetime.datetime]) – A custom timestamp to associate with the value, or None to have the server specify the timestamp.

Raises

- **ValueError** – if path is empty or invalid.
- **ValueError** – if path or value is None.
- **ValueError** – if data_type is invalid.
- **ValueError** – if value has the wrong data type.
- **ReferenceError** – if the writer has been closed.
- *ApiException* – if the API call fails.

Return type
None

write_async(*path*, *data_type*, *value*, *, *timestamp=None*)

Asynchronously write a tag's value.

Parameters

- **path** (str) – The path of the tag to write.
- **data_type** (*nisystemlink.clients.tag.DataType*) – The data type of the value to write.
- **value** (str) – The tag value to write.
- **timestamp** (Optional[datetime.datetime]) – A custom timestamp to associate with the value, or None to have the server specify the timestamp.

Return type
Awaitable[None]

Returns
A task representing the asynchronous operation.

Raises

- **ValueError** – if path is empty or invalid.
- **ValueError** – if path or value is None.
- **ValueError** – if data_type is invalid.
- **ValueError** – if value has the wrong data type.
- **ReferenceError** – if the writer has been closed.
- *ApiException* – if the API call fails.

class nisystemlink.clients.tag.**DataType**(*value*)

Represents the different data types for a SystemLink tag.

BOOLEAN = 4

A boolean tag.

Bool tags support collecting aggregate values for the count.

DATE_TIME = 6

A date and time tag that stores values in UTC ISO 8601 format.

DateTime tags support collecting aggregate values for the count.

DOUBLE = 1

A 64-bit floating-point tag following the IEEE standard.

Double tags support collecting aggregate values for the min, max, mean, and count.

INT32 = 2

A 32-bit signed integer tag.

Int32 tags support collecting aggregate values for the min, max, mean, and count. The mean is represented as a double.

STRING = 3

A string tag for arbitrary values.

String tags support collecting aggregate values for the count.

UINT64 = 5

A 64-bit unsigned integer tag.

UInt64 tags support collecting aggregate values for the min, max, mean, and count. The mean is represented as a double value and will truncate large values.

UNKNOWN = 0

An unknown or invalid data type.

Not a valid input to API calls, but used to represent tags whose data type isn't recognized.

property api_name: str

Web API name of the enum value.

class nisystemlink.clients.tag.**ITagReader**

Provides an interface for reading the current and aggregate values of a single SystemLink tag.

get_tag_reader(*path*, *data_type*)

Get a *TagValueReader* for this path.

Parameters

- **path** (str) – The path of the tag to read.
- **data_type** (*nisystemlink.clients.tag.DataType*) – The data type of the tag to read.

Return type
nisystemlink.clients.tag.TagValueReader

read(*path*, *, *include_timestamp=False*, *include_aggregates=False*)

Retrieve the current value of the tag with the given path from the server.

Optionally retrieves the aggregate values as well. The tag must exist.

Parameters

- **path** (str) – The path of the tag to read.
- **include_timestamp** (bool) – True to include the timestamp associated with the value in the result.
- **include_aggregates** (bool) – True to include the tag's aggregate values in the result if the tag is set to *TagData.collect_aggregates*.

Return type
Optional[*nisystemlink.clients.tag.TagWithAggregates*]

Returns
The value, and the timestamp and/or aggregate values if requested, or None if the tag exists but doesn't have a value.

Raises

- **ValueError** – if path is empty or invalid.
- **ValueError** – if path is None.
- **ReferenceError** – if the reader has been closed.
- ***ApiException*** – if the API call fails.

async read_async(*path*, *, *include_timestamp=False*, *include_aggregates=False*)

Asynchronously retrieve the current value of the tag with the given path from the server.

Optionally retrieves the aggregate values as well. The tag must exist.

Parameters

- **path** (str) – The path of the tag to read.
- **include_timestamp** (bool) – True to include the timestamp associated with the value in the result.
- **include_aggregates** (bool) – True to include the tag's aggregate values in the result if the tag is set to *TagData.collect_aggregates*.

Return type
Optional[*nisystemlink.clients.tag.TagWithAggregates*]

Returns
The value, and the timestamp and/or aggregate values if requested, or None if the tag exists but doesn't have a value.

Raises

- **ValueError** – if path is empty or invalid.
- **ValueError** – if path is None.
- **ReferenceError** – if the reader has been closed.
- ***ApiException*** – if the API call fails.

class nisystemlink.clients.tag.**ITagWriter**

Provides an interface for writing the current value of a single SystemLink tag.

get_tag_writer(*path*, *data_type*)

Get a *TagValueWriter* for this path.

Parameters

- **path** (str) – The path of the tag to write.
- **data_type** (*nisystemlink.clients.tag.DataType*) – The data type of the tag to write.

Return type
nisystemlink.clients.tag.TagValueWriter

write(*path*, *data_type*, *value*, *, *timestamp=None*)

Write a tag's value.

Parameters

- **path** (str) – The path of the tag to write.
- **data_type** (*nisystemlink.clients.tag.DataType*) – The data type of the value to write.
- **value** (Union[bool, int, float, str, datetime.datetime]) – The tag value to write.
- **timestamp** (Optional[datetime.datetime]) – A custom timestamp to associate with the value, or None to have the server specify the timestamp.

Raises

- **ValueError** – if path is empty or invalid.
- **ValueError** – if path or value is None.
- **ValueError** – if data_type is invalid.
- **ValueError** – if value has the wrong data type.
- **ReferenceError** – if the writer has been closed.
- ***ApiException*** – if the API call fails.

Return type
None

write_async(*path*, *data_type*, *value*, *, *timestamp=None*)

Asynchronously write a tag's value.

Parameters

- **path** (str) – The path of the tag to write.
- **data_type** (*nisystemlink.clients.tag.DataType*) – The data type of the value to write.
- **value** (str) – The tag value to write.
- **timestamp** (Optional[datetime.datetime]) – A custom timestamp to associate with the value, or None to have the server specify the timestamp.

Return type
Awaitable[None]

Returns
A task representing the asynchronous operation.

Raises

- **ValueError** – if path is empty or invalid.
- **ValueError** – if path or value is None.
- **ValueError** – if data_type is invalid.
- **ValueError** – if value has the wrong data type.
- **ReferenceError** – if the writer has been closed.
- ***ApiException*** – if the API call fails.

class nisystemlink.clients.tag.**RetentionType**(*value*)

Represents the different ways for the SystemLink tag historian to retain the history of tag values.

COUNT = 3

A set number of historical values for the tag are retained.

DURATION = 2

Historical values for the tag are retained for a set number of days.

INVALID = 0

An unknown or invalid tag retention type.

Not a valid input to API calls, but used to represent tags whose retention type isn't recognized.

NONE = 1

No history for the tag's value is retained.

PERMANENT = 4

All of the tag's values are retained until the tag is deleted.

class nisystemlink.clients.tag.**TagData**(*path*, *data_type=None*, *keywords=None*, *properties=None*)

Contains the metadata for a SystemLink tag.

__init__(*path*, *data_type=None*, *keywords=None*, *properties=None*)

Initialize an instance.

Parameters

- **path** (str) – The tag's path, which uses a dot-separated hierarchy to uniquely identify the tag on the server.
- **data_type** (Optional[*nisystemlink.clients.tag.DataType*]) – The data type for the tag's values.
- **keywords** (Optional[Iterable[str]]) – The tag's keywords.

- **properties** (Optional[Dict[str, str]]) – The tag's properties.

clear_retention()

Clear all retention settings, setting it to use a *TagData.retention_type* of *RetentionType.NONE*.

Parameters
tag – The tag whose retention will be cleared.

Return type
None

property collect_aggregates: bool

Whether the server should keep aggregate information for the tag.

The information collected depends on the tag's *data_type*.

property data_type: *DataType*

The data type for the tag's values.

Changing the data type of an existing tag requires deleting the tag and creating a new one of a different data type.

property keywords: List[str]

The list of keywords associated with the tag.

property path: str

The tag's path, which uses a dot-separated hierarchy to uniquely identify the tag on the server.

property properties: Dict[str, str]

The properties associated with the tag.

replace_keywords(*keywords*)

Replace all of the tag's *keywords* with those in keywords.

Parameters
keywords (Iterable[str]) – The tag's new keywords, or None to clear all keywords.

Return type
None

replace_properties(*properties*)

Replace all of the tag's *properties* with those in properties.

Parameters
properties (Dict[str, str]) – The tag's new properties, or None to clear all properties.

Return type
None

property retention_count: Optional[int]

The number of historical values to retain when *retention_type* is *RetentionType.COUNT*, or None to use the server-specified default of 10000.

property retention_days: Optional[int]

The number of days to keep a tag's historical values when *retention_type* is *RetentionType.DURATION*, or None to use the server-specified default of 30 days.

property retention_type: *RetentionType*

How the tag's historical values are retained by the tag historian, if available.

The *retention_count* and *retention_days* properties can further customize when values are removed from the historian.

The tag historian is an optional component for SystemLink Server installations, and is not available in SystemLink Cloud. The tag's historical values are not retained when the tag historian is not available, regardless of the retention type.

`set_retention_count`(*count*)

Set the number of historical values to retain.

Parameters
count (`int`) – The number of historical values to retain.

Return type
`None`

`set_retention_days`(*days*)

Set the historical values to be retained for the specified number of days.

Parameters
days (`int`) – The number of days a historical value will be kept.

Return type
`None`

`validate_path`()

Validate the path as an input and returns it.

Clients do not typically call this method directly.

Return type
`str`

Returns
The validated path.

Raises
ValueError – if the tag's path is invalid.

`validate_type`(*required_type*)

Validate that the tag's data type matches `required_type`.

Clients do not typically call this method directly.

Parameters
required_type (*nisystemlink.clients.tag.DataType*) – The data type required by the API.

Raises

- **ValueError** – if this is not a tag of the required type with a valid path.
- **ValueError** – if `required_type` is *DataType.UNKNOWN*.

Return type
`None`

class nisystemlink.clients.tag.**TagDataUpdate**(*path*, *data_type*, *keywords=None*, *properties=None*)

Contains information for updating parts of a tag's metadata on the server when used with the *TagManager.update()* method.

The update can add keywords, add new properties, change the value of existing properties, modify the collect aggregates setting, and modify retention settings. To remove a keyword or property, pass the entire *TagData* to the *TagManager.update()* method instead.

__init__(*path*, *data_type*, *keywords=None*, *properties=None*)

Initialize an update of a tag's keywords and/or properties.

Keywords and properties included in the update that are missing from the tag's metadata on the server are added, and properties that exist with a different value are replaced. At least one of `keywords` or `properties` must be specified.

Parameters

- **path** (`str`) – The path of the tag to update.
- **data_type** (*nisystemlink.clients.tag.DataType*) – The data type of the tag to update.
- **keywords** (`Optional[Iterable[str]]`) – The list of keywords that will be added to the tag's metadata on the server, or None to not add any keywords.
- **properties** (`Optional[Dict[str, str]]`) – The properties that will be added to or replaced in the tag's metadata on the server, or None to not modify any properties.

Raises

- **ValueError** – if `path` is None.
- **ValueError** – if both `keywords` and `properties` are None.

property collect_aggregates: Optional[bool]

The *TagData.collect_aggregates* setting to send with the update, or None to not send a value.

property data_type: *DataType*

The data type for the tag's values.

classmethod from_tagdata(*data*, *fields*)

Create an update by taking one or more fields from a *TagData*.

Parameters

- **data** (*nisystemlink.clients.tag.TagData*) – The metadata to send to the server in the update.
- **fields** (*nisystemlink.clients.tag.TagUpdateFields*) – One or more fields to include in the update.

Raises

- **ValueError** – if `data` is None.
- **ValueError** – if `fields` has no fields or invalid fields.

Return type

nisystemlink.clients.tag.TagDataUpdate

property keywords: Optional[Tuple[str, ...]]

The list of keywords to send with the update, or None to not send any keywords.

property path: str

The tag's path, which uses a dot-separated hierarchy to uniquely identify the tag on the server.

property properties: Optional[Dict[str, str]]

The properties send with the update, or None to not send any properties.

class nisystemlink.clients.tag.**TagManager**(*configuration=None*)

Represents common ways to create, read, and query SystemLink tags for a specific server connection.

__init__(*configuration=None*)

Initialize an instance.

Parameters
configuration (Optional[*nisystemlink.clients.core.HttpConfiguration*]) – Defines the web server to connect to and information about how to connect.

Raises
ApiException – if the current system cannot communicate with a SystemLink Server, or if the configuration provided by SystemLink Client cannot be found.

create_selection(*tags*)

Create an *TagSelection* that initially contains the given `tags` without retrieving any additional data from the server.

Parameters
tags (List[*nisystemlink.clients.tag.TagData*]) – The tags to include in the selection.

Return type
nisystemlink.clients.tag.TagSelection

Returns
The created selection.

Raises

- **ValueError** – if any of the given `tags` is None or has an invalid path.
- **ValueError** – if `tags` is None.

create_writer(**, buffer_size=None, max_buffer_time=None*)

Create a tag writer that buffers tag values until *send_buffered_writes()* is called on the returned object, `buffer_size` writes have been buffered, or `max_buffer_time` time has past since buffering a value, at which point the writes will be sent automatically.

Parameters

- **buffer_size** (Optional[int]) – The maximum number of tag writes to buffer before automatically sending them to the server.
- **max_buffer_time** (Optional[datetime.timedelta]) – The amount of time before writes are sent.

Return type
nisystemlink.clients.tag.BufferedTagWriter

Returns
The created writer. Close the writer to free resources.

Raises

- **ValueError** – if `buffer_size` and `max_buffer_time` are both None.
- **ValueError** – if `buffer_size` is less than one.

delete(*tags*)

Delete one or more tags from the server.

Parameters
tags (Iterable[Union[*nisystemlink.clients.tag.TagData*, str]]) – The tags (or tag paths) to delete.

Raises

- **ValueError** – if `tags` is None.
- **ValueError** – if `tags` contains any invalid tags.
- ***ApiException*** – if the API call fails.

Return type
`None`

delete_async(*tags=None*)

Asynchronously delete one or more tags from the server.

Parameters
tags (`Optional[Iterable[Union[nisystemlink.clients.tag.TagData, str]]]`) – The tags (or tag paths) to delete.

Return type
`Awaitable[None]`

Returns
A task representing the asynchronous operation.

Raises

- **ValueError** – if `tags` is None.
- **ValueError** – if `tags` contains any invalid tags.
- ***ApiException*** – if the API call fails.

get_tag_reader(*path*, *data_type*)

Get a *TagValueReader* for this path.

Parameters

- **path** (`str`) – The path of the tag to read.
- **data_type** (*nisystemlink.clients.tag.DataType*) – The data type of the tag to read.

Return type
nisystemlink.clients.tag.TagValueReader

open(*path*, *data_type=None*, ***, *create=None*)

Query the server for the metadata of a tag, optionally creating it if it doesn't already exist.

If `data_type` is provided, `create` defaults to True. If `data_type` is not provided, `create` cannot be set to True.

The call fails if the tag already exists as a different data type than specified or if it doesn't exist and `create` is False.

Parameters

- **path** (`str`) – The path of the tag to open.
- **data_type** (`Optional[`*nisystemlink.clients.tag.DataType*`]`) – The expected data type of the tag.
- **create** (`Optional[bool]`) – True to create the tag if it doesn't already exist, False to fail if it doesn't exist.

Return type
nisystemlink.clients.tag.TagData

Returns
Information about the tag.

Raises

- **ValueError** – if path is None or empty.
- **ValueError** – if data_type is invalid.
- **ValueError** – if create is True, but data_type is None.
- ***ApiException*** – if the API call fails.

async open_async(*path*, *data_type=None*, ***, *create=None*)

Asynchronously query the server for the metadata of a tag, optionally creating it if it doesn't already exist.

The call fails if the tag already exists as a different data type than specified or if it doesn't exist and create is False.

Parameters

- **path** (str) – The path of the tag to open.
- **data_type** (Optional[*nisystemlink.clients.tag.DataType*]) – The expected data type of the tag.
- **create** (Optional[bool]) – True to create the tag if it doesn't already exist, False to fail if it doesn't exist.

Return type
nisystemlink.clients.tag.TagData

Returns
A task representing the asynchronous operation. On success, contains information about the tag.

Raises

- **ValueError** – if path is None or empty.
- **ValueError** – if data_type is invalid.
- **ValueError** – if create is True, but data_type is None.
- ***ApiException*** – if the API call fails.

open_selection(*paths*)

Query the server for the metadata for the given tag paths and return the results in a *TagSelection*.

Parameters
paths (List[str]) – The paths of the tags to include in the selection. May include glob-style wildcards.

Return type
nisystemlink.clients.tag.TagSelection

Returns
The created selection with *TagSelection.metadata* containing the metadata.

Raises

- **ValueError** – if any of the given paths is None or invalid.
- **ValueError** – if paths contains duplicate paths.
- **ValueError** – if paths is None.

- **ApiException** – if the API call fails.

open_selection_async(*paths*)

Asynchronously query the server for the metadata for the given tag `paths` and return the results in a *TagSelection*.

Parameters

paths (List[str]) – The paths of the tags to include in the selection. May include glob-style wildcards.

Return type

Awaitable[*nisystemlink.clients.tag.TagSelection*]

Returns

A task representing the asynchronous operation. On success, contains the created selection with *TagSelection.metadata* containing the metadata.

Raises

- **ValueError** – if any of the given `paths` is None or invalid.
- **ValueError** – if `paths` contains duplicate paths.
- **ValueError** – if `paths` is None.
- **ApiException** – if the API call fails.

query(*paths=None*, *keywords=None*, *properties=None*, ***, *skip=0*, *take=None*)

Query the server for available tags matching the given criteria.

Parameters

- **paths** (Optional[Sequence[str]]) – List of tag paths to include in the result. May include glob-style wildcards.
- **keywords** (Optional[Iterable[str]]) – List of keywords that tags must have, or None.
- **properties** (Optional[Dict[str, str]]) – Mapping of properties and their values that tags must have, or None.
- **skip** (int) – The number of tags to initially skip in the results.
- **take** (Optional[int]) – The number of tags to include in each page of results.

Return type

nisystemlink.clients.tag.TagQueryResultCollection

Returns

A *TagQueryResultCollection* containing the first page of results. Enumerating the collection will retrieve additional pages according to the `take` parameter.

Raises

- **ValueError** – if `skip` or `take` is negative.
- **ValueError** – if `paths` is an empty list.
- **ValueError** – if any of `paths` are None.
- **ApiException** – if the API call fails.

async query_async(*paths=None*, *keywords=None*, *properties=None*, ***, *skip=0*, *take=None*)

Asynchronously query the server for available tags matching the given criteria.

Parameters

- **paths** (Optional[Sequence[str]]) – List of tag paths to include in the result. May include glob-style wildcards.
- **keywords** (Optional[Iterable[str]]) – List of keywords that tags must have, or None.
- **properties** (Optional[Dict[str, str]]) – Mapping of properties and their values that tags must have, or None.
- **skip** (int) – The number of tags to initially skip in the results.
- **take** (Optional[int]) – The number of tags to include in each page of results.

Return type
nisystemlink.clients.tag.AsyncTagQueryResultCollection

Returns
A task representing the asynchronous operation. On success, contains a *TagQueryResultCollection* containing the first page of results. Enumerating the collection will retrieve additional pages according to the take parameter.

Raises

- **ValueError** – if skip is negative.
- **ValueError** – if take is negative.
- ***ApiException*** – if the API call fails.

read(*path*, *, *include_timestamp=False*, *include_aggregates=False*)

Retrieve the current value of the tag with the given path from the server.

Optionally retrieves the aggregate values as well. The tag must exist.

Parameters

- **path** (str) – The path of the tag to read.
- **include_timestamp** (bool) – True to include the timestamp associated with the value in the result.
- **include_aggregates** (bool) – True to include the tag's aggregate values in the result if the tag is set to *TagData.collect_aggregates*.

Return type
Optional[*nisystemlink.clients.tag.TagWithAggregates*]

Returns
The value, and the timestamp and/or aggregate values if requested, or None if the tag exists but doesn't have a value.

Raises

- **ValueError** – if path is empty or invalid.
- **ValueError** – if path is None.
- **ReferenceError** – if the reader has been closed.
- ***ApiException*** – if the API call fails.

async read_async(*path*, *, *include_timestamp=False*, *include_aggregates=False*)

Asynchronously retrieve the current value of the tag with the given path from the server.

Optionally retrieves the aggregate values as well. The tag must exist.

Parameters

- **path** (str) – The path of the tag to read.
- **include_timestamp** (bool) – True to include the timestamp associated with the value in the result.
- **include_aggregates** (bool) – True to include the tag's aggregate values in the result if the tag is set to *TagData.collect_aggregates*.

Return type
Optional[*nisystemlink.clients.tag.TagWithAggregates*]

Returns
The value, and the timestamp and/or aggregate values if requested, or None if the tag exists but doesn't have a value.

Raises

- **ValueError** – if path is empty or invalid.
- **ValueError** – if path is None.
- **ReferenceError** – if the reader has been closed.
- ***ApiException*** – if the API call fails.

refresh(*tags*)

Populate the given tags with the latest metadata from the server.

Only the *TagData.path* needs to be initialized. Tags that don't exist on the server will have their *TagData.data_type* set to *DataType.UNKNOWN*.

Parameters
tags (List[*nisystemlink.clients.tag.TagData*]) – The tags to refresh.

Raises

- **ValueError** – if any tags are None or have invalid paths.
- **ValueError** – if tags is None.
- ***ApiException*** – if the API call fails.

Return type
None

async refresh_async(*tags*)

Asynchronously populate the given tags with the latest metadata from the server.

Only the *TagData.path* needs to be initialized. Tags that don't exist on the server will have their *TagData.data_type* set to *DataType.UNKNOWN*.

Parameters
tags (List[*nisystemlink.clients.tag.TagData*]) – The tags to refresh.

Return type
None

Returns
A task representing the asynchronous operation.

Raises

- **ValueError** – if any tags are None or have invalid paths.
- **ValueError** – if tags is None.

- *ApiException* – if the API call fails.

update(*updates*)

Update the metadata of one or more tags on the server, creating tags that don't exist.

If `updates` contains *TagData* objects, existing metadata will be replaced. If `updates` contains *TagDataUpdate* objects instead, tags that already exist will have their existing keywords, properties, and settings merged with those specified in the corresponding *TagDataUpdate*.

The call fails if any of the tags already exist as a different data type.

Parameters

updates (Union[Sequence[*nisystemlink.clients.tag.TagData*], Sequence[*nisystemlink.clients.tag.TagDataUpdate*]]) – The tags to update (if *TagData* objects are given), or the tag metadata updates to send (if *TagDataUpdate* objects are given).

Raises

- **ValueError** – if `updates` is None or empty.
- **ValueError** – if `updates` contains any invalid tags.
- **ValueError** – if `updates` contains both `TagData` objects and `TagDataUpdate` objects.
- *ApiException* – if the API call fails.

Return type

None

async update_async(*updates*)

Asynchronously update the metadata of one or more tags on the server, creating tags that don't exist.

If `updates` contains *TagData* objects, existing metadata will be replaced. If `updates` contains *TagDataUpdate* objects instead, tags that already exist will have their existing keywords, properties, and settings merged with those specified in the corresponding *TagDataUpdate*.

Parameters

updates (Union[Sequence[*nisystemlink.clients.tag.TagData*], Sequence[*nisystemlink.clients.tag.TagDataUpdate*]]) – The tags to update (if *TagData* objects are given), or the tag metadata updates to send (if *TagDataUpdate* objects are given).

Return type

None

Returns

A task representing the asynchronous operation.

Raises

- **ValueError** – if `updates` is None or empty.
- **ValueError** – if `updates` contains any invalid tags.
- **ValueError** – if `updates` contains both `TagData` objects and `TagDataUpdate` objects.
- *ApiException* – if the API call fails.

class nisystemlink.clients.tag.**TagPathUtilities**

Contains helper methods for interacting with tag paths.

classmethod validate(*path*)

Validate `path` as an input tag path.

Clients do not typically need to call this method directly.

Parameters
path (`str`) – The tag path to validate.

Return type
`str`

Returns
The validated path.

Raises

- **ValueError** – if the path is invalid.
- **ValueError** – if `path` is None.

classmethod validate_query(*path*)

Validate `path` as a tag path query.

Clients do not typically need to call this method directly.

Parameters
path (`str`) – The tag path to validate.

Return type
`str`

Returns
The validated path.

Raises

- **ValueError** – if the path is invalid.
- **ValueError** – if `path` is None.

class nisystemlink.clients.tag.**TagQueryResultCollection**(*first_page*, *total_count*, *skip*)

Represents a paginated list of tags returned by a query.

Iterating over the collection makes additional server requests to retrieve pages after the first.

property total_count: int

The total number of tags matched by the query at the time the query was made.

class nisystemlink.clients.tag.**TagSelection**(*tags*, *paths=None*)

Represents a set of tags that can be read, written, or deleted together.

Tags may be specified using glob-style wildcards to include multiple tags with a common path. Call *close()* to free resources. Tag reads are buffered, and the latest values are only retrieved on first read or by calling *refresh_values()*. Reads for tags that aren't in the selection return None, even if the tag exists on the server.

Note that *TagSelection* objects support using the `with` statement (or the `async with` statement), to *close()* the selection automatically on exit.

add_tags(*tags*)

Add one or more tags to the selection.

Tags that are already in the selection are ignored. The tags as given are immediately available in the *metadata* collection. Use *refresh_metadata()* to get the latest data for the tags. The tags will be available in the *values* collection but won't have latest a latest value until *refresh_values()* is called.

Parameters
tags (List[*nisystemlink.clients.tag.TagData*]) – The tags to add to the selection.

Raises
- **ValueError** – if any of the given tags are None or have an invalid path.
- **ValueError** – if tags is None.
- **ReferenceError** – if the selection has been closed.

Return type
None

clear_tags()

Remove all tags from the selection.

Raises
ReferenceError – if the selection has been closed.

Return type
None

close()

Clean up server resources associated with the selection.

Return type
None

async close_async()

Asynchronously clean up server resources associated with the selection.

Return type
None

Returns
A task representing the asynchronous operation.

create_subscription(*, *update_interval=None*)

Subscribe to receive events when tags in the selection are written to.

Updates will be queried from the server using the specified or default update interval.

The subscription will include any tags that are currently included in the selection when the subscription is created. That list can be seen via the *metadata* property. The subscription will also attempt to include any non-wildcard paths that are in the selection's current list of *paths*. Often, the list of *paths* directly coincides with the *metadata*, but the former may contain paths that did not exist when the selection's metadata was last updated from the server, e.g. via *refresh()*.

Closing, adding tags, or removing tags from the selection will not affect previously created subscriptions.

Parameters
update_interval (Optional[datetime.timedelta]) – How often to receive tag updates notifications from the server. Default is datetime.timedelta(seconds=30).

Return type
nisystemlink.clients.tag.TagSubscription

Returns
The created subscription.

Raises
- **ValueError** – if update_interval is negative.

- **ReferenceError** – if the selection has been closed.
- *ApiException* – if the API call fails.

create_subscription_async(*, *update_interval=None*)

Asynchronously subscribe to receive events when tags in the selection are written to.

Updates will be queried from the server using the specified or default update interval.

Closing, adding tags, or removing tags from the selection will not affect previously created subscriptions.

Parameters
update_interval (Optional[datetime.timedelta]) – How often to receive tag updates notifications from the server. Depending on the *TagManager* implementation in use, this may involve polling the server or have a minimum value.

Return type
Awaitable[*nisystemlink.clients.tag.TagSubscription*]

Returns
A task representing the asynchronous operation. On success, contains the created subscription.

Raises

- **ReferenceError** – if the selection has been closed.
- *ApiException* – if the API call fails.

delete_tags_from_server()

Delete all tags in the selection from the server.

The tags are not removed from the selection but are removed from *metadata* and *values*. If any of the tags are recreated, a call to *refresh_metadata()* will restore them in the collection of tags.

Raises

- **ReferenceError** – if the selection has been closed.
- *ApiException* – if the API call fails.

Return type
None

async delete_tags_from_server_async()

Asynchronously delete all tags in the selection from the server.

The tags are not removed from the selection but are removed from *metadata* and *values*. If any of the tags are recreated, a call to *refresh_metadata()* will restore them in the collection of tags.

Return type
None

Returns
A task representing the asynchronous operation.

Raises

- **ReferenceError** – if the selection has been closed.
- *ApiException* – if the API call fails.

get_tag_reader(*path*, *data_type*)

Get a *TagValueReader* for this path.

Parameters

- **path** (str) – The path of the tag to read.
- **data_type** (*nisystemlink.clients.tag.DataType*) – The data type of the tag to read.

Return type
nisystemlink.clients.tag.TagValueReader

property metadata: Dict[str, *TagData*]

The most recently retrieved metadata for tags in the selection, indexed by *TagData.path*.

Tags in the selection that do not exist on the server will not appear in the collection. Call *refresh_metadata()* to update the data contained in the collection.

open_tags(*paths*)

Add one or more tags to the selection by path.

Tags that are already in the selection are ignored. The tags will not be available in the *metadata* collection until *refresh_metadata()* is called or the *values* collection until *refresh_values()* is called.

Parameters
paths (List[str]) – The tag paths to add to the selection. May include glob-style wildcards.

Raises

- **ValueError** – if any of the given paths are None or invalid.
- **ValueError** – if paths is None.
- **ReferenceError** – if the selection has been closed.

Return type
None

property paths: Tuple[str, ...]

The paths of all tags in the selection, including those that do not exist on the server.

When using wildcards, the original paths with the wildcards are included in the list, not the paths matched by the wildcards.

read(*path*, *, *include_timestamp=False*, *include_aggregates=False*)

Retrieve the current value of the tag with the given path from the server.

Optionally retrieves the aggregate values as well. The tag must exist.

Parameters

- **path** (str) – The path of the tag to read.
- **include_timestamp** (bool) – True to include the timestamp associated with the value in the result.
- **include_aggregates** (bool) – True to include the tag's aggregate values in the result if the tag is set to *TagData.collect_aggregates*.

Return type
Optional[*nisystemlink.clients.tag.TagWithAggregates*]

Returns
The value, and the timestamp and/or aggregate values if requested, or None if the tag exists but doesn't have a value.

Raises

- **ValueError** – if path is empty or invalid.
- **ValueError** – if path is None.
- **ReferenceError** – if the reader has been closed.
- ***ApiException*** – if the API call fails.

async read_async(*path*, ***, *include_timestamp=False*, *include_aggregates=False*)

Asynchronously retrieve the current value of the tag with the given path from the server.

Optionally retrieves the aggregate values as well. The tag must exist.

Parameters

- **path** (str) – The path of the tag to read.
- **include_timestamp** (bool) – True to include the timestamp associated with the value in the result.
- **include_aggregates** (bool) – True to include the tag's aggregate values in the result if the tag is set to *TagData.collect_aggregates*.

Return type

Optional[*nisystemlink.clients.tag.TagWithAggregates*]

Returns

The value, and the timestamp and/or aggregate values if requested, or None if the tag exists but doesn't have a value.

Raises

- **ValueError** – if path is empty or invalid.
- **ValueError** – if path is None.
- **ReferenceError** – if the reader has been closed.
- ***ApiException*** – if the API call fails.

refresh()

Refresh both *TagData* and current values for all tags in the selection, updating *metadata* and *values* accordingly.

Call *refresh_metadata()* or *refresh_values()* to only partially refresh.

Raises

- **ReferenceError** – if the selection has been closed.
- ***ApiException*** – if the API call fails.

Return type

None

async refresh_async()

Asynchronously refresh both the *TagData* and current values for all tags in the selection, updating *metadata* and *values* accordingly.

Call *refresh_metadata_async()* or *refresh_values_async()* to only partially refresh.

Return type

None

Returns

A task representing the asynchronous operation.

Raises

- **ReferenceError** – if the selection has been closed.
- ***ApiException*** – if the API call fails.

refresh_metadata()

Refresh the *TagData* for all tags in the selection, updating *metadata* accordingly.

values will also be updated with new and removed tags, but those readers will continue to return previously available values until *refresh_values()* is called.

Raises

- **ReferenceError** – if the selection has been closed.
- ***ApiException*** – if the API call fails.

Return type
None

async refresh_metadata_async()

Asynchronously refresh the *TagData* for all tags in the selection, updating *metadata* accordingly.

values will also be updated with new and removed tags, but those readers will continue to return previously available values until *refresh_values()* is called.

Return type
None

Returns
A task representing the asynchronous operation.

Raises

- **ReferenceError** – if the selection has been closed.
- ***ApiException*** – if the API call fails.

refresh_values()

Refresh the current value of tags in the selection returned by the readers in the *values* collection.

Readers will return None for tags that no longer exist on the server, or exist but haven't yet been written to. New readers will be added to the collection for tags that have values but have not yet been added to *metadata* by a call to *refresh_metadata()*.

Raises

- **ReferenceError** – if the selection has been closed.
- ***ApiException*** – if the API call fails.

Return type
None

async refresh_values_async()

Asynchronously refresh the current value of tags in the selection returned by the readers in the *values* collection.

Readers will return None for tags that no longer exist on the server, or exist but haven't yet been written to. New readers will be added to the collection for tags that have values but have not yet been added to *metadata* by a call to *refresh_metadata()*.

Return type
None

Returns
A task representing the asynchronous operation.

Raises

- **ReferenceError** – if the selection has been closed.
- ***ApiException*** – if the API call fails.

remove_tags(*tags*)

Remove one or more tags from the selection.

The tags are not removed from the *metadata* and *values* collections until the next *refresh_metadata()*.

Tags not in the selection are ignored. Tags that were added to the selection using wildcard paths can only be removed by including the same wildcard paths. Tags matched by multiple wildcard paths remain in the selection until all of the paths are removed.

Parameters
tags (List[Union[*nisystemlink.clients.tag.TagData*, str]]) – The tags to remove from the selection. Either strings (paths) or *TagData* objects can be given. For *TagData* objects, only the *TagData.path* is used.

Raises

- **ValueError** – if tags is None.
- **ValueError** – if any of the given tags are None or have an invalid path.
- **ReferenceError** – if the selection has been closed.

Return type
None

reset_aggregates()

Reset tag value aggregates on the server for all tags in the selection.

Tag historical values are not modified. Has no effect on tags that are not set to *TagData.collect_aggregates*. Use *refresh_values()* to retrieve the new aggregates.

Raises

- **ReferenceError** – if the selection has been closed.
- ***ApiException*** – if the API call fails.

Return type
None

reset_aggregates_async()

Asynchronously reset tag value aggregates on the server for all tags in the selection.

Tag historical values are not modified. Has no effect on tags that are not set to *TagData.collect_aggregates*. Use *refresh_values()* to retrieve the new aggregates.

Return type
Awaitable[None]

Returns
A task representing the asynchronous operation.

Raises

- **ReferenceError** – if the selection has been closed.
- ***ApiException*** – if the API call fails.

property values: Dict[str, *TagValueReader*]

A *TagValueReader* for reading the most recently retrieved value for tags in the selection, indexed by *TagValueReader.path*.

Tags in the selection that do not exist on the server will not appear in the collection. Readers for tags without values on the server will return None when read. Call *refresh_values()* to update the values returned by the readers in the dictionary.

class nisystemlink.clients.tag.**TagSubscription**(*paths*, *heartbeat_timer*)

Represents a subscription for changes to one or more tags' values.

Call *close()* to stop receiving events.

Note that *TagSubscription* objects support using the `with` statement (or the `async with` statement), to *close()* the subsription automatically on exit.

tag_changed

An event that is triggered when one of the subscription's tag changes. The callback will receive a *TagData* parameter and an `Optional` [*TagValueReader*] parameter.

Example:

```
def my_callback(tag: TagData, reader: Optional[TagValueReader]):
    print("{} changed".format(tag.path))
    if reader is None:
        print(" - unknown data type")
    else:
        value = reader.read()
        assert value is not None
        print(" - new value: {}".format(value.value))

subscription.tag_changed += my_callback
```

close()

Close server resources associated with the subscription.

Further tag writes will not trigger new events.

Return type
None

async close_async()

Asynchronously close server resources associated with the subscription.

Further tag writes will not trigger new events.

Return type
None

Returns
A task representing the asynchronous operation.

class nisystemlink.clients.tag.**TagUpdateFields**(*value*)

Represents the various *TagData* fields that may be included in a *TagDataUpdate*.

Fields that aren't included are left unmodified when updates are sent to the server.

ALL = 15

Specify that all fields in the *TagData* should be included in the update.

Keywords and properties that already exist on the server will not be removed, even if they are missing from the update.

COLLECT_AGGREGATES = 4

Specify that the tag's *TagData.collect_aggregates* setting will be modified on the server.

KEYWORDS = 1

Specify that entries in the *TagData.keywords* that are missing from the tag's metadata on the server will be added.

PROPERTIES = 2

Specify that entries in the *TagData.properties* that are missing from or different in the tag's metadata on the server will be added or replaced.

RETENTION = 8

Specify that the tag's *TagData.retention_type*, *TagData.retention_count*, and *TagData.retention_days* settings will be modified on the server.

class nisystemlink.clients.tag.**TagValueReader**(*reader*, *tag*)

Represents the ability to read a single tag's value using an *ITagReader*.

__init__(*reader*, *tag*)

Initialize an instance.

Parameters

- **reader** (*nisystemlink.clients.tag.ITagReader*) – The *ITagReader* to use when reading values.
- **tag** (*nisystemlink.clients.tag.TagData*) – The tag whose values will be read.
- **path** – The path of the tag whose values will be read.

Raises

ValueError – if `tag` is not a tag of a valid data type and with a valid path.

property data_type: *DataType*

The data type of the tag associated with the value.

property path: str

The path of the tag associated with the value.

read(*, *include_timestamp=False*, *include_aggregates=False*)

Read the current value of the tag.

Parameters

- **include_timestamp** (`bool`) – True to include the timestamp associated with the value in the result.
- **include_aggregates** (`bool`) – True to include the tag's aggregate values in the result if the tag is set to *TagData.collect_aggregates*.

Return type

`Optional[`*nisystemlink.clients.tag.TagWithAggregates*`[TypeVar(typing.Any)]]`

Returns

The value, and the timestamp and/or aggregate values if requested, or None if the tag exists but doesn't have a value.

Raises

- **ReferenceError** – if the underlying reader has been closed.
- ***ApiException*** – if the API call fails.

async read_async(*, *include_timestamp=False*, *include_aggregates=False*)

Read the current value of the tag.

Parameters

- **include_timestamp** (bool) – True to include the timestamp associated with the value in the result.
- **include_aggregates** (bool) – True to include the tag's aggregate values in the result if the tag is set to *TagData.collect_aggregates*.

Return type

Optional[*nisystemlink.clients.tag.TagWithAggregates*[TypeVar(typing.Any)]]

Returns

The value, and the timestamp and/or aggregate values if requested, or None if the tag exists but doesn't have a value.

Raises

- **ReferenceError** – if the underlying reader has been closed.
- ***ApiException*** – if the API call fails.

class nisystemlink.clients.tag.**TagValueWriter**(*writer*, *tag*)

Represents the ability to write a single tag's value using an *ITagWriter*.

__init__(*writer*, *tag*)

Initialize an instance.

Parameters

- **writer** (*nisystemlink.clients.tag.ITagWriter*) – The *ITagWriter* to use when writing values.
- **tag** (*nisystemlink.clients.tag.TagData*) – The tag whose values will be written.
- **path** – The path of the tag whose values will be written.

Raises

ValueError – if tag is not a tag of a valid type and with a valid path.

property data_type: ***DataType***

The data type of the tag associated with the value.

property path: **str**

The path of the tag associated with the value.

write(*value*, *, *timestamp=None*)

Write the tag's value.

Parameters

- **value** (TypeVar(typing.Any)) – The tag value to write.
- **timestamp** (Optional[datetime.datetime]) – A custom timestamp to associate with the value, or None to have the server specify the timestamp.

Raises

- **ReferenceError** – if the underlying writer has been closed.
- ***ApiException*** – if the API call fails.

Return type
None

write_async(*value*, *, *timestamp=None*)

Write the tag's value.

Parameters

- **value** (TypeVar(typing.Any)) – The tag value to write.
- **timestamp** (Optional[datetime.datetime]) – A custom timestamp to associate with the value, or None to have the server specify the timestamp.

Raises

- **ReferenceError** – if the underlying writer has been closed.
- ***ApiException*** – if the API call fails.

Return type
Awaitable[None]

class nisystemlink.clients.tag.**TagWithAggregates**(*path*, *data_type*, *value*, *timestamp=None*, *count=None*, *min=None*, *max=None*, *mean=None*)

Represents a generic tag value with optional timestamp and optional aggregate values.

property count: Optional[int]

The number of times the tag has been written, or None if the tag is not collecting aggregates.

property data_type: *DataType*

The data type of the value.

property max: Optional[Union[int, float]]

The maximum value of the tag, or None if the tag is not collecting aggregates or the data type of the tag does not track a maximum value.

property mean: Optional[float]

The mean value of the tag, or None if the tag is not collecting aggregates or the data type of the tag does not track a mean value.

property min: Optional[Union[int, float]]

The minimum value of the tag, or None if the tag is not collecting aggregates or the data type of the tag does not track a minimum value.

property path: str

The path of the tag associated with the value.

property timestamp: Optional[datetime]

The timestamp associated with the value, if available.

property value: *_Any*

The value of the tag.

1.2.3 nisystemlink.clients.dataframe

class nisystemlink.clients.dataframe.**DataFrameClient**(*configuration=None*)

__init__(*configuration=None*)

Initialize an instance.

Parameters

configuration (Optional[*nisystemlink.clients.core.HttpConfiguration*]) – Defines the web server to connect to and information about how to connect. If not provided, an instance of *JupyterHttpConfiguration* is used.

Raises

ApiException – if unable to communicate with the DataFrame Service.

api_info()

Get information about available API operations.

Return type

nisystemlink.clients.dataframe.models.ApiInfo

Returns

Information about available API operations.

Raises

ApiException – if unable to communicate with the DataFrame Service.

list_tables(*take=None*, *id=None*, *order_by=None*, *order_by_descending=None*, *continuation_token=None*, *workspace=None*)

Lists available tables on the SystemLink DataFrame service.

Parameters

- **take** (Optional[int]) – Limits the returned list to the specified number of results. Defaults to 1000.
- **id** (Optional[List[str]]) – List of table IDs to filter by.
- **order_by** (Optional[Literal['CREATED_AT', 'METADATA_MODIFIED_AT', 'NAME', 'NUMBER_OF_ROWS', 'ROWS_MODIFIED_AT']]) – The sort order of the returned list of tables.
- **order_by_descending** (Optional[bool]) – Whether to sort descending instead of ascending. Defaults to false.
- **continuation_token** (Optional[str]) – The token used to paginate results.
- **workspace** (Optional[List[str]]) – List of workspace IDs to filter by.

Return type

nisystemlink.clients.dataframe.models.PagedTables

Returns

The list of tables with a continuation token.

Raises

ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

create_table(*table*)

Create a new table with the provided metadata and column definitions.

Parameters
table (*nisystemlink.clients.dataframe.models.CreateTableRequest*) – The request to create the table.

Return type
str

Returns
The ID of the newly created table.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

query_tables(*query*)

Queries available tables on the SystemLink DataFrame service and returns their metadata.

Parameters
query (*nisystemlink.clients.dataframe.models.QueryTablesRequest*) – The request to query tables.

Return type
nisystemlink.clients.dataframe.models.PagedTables

Returns
The list of tables with a continuation token.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

get_table_metadata(*id*)

Retrieves the metadata and column information for a single table identified by its ID.

Parameters
id (*str*) – Unique ID of a data table.

Return type
nisystemlink.clients.dataframe.models.TableMetadata

Returns
The metadata for the table.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

modify_table(*id*, *update*)

Modify properties of a table or its columns.

Parameters

- **id** (str) – Unique ID of a data table.
- **update** (*nisystemlink.clients.dataframe.models.ModifyTableRequest*) – The metadata to update.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

Return type
None

delete_table(*id*)

Deletes a table.

Parameters
id (*str*) – Unique ID of a data table.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

Return type
None

delete_tables(*ids*)

Deletes multiple tables.

Parameters
ids (*List[str]*) – List of unique IDs of data tables.

Return type
Optional[*nisystemlink.clients.dataframe.models.DeleteTablesPartialSuccess*]

Returns
A partial success if any tables failed to delete, or None if all tables were deleted successfully.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

modify_tables(*updates*)

Modify the properties associated with the tables identified by their IDs.

Parameters
updates (*nisystemlink.clients.dataframe.models.ModifyTablesRequest*) – The table modifications to apply.

Return type
Optional[*nisystemlink.clients.dataframe.models.ModifyTablesPartialSuccess*]

Returns
A partial success if any tables failed to be modified, or None if all tables were modified successfully.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

get_table_data(*id*, *columns=None*, *order_by=None*, *order_by_descending=None*, *take=None*, *continuation_token=None*)

Reads raw data from the table identified by its ID.

Parameters

- **id** (str) – Unique ID of a data table.
- **columns** (Optional[List[str]]) – Columns to include in the response. Data will be returned in the same order as the columns. If not specified, all columns are returned.
- **order_by** (Optional[List[str]]) – List of columns to sort by. Multiple columns may be specified to order rows that have the same value for prior columns. The columns used for

ordering do not need to be included in the columns list, in which case they are not returned. If not specified, then the order in which results are returned is undefined.

- **order_by_descending** (Optional[bool]) – Whether to sort descending instead of ascending. Defaults to false.
- **take** (Optional[int]) – Limits the returned list to the specified number of results. Defaults to 500.
- **continuation_token** (Optional[str]) – The token used to paginate results.

Return type
nisystemlink.clients.dataframe.models.PagedTableRows

Returns
The table data and total number of rows with a continuation token.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

append_table_data(*id*, *data*)

Appends one or more rows of data to the table identified by its ID.

Parameters

- **id** (str) – Unique ID of a data table.
- **data** (*nisystemlink.clients.dataframe.models.AppendTableDataRequest*) – The rows of data to append and any additional options.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

Return type
None

query_table_data(*id*, *query*)

Reads rows of data that match a filter from the table identified by its ID.

Parameters

- **id** (str) – Unique ID of a data table.
- **query** (*nisystemlink.clients.dataframe.models.QueryTableDataRequest*) – The filtering and sorting to apply when reading data.

Return type
nisystemlink.clients.dataframe.models.PagedTableRows

Returns
The table data and total number of rows with a continuation token.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

export_table_data(*id*, *query*)

Exports rows of data that match a filter from the table identified by its ID.

Parameters

- **id** (str) – Unique ID of a data table.

- **query** (*nisystemlink.clients.dataframe.models.ExportTableDataRequest*) – The filtering, sorting, and export format to apply when exporting data.

Return type
nisystemlink.clients.core.helpers.IteratorFileLike

Returns
A file-like object for reading the exported data.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

query_decimated_data(*id*, *query*)

Reads decimated rows of data from the table identified by its ID.

Parameters

- **id** (str) – Unique ID of a data table.
- **query** (*nisystemlink.clients.dataframe.models.QueryDecimatedDataRequest*) – The filtering and decimation options to apply when reading data.

Return type
nisystemlink.clients.dataframe.models.TableRows

Returns
The decimated table data.

Raises
ApiException – if unable to communicate with the DataFrame Service or provided an invalid argument.

pydantic model nisystemlink.clients.dataframe.models.**ApiInfo**

Information about the available API operations.


```
{
   "title": "ApiInfo",
   "description": "Information about the available API operations.",
   "type": "object",
   "properties": {
      "operations": {
         "$ref": "#/definitions/OperationsV1"
      }
   },
   "required": [
      "operations"
   ],
   "definitions": {
      "Operation": {
         "title": "Operation",
         "description": "Represents an operation that can be performed on a data␣
↪frame.",
         "type": "object",
         "properties": {
            "available": {
               "title": "Available",
```


(continues on next page)

(continued from previous page)


```
                "type": "boolean"
            },
            "version": {
                "title": "Version",
                "type": "integer"
            }
        },
        "required": [
            "available",
            "version"
        ]
    },
    "OperationsV1": {
        "title": "OperationsV1",
        "description": "The operations available in the routes provided by the v1␣
↪HTTP API.",
        "type": "object",
        "properties": {
            "createTables": {
                "$ref": "#/definitions/Operation"
            },
            "deleteTables": {
                "$ref": "#/definitions/Operation"
            },
            "modifyMetadata": {
                "$ref": "#/definitions/Operation"
            },
            "listTables": {
                "$ref": "#/definitions/Operation"
            },
            "readData": {
                "$ref": "#/definitions/Operation"
            },
            "writeData": {
                "$ref": "#/definitions/Operation"
            }
        },
        "required": [
            "createTables",
            "deleteTables",
            "modifyMetadata",
            "listTables",
            "readData",
            "writeData"
        ]
    }
  }
}
```


Fields

- *operations*

field operations: *OperationsV1* **[Required]**

pydantic model nisystemlink.clients.dataframe.models.**AppendTableDataRequest**

Contains the rows to append and optional flags. The `frame` field is required unless `endOfData` is true.


```
{
   "title": "AppendTableDataRequest",
   "description": "Contains the rows to append and optional flags. The ``frame``␣
↪field is\nrequired unless ``endOfData`` is true.",
   "type": "object",
   "properties": {
      "frame": {
         "$ref": "#/definitions/DataFrame"
      },
      "endOfData": {
         "title": "Endofdata",
         "type": "boolean"
      }
   },
   "definitions": {
      "DataFrame": {
         "title": "DataFrame",
         "description": "Data read from or to be written to a table.\n\nValues may␣
↪be ``None`` (if the column is of type ``NULLABLE``) or encoded as\na string in a␣
↪format according to each column's datatype:\n\n* BOOL: One of ``\"true\"`` or ``\
↪"false\"``, case-insensitive.\n* INT32: Any integer number in the range [-
↪2147483648, 2147483647],\n  surrounded by quotes.\n* INT64: Any integer number in␣
↪the range [-9223372036854775808,\n  9223372036854775807], surrounded by quotes.\
↪n* FLOAT32: A decimal number using a period for the decimal point, optionally\n ␣
↪in scientific notation, in the range [-3.40282347E+38, 3.40282347E+38],\n ␣
↪surrounded by quotes. Not all values within the range can be represented\n  with␣
↪32 bits. To preserve the exact binary encoding of the value when\n  converting to␣
↪a string, clients should serialize 9 digits after the\n  decimal. Instead of a␣
↪number, the value may be ``\"NaN\"`` (not a number),\n  ``\"Infinity\"``␣
↪(positive infinity), or ``\"-Infinity\"`` (negative\n  infinity), case-sensitive.\
↪n* FLOAT64: A decimal number using a period for the decimal point, optionally\n ␣
↪in scientific notation, in the range [-1.7976931348623157E+308,\n  1.
↪7976931348623157E+308], surrounded by quotes. Not all values within the\n  range␣
↪can be represented with 64 bits. To preserve the exact binary\n  encoding of the␣
↪value when converting to a string, clients should\n  serialize 17 digits after␣
↪the decimal. Instead of a number, the value may\n  be ``\"NaN\"`` (not a number),␣
↪``\"Infinity\"`` (positive infinity), or\n  ``\"-Infinity\"`` (negative infinity),
↪ case-sensitive.\n* STRING: Any quoted string.\n* TIMESTAMP: A date and time with␣
↪millisecond precision in ISO-8601 format\n  and time zone. For example: ``\"2022-
↪08-19T16:17:30.123Z\"``. If a time zone\n  is not provided, UTC is assumed. If a␣
↪time zone other than UTC is\n  provided, the value will be converted to UTC. If␣
↪more than three digits of\n  fractional seconds are provided, the time will be␣
↪truncated to three\n  digits (i.e. milliseconds).\n\nThe format is the same as a␣
↪serialized Pandas DataFrame with orient=\"split\"\nand index=False. See\nhttps://
↪pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_json.html.\n\nWhen␣
↪providing a DataFrame for appending rows, any of the table's columns\nnot␣
↪specified will receive a value of ``None``. If any such columns aren't\nnullable,␣
↪an error will be returned. If the entire columns property is left\nout, each row␣
```


(continues on next page)

(continued from previous page)


```
↪is assumed to contain all columns in the order specified when\nthe table was␣
↪created.",
         "type": "object",
         "properties": {
            "columns": {
               "title": "Columns",
               "type": "array",
               "items": {
                  "type": "string"
               }
            },
            "data": {
               "title": "Data",
               "type": "array",
               "items": {
                  "type": "array",
                  "items": {
                     "type": "string"
                  }
               }
            }
         },
         "required": [
            "data"
         ]
      }
   }
}
```


Fields

- *end_of_data*
- *frame*

field end_of_data: Optional[bool] = None

Whether the table should expect any additional rows to be appended in future requests.

field frame: Optional[*DataFrame*] = None

The data frame containing the rows to append.

pydantic model nisystemlink.clients.dataframe.models.**Column**

Defines a single column in a table.


```
{
   "title": "Column",
   "description": "Defines a single column in a table.",
   "type": "object",
   "properties": {
      "name": {
         "title": "Name",
         "type": "string"
      },
```


(continues on next page)

(continued from previous page)


```
      "dataType": {
         "$ref": "#/definitions/DataType"
      },
      "columnType": {
         "default": "NORMAL",
         "allOf": [
            {
               "$ref": "#/definitions/ColumnType"
            }
         ]
      },
      "properties": {
         "title": "Properties",
         "type": "object",
         "additionalProperties": {
            "type": "string"
         }
      }
   },
   "required": [
      "name",
      "dataType"
   ],
   "definitions": {
      "DataType": {
         "title": "DataType",
         "description": "Represents the different data types for a table column.",
         "enum": [
            "BOOL",
            "FLOAT32",
            "FLOAT64",
            "INT32",
            "INT64",
            "STRING",
            "TIMESTAMP"
         ],
         "type": "string"
      },
      "ColumnType": {
         "title": "ColumnType",
         "description": "Represents the different column types for a table column.",
         "enum": [
            "NORMAL",
            "INDEX",
            "NULLABLE"
         ],
         "type": "string"
      }
   }
}
```


Fields

- *column_type*
- *data_type*
- *name*
- *properties*

field column_type: *ColumnType* = ColumnType.Normal

The column type. Defaults to ColumnType.Normal.

field data_type: *DataType* [Required]

The data type of the column.

field name: str [Required]

The column name, which must be unique across all columns in the table.

field properties: Optional[Dict[str, str]] = None

User-defined properties associated with the column.

pydantic model nisystemlink.clients.dataframe.models.**ColumnFilter**

A filter to apply to the table data.


```
{
   "title": "ColumnFilter",
   "description": "A filter to apply to the table data.",
   "type": "object",
   "properties": {
      "column": {
         "title": "Column",
         "type": "string"
      },
      "operation": {
         "$ref": "#/definitions/FilterOperation"
      },
      "value": {
         "title": "Value",
         "type": "string"
      }
   },
   "required": [
      "column",
      "operation"
   ],
   "definitions": {
      "FilterOperation": {
         "title": "FilterOperation",
         "description": "Represents the different operations that can be used in a␣
↪filter.",
         "enum": [
            "EQUALS",
            "NOT_EQUALS",
            "LESS_THAN",
            "LESS_THAN_EQUALS",
            "GREATER_THAN",
            "GREATER_THAN_EQUALS",
```


(continues on next page)

(continued from previous page)

```
            "CONTAINS",
            "NOT_CONTAINS"
         ],
         "type": "string"
      }
   }
}
```

Fields

- *column*
- *operation*
- *value*

field column: str [Required]

The name of the column to use for filtering.

field operation: *FilterOperation* [Required]

How to compare the column's value with the specified value.

An error is returned if the column's data type does not support the specified operation: * String columns only support `EQUALS`, `NOT_EQUALS`, `CONTAINS`, and `NOT_CONTAINS`. * Non-string columns do not support `CONTAINS` or `NOT_CONTAINS`. * When `value` is `None`, the operation must be `EQUALS` or `NOT_EQUALS`. * When `value` is `NaN` for a floating-point column, the operation must be `NOT_EQUALS`.

field value: Optional[str] = None

The comparison value to use for filtering. An error will be returned if the value cannot be converted to the column's data type.

pydantic model nisystemlink.clients.dataframe.models.**ColumnMetadataPatch**

Specifies column properties to add, modify, or delete when editing table metadata.

```
{
   "title": "ColumnMetadataPatch",
   "description": "Specifies column properties to add, modify, or delete when␣
↪editing table metadata.",
   "type": "object",
   "properties": {
      "name": {
         "title": "Name",
         "type": "string"
      },
      "properties": {
         "title": "Properties",
         "type": "object",
         "additionalProperties": {
            "type": "string"
         }
      }
   },
   "required": [
      "name",
```

(continues on next page)

(continued from previous page)

```
      "properties"
   ]
}
```

Fields

- *name*
- *properties*

field name: str [Required]

The name of the column to modify.

field properties: Dict[str, Optional[str]] [Required]

The properties to modify. A map of key value properties containing the metadata to be added or modified. Setting a property value to `None` will delete the property.

pydantic model nisystemlink.clients.dataframe.models.**ColumnOrderBy**

Specifies a column to order by and the ordering direction.

```
{
   "title": "ColumnOrderBy",
   "description": "Specifies a column to order by and the ordering direction.",
   "type": "object",
   "properties": {
      "column": {
         "title": "Column",
         "type": "string"
      },
      "descending": {
         "title": "Descending",
         "type": "boolean"
      }
   },
   "required": [
      "column"
   ]
}
```

Fields

- *column*
- *descending*

field column: str [Required]

The name of the column to order by.

field descending: Optional[bool] = None

Whether the ordering should be in descending order.

class nisystemlink.clients.dataframe.models.**ColumnType**(*value*)

Represents the different column types for a table column.

Index = 'INDEX'

The column provides a unique value per row. Each table must provide exactly one INDEX column. The column's *DataType* must be INT32, INT64, or TIMESTAMP.

Normal = 'NORMAL'

The column has no special properties. This is the default.

Nullable = 'NULLABLE'

Rows may contain null values for this column. When appending rows, NULLABLE columns may be left out entirely, in which case all rows being appended will use null values for that column.

pydantic model nisystemlink.clients.dataframe.models.**CreateTableRequest**

Contains information needed to create a table, including its properties and column definitions.


```
{
   "title": "CreateTableRequest",
   "description": "Contains information needed to create a table, including its␣
↪properties and column definitions.",
   "type": "object",
   "properties": {
      "columns": {
         "title": "Columns",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Column"
         }
      },
      "name": {
         "title": "Name",
         "type": "string"
      },
      "properties": {
         "title": "Properties",
         "type": "object",
         "additionalProperties": {
            "type": "string"
         }
      },
      "workspace": {
         "title": "Workspace",
         "type": "string"
      }
   },
   "required": [
      "columns"
   ],
   "definitions": {
      "DataType": {
         "title": "DataType",
         "description": "Represents the different data types for a table column.",
         "enum": [
            "BOOL",
            "FLOAT32",
            "FLOAT64",
```


(continues on next page)

(continued from previous page)


```
            "INT32",
            "INT64",
            "STRING",
            "TIMESTAMP"
         ],
         "type": "string"
      },
      "ColumnType": {
         "title": "ColumnType",
         "description": "Represents the different column types for a table column.",
         "enum": [
            "NORMAL",
            "INDEX",
            "NULLABLE"
         ],
         "type": "string"
      },
      "Column": {
         "title": "Column",
         "description": "Defines a single column in a table.",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "type": "string"
            },
            "dataType": {
               "$ref": "#/definitions/DataType"
            },
            "columnType": {
               "default": "NORMAL",
               "allOf": [
                  {
                     "$ref": "#/definitions/ColumnType"
                  }
               ]
            },
            "properties": {
               "title": "Properties",
               "type": "object",
               "additionalProperties": {
                  "type": "string"
               }
            }
         },
         "required": [
            "name",
            "dataType"
         ]
      }
   }
}
```

Fields

- *columns*
- *name*
- *properties*
- *workspace*

field columns: List[*Column*] [Required]

The list of columns in the table. Exactly one column must have a *ColumnType* of INDEX.

field name: Optional[str] = None

The name to associate with the table. When not specified, a name will be assigned from the table's ID.

field properties: Optional[Dict[str, str]] = None

User-defined properties to associate with the table.

field workspace: Optional[str] = None

The workspace to create the table in. Uses the default workspace when not specified.

pydantic model nisystemlink.clients.dataframe.models.**DataFrame**

Data read from or to be written to a table.

Values may be `None` (if the column is of type `NULLABLE`) or encoded as a string in a format according to each column's datatype:

- BOOL: One of `"true"` or `"false"`, case-insensitive.
- INT32: Any integer number in the range [-2147483648, 2147483647], surrounded by quotes.
- INT64: Any integer number in the range [-9223372036854775808, 9223372036854775807], surrounded by quotes.
- FLOAT32: A decimal number using a period for the decimal point, optionally in scientific notation, in the range [-3.40282347E+38, 3.40282347E+38], surrounded by quotes. Not all values within the range can be represented with 32 bits. To preserve the exact binary encoding of the value when converting to a string, clients should serialize 9 digits after the decimal. Instead of a number, the value may be `"NaN"` (not a number), `"Infinity"` (positive infinity), or `"-Infinity"` (negative infinity), case-sensitive.
- FLOAT64: A decimal number using a period for the decimal point, optionally in scientific notation, in the range [-1.7976931348623157E+308, 1.7976931348623157E+308], surrounded by quotes. Not all values within the range can be represented with 64 bits. To preserve the exact binary encoding of the value when converting to a string, clients should serialize 17 digits after the decimal. Instead of a number, the value may be `"NaN"` (not a number), `"Infinity"` (positive infinity), or `"-Infinity"` (negative infinity), case-sensitive.
- STRING: Any quoted string.
- TIMESTAMP: A date and time with millisecond precision in ISO-8601 format and time zone. For example: `"2022-08-19T16:17:30.123Z"`. If a time zone is not provided, UTC is assumed. If a time zone other than UTC is provided, the value will be converted to UTC. If more than three digits of fractional seconds are provided, the time will be truncated to three digits (i.e. milliseconds).

The format is the same as a serialized Pandas DataFrame with orient="split" and index=False. See https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_json.html.

When providing a DataFrame for appending rows, any of the table's columns not specified will receive a value of `None`. If any such columns aren't nullable, an error will be returned. If the entire columns property is left out, each row is assumed to contain all columns in the order specified when the table was created.

```
{
   "title": "DataFrame",
   "description": "Data read from or to be written to a table.\n\nValues may be␣
↪``None`` (if the column is of type ``NULLABLE``) or encoded as\na string in a␣
↪format according to each column's datatype:\n\n* BOOL: One of ``\"true\"`` or ``\
↪"false\"``, case-insensitive.\n* INT32: Any integer number in the range [-
↪2147483648, 2147483647],\n  surrounded by quotes.\n* INT64: Any integer number in␣
↪the range [-9223372036854775808,\n  9223372036854775807], surrounded by quotes.\
↪n* FLOAT32: A decimal number using a period for the decimal point, optionally\n ␣
↪in scientific notation, in the range [-3.40282347E+38, 3.40282347E+38],\n ␣
↪surrounded by quotes. Not all values within the range can be represented\n  with␣
↪32 bits. To preserve the exact binary encoding of the value when\n  converting to␣
↪a string, clients should serialize 9 digits after the\n  decimal. Instead of a␣
↪number, the value may be ``\"NaN\"`` (not a number),\n  ``\"Infinity\"``␣
↪(positive infinity), or ``\"-Infinity\"`` (negative\n  infinity), case-sensitive.\
↪n* FLOAT64: A decimal number using a period for the decimal point, optionally\n ␣
↪in scientific notation, in the range [-1.7976931348623157E+308,\n  1.
↪7976931348623157E+308], surrounded by quotes. Not all values within the\n  range␣
↪can be represented with 64 bits. To preserve the exact binary\n  encoding of the␣
↪value when converting to a string, clients should\n  serialize 17 digits after␣
↪the decimal. Instead of a number, the value may\n  be ``\"NaN\"`` (not a number),␣
↪``\"Infinity\"`` (positive infinity), or\n  ``\"-Infinity\"`` (negative infinity),
↪ case-sensitive.\n* STRING: Any quoted string.\n* TIMESTAMP: A date and time with␣
↪millisecond precision in ISO-8601 format\n  and time zone. For example: ``\"2022-
↪08-19T16:17:30.123Z\"``. If a time zone\n  is not provided, UTC is assumed. If a␣
↪time zone other than UTC is\n  provided, the value will be converted to UTC. If␣
↪more than three digits of\n  fractional seconds are provided, the time will be␣
↪truncated to three\n  digits (i.e. milliseconds).\n\nThe format is the same as a␣
↪serialized Pandas DataFrame with orient=\"split\"\nand index=False. See\nhttps://
↪pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_json.html.\n\nWhen␣
↪providing a DataFrame for appending rows, any of the table's columns\nnot␣
↪specified will receive a value of ``None``. If any such columns aren't\nnullable,␣
↪an error will be returned. If the entire columns property is left\nout, each row␣
↪is assumed to contain all columns in the order specified when\nthe table was␣
↪created.",
   "type": "object",
   "properties": {
      "columns": {
         "title": "Columns",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "data": {
         "title": "Data",
         "type": "array",
         "items": {
            "type": "array",
            "items": {
               "type": "string"
            }
         }
```


(continues on next page)

(continued from previous page)

```
        }
    },
    "required": [
        "data"
    ]
}
```

Fields

- *columns*
- *data*

field columns: Optional[List[str]] = None

The names and order of the columns included in the data frame.

field data: List[List[Optional[str]]] [Required]

The data for each row with the order specified in the columns property. Must contain a value for each column in the columns property.

class nisystemlink.clients.dataframe.models.**DataType**(*value*)

Represents the different data types for a table column.

Bool = 'BOOL'

32-bit IEEE 754 floating-point number.

Float32 = 'FLOAT32'

32-bit IEEE 754 floating-point number.

Float64 = 'FLOAT64'

64-bit IEEE 754 floating-point number.

Int32 = 'INT32'

32-bit signed integers.

Int64 = 'INT64'

64-bit signed integers.

String = 'STRING'

Arbitrary string data.

Timestamp = 'TIMESTAMP'

Date and time represented in UTC with millisecond precision.

class nisystemlink.clients.dataframe.models.**DecimationMethod**(*value*)

Represents the different methods that can be used to decimate data.

EntryExit = 'ENTRY_EXIT'

Creates an `x_column` ordered set which will be divided in the number of `intervals` specified. For each of the intervals, the first and last row within the interval will be returned in addition to the maximum and minimum values for all the columns specified in `y_columns`.

Lossy = 'LOSSY'

Creates an `x_column` ordered set and returns an uniformly distributed sample of rows with as many rows as the number specified as `intervals`.

MaxMin = 'MAX_MIN'

Creates an `x_column` ordered set which will be divided in the number of `intervals` specified. For each of the intervals, the maximum and minimum values for all the columns specified in `y_columns` will be returned.

pydantic model nisystemlink.clients.dataframe.models.**DecimationOptions**

Contains the parameters to use for data decimation.


```
{
   "title": "DecimationOptions",
   "description": "Contains the parameters to use for data decimation.",
   "type": "object",
   "properties": {
      "xColumn": {
         "title": "Xcolumn",
         "type": "string"
      },
      "yColumns": {
         "title": "Ycolumns",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "intervals": {
         "title": "Intervals",
         "type": "integer"
      },
      "method": {
         "$ref": "#/definitions/DecimationMethod"
      }
   },
   "definitions": {
      "DecimationMethod": {
         "title": "DecimationMethod",
         "description": "Represents the different methods that can be used to␣
↪decimate data.",
         "enum": [
            "LOSSY",
            "MAX_MIN",
            "ENTRY_EXIT"
         ],
         "type": "string"
      }
   }
}
```


Fields

- *intervals*
- *method*
- *x_column*
- *y_columns*

field intervals: Optional[int] = None

Number of intervals to use for decimation. Defaults to 1000.

field method: Optional[*DecimationMethod*] = None

Specifies the method used to decimate the data. Defaults to *DecimationMethod.Lossy*

field x_column: Optional[str] = None

The name of the column that will be used as the x-axis for decimating the data. The column in the table that was specified as Index will be used if this field is excluded. Only numeric columns are supported. i.e. INT32, INT64, FLOAT32, FLOAT64 and TIMESTAMP.

field y_columns: Optional[List[str]] = None

A list of columns to decimate by. This property is only needed when the specified method is MAX_MIN or ENTRY_EXIT. Only numeric columns are supported. i.e. INT32, INT64, FLOAT32, FLOAT64 and TIMESTAMP.

pydantic model nisystemlink.clients.dataframe.models.**DeleteTablesPartialSuccess**

The result of deleting multiple tables when one or more tables could not be deleted.


```
{
   "title": "DeleteTablesPartialSuccess",
   "description": "The result of deleting multiple tables when one or more tables␣
↪could not be deleted.",
   "type": "object",
   "properties": {
      "deletedTableIds": {
         "title": "Deletedtableids",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "failedTableIds": {
         "title": "Failedtableids",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "error": {
         "$ref": "#/definitions/ApiError"
      }
   },
   "required": [
      "deletedTableIds",
      "failedTableIds",
      "error"
   ],
   "definitions": {
      "ApiError": {
         "title": "ApiError",
         "description": "Represents the standard error structure for SystemLink API␣
↪responses.",
         "type": "object",
```


(continues on next page)

(continued from previous page)

```
        "properties": {
           "name": {
              "title": "Name",
              "type": "string"
           },
           "code": {
              "title": "Code",
              "type": "integer"
           },
           "message": {
              "title": "Message",
              "type": "string"
           },
           "args": {
              "title": "Args",
              "default": [],
              "type": "array",
              "items": {
                 "type": "string"
              }
           },
           "resourceType": {
              "title": "Resourcetype",
              "type": "string"
           },
           "resourceId": {
              "title": "Resourceid",
              "type": "string"
           },
           "innerErrors": {
              "title": "Innererrors",
              "default": [],
              "type": "array",
              "items": {
                 "$ref": "#/definitions/ApiError"
              }
           }
        }
     }
  }
}
```

Fields

- *deleted_table_ids*
- *error*
- *failed_table_ids*

field deleted_table_ids: List[str] [Required]

The IDs of the tables that were successfully deleted.

field error: *ApiError* **[Required]**

The error that occurred when deleting the tables.

field failed_table_ids: **List[str] [Required]**

The IDs of the tables that could not be deleted.

class nisystemlink.clients.dataframe.models.**ExportFormat**(*value*)

The format of the exported data.

CSV = 'CSV'

Comma-separated values.

pydantic model nisystemlink.clients.dataframe.models.**ExportTableDataRequest**

Specifies the parameters for a data export with ordering and filtering.


```
{
   "title": "ExportTableDataRequest",
   "description": "Specifies the parameters for a data export with ordering and␣
↪filtering.",
   "type": "object",
   "properties": {
      "columns": {
         "title": "Columns",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "orderBy": {
         "title": "Orderby",
         "type": "array",
         "items": {
            "$ref": "#/definitions/ColumnOrderBy"
         }
      },
      "filters": {
         "title": "Filters",
         "type": "array",
         "items": {
            "$ref": "#/definitions/ColumnFilter"
         }
      },
      "responseFormat": {
         "$ref": "#/definitions/ExportFormat"
      }
   },
   "required": [
      "responseFormat"
   ],
   "definitions": {
      "ColumnOrderBy": {
         "title": "ColumnOrderBy",
         "description": "Specifies a column to order by and the ordering direction.
↪",
```


(continues on next page)

(continued from previous page)

```
        "type": "object",
        "properties": {
           "column": {
              "title": "Column",
              "type": "string"
           },
           "descending": {
              "title": "Descending",
              "type": "boolean"
           }
        },
        "required": [
           "column"
        ]
     },
     "FilterOperation": {
        "title": "FilterOperation",
        "description": "Represents the different operations that can be used in a␣
↪filter.",
        "enum": [
           "EQUALS",
           "NOT_EQUALS",
           "LESS_THAN",
           "LESS_THAN_EQUALS",
           "GREATER_THAN",
           "GREATER_THAN_EQUALS",
           "CONTAINS",
           "NOT_CONTAINS"
        ],
        "type": "string"
     },
     "ColumnFilter": {
        "title": "ColumnFilter",
        "description": "A filter to apply to the table data.",
        "type": "object",
        "properties": {
           "column": {
              "title": "Column",
              "type": "string"
           },
           "operation": {
              "$ref": "#/definitions/FilterOperation"
           },
           "value": {
              "title": "Value",
              "type": "string"
           }
        },
        "required": [
           "column",
           "operation"
        ]
```

(continues on next page)

(continued from previous page)

```
      },
      "ExportFormat": {
         "title": "ExportFormat",
         "description": "The format of the exported data.",
         "enum": [
            "CSV"
         ],
         "type": "string"
      }
   }
}
```

Fields

- *columns*
- *filters*
- *order_by*
- *response_format*

field columns: Optional[List[str]] = None

The names of columns to include in the export. The export will include the columns in the same order specified in this parameter. All columns are included in the order specified at table creation if this property is excluded.

field filters: Optional[List[*ColumnFilter*]] = None

A list of columns to filter by. Only rows whose columns contain values matching all of the specified filters are returned. The columns used for filtering do not need to be included in the columns list, in which case they are not included in the export.

field order_by: Optional[List[*ColumnOrderBy*]] = None

A list of columns to order the results by. Multiple columns may be specified to order rows that have the same value for prior columns. The columns used for sorting do not need to be included in the columns list, in which case they are not included in the export.

field response_format: *ExportFormat* [Required]

The format of the exported data. The only response format currently supported is CSV.

class nisystemlink.clients.dataframe.models.**FilterOperation**(*value*)

Represents the different operations that can be used in a filter.

pydantic model nisystemlink.clients.dataframe.models.**ModifyTableRequest**

Contains the metadata properties to modify. Values not included will remain unchanged.

```
{
   "title": "ModifyTableRequest",
   "description": "Contains the metadata properties to modify. Values not included␣
↪will remain unchanged.",
   "type": "object",
   "properties": {
      "metadataRevision": {
         "title": "Metadatarevision",
         "type": "integer"
```

(continues on next page)

(continued from previous page)

```
      },
      "name": {
         "title": "Name",
         "type": "string"
      },
      "workspace": {
         "title": "Workspace",
         "type": "string"
      },
      "properties": {
         "title": "Properties",
         "type": "object",
         "additionalProperties": {
            "type": "string"
         }
      },
      "columns": {
         "title": "Columns",
         "type": "array",
         "items": {
            "$ref": "#/definitions/ColumnMetadataPatch"
         }
      }
   },
   "definitions": {
      "ColumnMetadataPatch": {
         "title": "ColumnMetadataPatch",
         "description": "Specifies column properties to add, modify, or delete when␣
↪editing table metadata.",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "type": "string"
            },
            "properties": {
               "title": "Properties",
               "type": "object",
               "additionalProperties": {
                  "type": "string"
               }
            }
         },
         "required": [
            "name",
            "properties"
         ]
      }
   }
}
```

Fields

- *columns*
- *metadata_revision*
- *name*
- *properties*
- *workspace*

field columns: Optional[List[*ColumnMetadataPatch*]] = None

Updates to the column properties. Cannot add or remove columns, or change the name of a column.

field metadata_revision: Optional[int] = None

When specified, this is an integer that must match the last known revision number of the table, incremented by one. If it doesn't match the current `metadataRevision` incremented by one at the time of execution, the modify request will be rejected with a 409 Conflict. This is used to ensure that changes to this table's metadata are based on a known, previous state.

field name: Optional[str] = None

The new name of the table. Setting to `None` will reset the name to the table's ID.

field properties: Optional[Dict[str, Optional[str]]] = None

The properties to modify. A map of key value properties containing the metadata to be added or modified. Setting a property value to `None` will delete the property.

field workspace: Optional[str] = None

The new workspace for the table. Setting to `None` will reset to the default workspace. Changing the workspace requires permission to delete the table in its current workspace and permission to create the table in its new workspace.

pydantic model `nisystemlink.clients.dataframe.models.`**ModifyTablesPartialSuccess**

The result of modifying multiple tables when one or more tables could not be modified.


```
{
   "title": "ModifyTablesPartialSuccess",
   "description": "The result of modifying multiple tables when one or more tables␣
↪could not be modified.",
   "type": "object",
   "properties": {
      "modifiedTableIds": {
         "title": "Modifiedtableids",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "failedModifications": {
         "title": "Failedmodifications",
         "type": "array",
         "items": {
            "$ref": "#/definitions/TableMetadataModification"
         }
      },
      "error": {
         "$ref": "#/definitions/ApiError"
```


(continues on next page)

(continued from previous page)

```
      }
   },
   "required": [
      "modifiedTableIds",
      "failedModifications",
      "error"
   ],
   "definitions": {
      "TableMetadataModification": {
         "title": "TableMetadataModification",
         "description": "Contains the metadata properties to modify. Values not␣
↪included in the\nrequest or included with a ``None`` value will remain unchanged.
↪",
         "type": "object",
         "properties": {
            "id": {
               "title": "Id",
               "type": "string"
            },
            "metadataRevision": {
               "title": "Metadatarevision",
               "type": "integer"
            },
            "name": {
               "title": "Name",
               "type": "string"
            },
            "workspace": {
               "title": "Workspace",
               "type": "string"
            },
            "properties": {
               "title": "Properties",
               "type": "object",
               "additionalProperties": {
                  "type": "string"
               }
            }
         },
         "required": [
            "id"
         ]
      },
      "ApiError": {
         "title": "ApiError",
         "description": "Represents the standard error structure for SystemLink API␣
↪responses.",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "type": "string"
```

(continues on next page)

(continued from previous page)

```
            },
            "code": {
               "title": "Code",
               "type": "integer"
            },
            "message": {
               "title": "Message",
               "type": "string"
            },
            "args": {
               "title": "Args",
               "default": [],
               "type": "array",
               "items": {
                  "type": "string"
               }
            },
            "resourceType": {
               "title": "Resourcetype",
               "type": "string"
            },
            "resourceId": {
               "title": "Resourceid",
               "type": "string"
            },
            "innerErrors": {
               "title": "Innererrors",
               "default": [],
               "type": "array",
               "items": {
                  "$ref": "#/definitions/ApiError"
               }
            }
         }
      }
   }
}
```

Fields

- *error*
- *failed_modifications*
- *modified_table_ids*

field error: *ApiError* [Required]

The error that occurred when modifying the tables.

field failed_modifications: List[*TableMetadataModification*] [Required]

The requested modifications that could not be applied.

field modified_table_ids: List[str] [Required]

The IDs of the tables that were successfully modified.

pydantic model nisystemlink.clients.dataframe.models.**ModifyTablesRequest**

Contains one or more table modifications to apply.


```
{
   "title": "ModifyTablesRequest",
   "description": "Contains one or more table modifications to apply.",
   "type": "object",
   "properties": {
      "tables": {
         "title": "Tables",
         "type": "array",
         "items": {
            "$ref": "#/definitions/TableMetadataModification"
         }
      },
      "replace": {
         "title": "Replace",
         "type": "boolean"
      }
   },
   "required": [
      "tables"
   ],
   "definitions": {
      "TableMetadataModification": {
         "title": "TableMetadataModification",
         "description": "Contains the metadata properties to modify. Values not␣
↪included in the\nrequest or included with a ``None`` value will remain unchanged.
↪",
         "type": "object",
         "properties": {
            "id": {
               "title": "Id",
               "type": "string"
            },
            "metadataRevision": {
               "title": "Metadatarevision",
               "type": "integer"
            },
            "name": {
               "title": "Name",
               "type": "string"
            },
            "workspace": {
               "title": "Workspace",
               "type": "string"
            },
            "properties": {
               "title": "Properties",
               "type": "object",
               "additionalProperties": {
                  "type": "string"
               }
```


(continues on next page)

(continued from previous page)

```
            }
         },
         "required": [
            "id"
         ]
      }
   }
}
```

Fields

- *replace*
- *tables*

field replace: Optional[bool] = None

When true, existing properties are replaced instead of merged.

field tables: List[*TableMetadataModification*] [Required]

The table modifications to apply. Each table may only appear once in the list.

pydantic model nisystemlink.clients.dataframe.models.**Operation**

Represents an operation that can be performed on a data frame.

```
{
   "title": "Operation",
   "description": "Represents an operation that can be performed on a data frame.",
   "type": "object",
   "properties": {
      "available": {
         "title": "Available",
         "type": "boolean"
      },
      "version": {
         "title": "Version",
         "type": "integer"
      }
   },
   "required": [
      "available",
      "version"
   ]
}
```

Fields

- *available*
- *version*

field available: bool [Required]

Whether or not the operation is available to the caller (e.g. due to permissions).

field version: int [Required]

The version of the available operation.

pydantic model nisystemlink.clients.dataframe.models.**OperationsV1**

The operations available in the routes provided by the v1 HTTP API.


```
{
   "title": "OperationsV1",
   "description": "The operations available in the routes provided by the v1 HTTP␣
↪API.",
   "type": "object",
   "properties": {
      "createTables": {
         "$ref": "#/definitions/Operation"
      },
      "deleteTables": {
         "$ref": "#/definitions/Operation"
      },
      "modifyMetadata": {
         "$ref": "#/definitions/Operation"
      },
      "listTables": {
         "$ref": "#/definitions/Operation"
      },
      "readData": {
         "$ref": "#/definitions/Operation"
      },
      "writeData": {
         "$ref": "#/definitions/Operation"
      }
   },
   "required": [
      "createTables",
      "deleteTables",
      "modifyMetadata",
      "listTables",
      "readData",
      "writeData"
   ],
   "definitions": {
      "Operation": {
         "title": "Operation",
         "description": "Represents an operation that can be performed on a data␣
↪frame.",
         "type": "object",
         "properties": {
            "available": {
               "title": "Available",
               "type": "boolean"
            },
            "version": {
               "title": "Version",
               "type": "integer"
```


(continues on next page)

(continued from previous page)

```
            }
         },
         "required": [
            "available",
            "version"
         ]
      }
   }
}
```

Fields

- *create_tables*
- *delete_tables*
- *list_tables*
- *modify_metadata*
- *read_data*
- *write_data*

field create_tables: *Operation* **[Required]**

The ability to create new data tables.

field delete_tables: *Operation* **[Required]**

The ability to delete tables and all of their data.

field list_tables: *Operation* **[Required]**

The ability to locate and read metadata for tables.

field modify_metadata: *Operation* **[Required]**

The ability to modify metadata for tables.

field read_data: *Operation* **[Required]**

The ability to query and read data from tables.

field write_data: *Operation* **[Required]**

The ability to append rows of data to tables.

pydantic model nisystemlink.clients.dataframe.models.**PagedTableRows**

Contains the result of a query for rows of data.

```
{
   "title": "PagedTableRows",
   "description": "Contains the result of a query for rows of data.",
   "type": "object",
   "properties": {
      "continuationToken": {
         "title": "Continuationtoken",
         "type": "string"
      },
      "frame": {
         "$ref": "#/definitions/DataFrame"
```

(continues on next page)

(continued from previous page)

```
      },
      "totalRowCount": {
         "title": "Totalrowcount",
         "type": "integer"
      }
   },
   "required": [
      "frame",
      "totalRowCount"
   ],
   "definitions": {
      "DataFrame": {
         "title": "DataFrame",
         "description": "Data read from or to be written to a table.\n\nValues may␣
↪be ``None`` (if the column is of type ``NULLABLE``) or encoded as\na string in a␣
↪format according to each column's datatype:\n\n* BOOL: One of ``\"true\"`` or ``\
↪"false\"``, case-insensitive.\n* INT32: Any integer number in the range [-
↪2147483648, 2147483647],\n  surrounded by quotes.\n* INT64: Any integer number in␣
↪the range [-9223372036854775808,\n  9223372036854775807], surrounded by quotes.\
↪n* FLOAT32: A decimal number using a period for the decimal point, optionally\n ␣
↪in scientific notation, in the range [-3.40282347E+38, 3.40282347E+38],\n ␣
↪surrounded by quotes. Not all values within the range can be represented\n  with␣
↪32 bits. To preserve the exact binary encoding of the value when\n  converting to␣
↪a string, clients should serialize 9 digits after the\n  decimal. Instead of a␣
↪number, the value may be ``\"NaN\"`` (not a number),\n  ``\"Infinity\"``␣
↪(positive infinity), or ``\"-Infinity\"`` (negative\n  infinity), case-sensitive.\
↪n* FLOAT64: A decimal number using a period for the decimal point, optionally\n ␣
↪in scientific notation, in the range [-1.7976931348623157E+308,\n  1.
↪7976931348623157E+308], surrounded by quotes. Not all values within the\n  range␣
↪can be represented with 64 bits. To preserve the exact binary\n  encoding of the␣
↪value when converting to a string, clients should\n  serialize 17 digits after␣
↪the decimal. Instead of a number, the value may\n  be ``\"NaN\"`` (not a number),␣
↪``\"Infinity\"`` (positive infinity), or\n  ``\"-Infinity\"`` (negative infinity),
↪ case-sensitive.\n* STRING: Any quoted string.\n* TIMESTAMP: A date and time with␣
↪millisecond precision in ISO-8601 format\n  and time zone. For example: ``\"2022-
↪08-19T16:17:30.123Z\"``. If a time zone\n  is not provided, UTC is assumed. If a␣
↪time zone other than UTC is\n  provided, the value will be converted to UTC. If␣
↪more than three digits of\n  fractional seconds are provided, the time will be␣
↪truncated to three\n  digits (i.e. milliseconds).\n\nThe format is the same as a␣
↪serialized Pandas DataFrame with orient=\"split\"\nand index=False. See\nhttps://
↪pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_json.html.\n\nWhen␣
↪providing a DataFrame for appending rows, any of the table's columns\nnot␣
↪specified will receive a value of ``None``. If any such columns aren't\nnullable,␣
↪an error will be returned. If the entire columns property is left\nout, each row␣
↪is assumed to contain all columns in the order specified when\nthe table was␣
↪created.",
         "type": "object",
         "properties": {
            "columns": {
               "title": "Columns",
               "type": "array",
               "items": {
```

(continues on next page)

(continued from previous page)


```
                    "type": "string"
                }
            },
            "data": {
                "title": "Data",
                "type": "array",
                "items": {
                    "type": "array",
                    "items": {
                        "type": "string"
                    }
                }
            }
        },
        "required": [
            "data"
        ]
    }
}
```


Fields

- *frame*
- *total_row_count*

field frame: *DataFrame* [Required]

The data frame containing the rows of data.

field total_row_count: int [Required]

The total number of rows matched by the query across all pages of results.

pydantic model nisystemlink.clients.dataframe.models.**PagedTables**

The response for a table query containing the matched tables.


```
{
    "title": "PagedTables",
    "description": "The response for a table query containing the matched tables.",
    "type": "object",
    "properties": {
        "continuationToken": {
            "title": "Continuationtoken",
            "type": "string"
        },
        "tables": {
            "title": "Tables",
            "type": "array",
            "items": {
                "$ref": "#/definitions/TableMetadata"
            }
        }
    },
```


(continues on next page)

(continued from previous page)

```
  "required": [
     "tables"
  ],
  "definitions": {
     "DataType": {
        "title": "DataType",
        "description": "Represents the different data types for a table column.",
        "enum": [
           "BOOL",
           "FLOAT32",
           "FLOAT64",
           "INT32",
           "INT64",
           "STRING",
           "TIMESTAMP"
        ],
        "type": "string"
     },
     "ColumnType": {
        "title": "ColumnType",
        "description": "Represents the different column types for a table column.",
        "enum": [
           "NORMAL",
           "INDEX",
           "NULLABLE"
        ],
        "type": "string"
     },
     "Column": {
        "title": "Column",
        "description": "Defines a single column in a table.",
        "type": "object",
        "properties": {
           "name": {
              "title": "Name",
              "type": "string"
           },
           "dataType": {
              "$ref": "#/definitions/DataType"
           },
           "columnType": {
              "default": "NORMAL",
              "allOf": [
                 {
                    "$ref": "#/definitions/ColumnType"
                 }
              ]
           },
           "properties": {
              "title": "Properties",
              "type": "object",
              "additionalProperties": {
```

(continues on next page)

(continued from previous page)

```
                "type": "string"
            }
         }
      },
      "required": [
         "name",
         "dataType"
      ]
   },
   "TableMetadata": {
      "title": "TableMetadata",
      "description": "Contains information about a table, including its␣
↪properties and column definitions.",
      "type": "object",
      "properties": {
         "columns": {
            "title": "Columns",
            "type": "array",
            "items": {
               "$ref": "#/definitions/Column"
            }
         },
         "createdAt": {
            "title": "Createdat",
            "type": "string",
            "format": "date-time"
         },
         "id": {
            "title": "Id",
            "type": "string"
         },
         "metadataModifiedAt": {
            "title": "Metadatamodifiedat",
            "type": "string",
            "format": "date-time"
         },
         "metadataRevision": {
            "title": "Metadatarevision",
            "type": "integer"
         },
         "name": {
            "title": "Name",
            "type": "string"
         },
         "properties": {
            "title": "Properties",
            "type": "object",
            "additionalProperties": {
               "type": "string"
            }
         },
         "rowCount": {
```

(continues on next page)

(continued from previous page)


```
                "title": "Rowcount",
                "type": "integer"
            },
            "rowsModifiedAt": {
                "title": "Rowsmodifiedat",
                "type": "string",
                "format": "date-time"
            },
            "supportsAppend": {
                "title": "Supportsappend",
                "type": "boolean"
            },
            "workspace": {
                "title": "Workspace",
                "type": "string"
            }
        },
        "required": [
            "columns",
            "createdAt",
            "id",
            "metadataModifiedAt",
            "metadataRevision",
            "name",
            "properties",
            "rowCount",
            "rowsModifiedAt",
            "supportsAppend",
            "workspace"
        ]
    }
}
```


Fields

- *tables*

field tables: List[*TableMetadata*] [Required]

The list of tables returned by the query.

pydantic model nisystemlink.clients.dataframe.models.**QueryDecimatedDataRequest**

Specifies the columns, filters and decimation parameters for a query.


```
{
    "title": "QueryDecimatedDataRequest",
    "description": "Specifies the columns, filters and decimation parameters for a␣
↪query.",
    "type": "object",
    "properties": {
        "columns": {
            "title": "Columns",
```


(continues on next page)

(continued from previous page)

```
            "type": "array",
            "items": {
               "type": "string"
            }
         },
         "filters": {
            "title": "Filters",
            "type": "array",
            "items": {
               "$ref": "#/definitions/ColumnFilter"
            }
         },
         "decimation": {
            "$ref": "#/definitions/DecimationOptions"
         }
      },
      "definitions": {
         "FilterOperation": {
            "title": "FilterOperation",
            "description": "Represents the different operations that can be used in a␣
↪filter.",
            "enum": [
               "EQUALS",
               "NOT_EQUALS",
               "LESS_THAN",
               "LESS_THAN_EQUALS",
               "GREATER_THAN",
               "GREATER_THAN_EQUALS",
               "CONTAINS",
               "NOT_CONTAINS"
            ],
            "type": "string"
         },
         "ColumnFilter": {
            "title": "ColumnFilter",
            "description": "A filter to apply to the table data.",
            "type": "object",
            "properties": {
               "column": {
                  "title": "Column",
                  "type": "string"
               },
               "operation": {
                  "$ref": "#/definitions/FilterOperation"
               },
               "value": {
                  "title": "Value",
                  "type": "string"
               }
            },
            "required": [
               "column",
```

(continues on next page)

(continued from previous page)

```
            "operation"
         ]
      },
      "DecimationMethod": {
         "title": "DecimationMethod",
         "description": "Represents the different methods that can be used to␣
↪decimate data.",
         "enum": [
            "LOSSY",
            "MAX_MIN",
            "ENTRY_EXIT"
         ],
         "type": "string"
      },
      "DecimationOptions": {
         "title": "DecimationOptions",
         "description": "Contains the parameters to use for data decimation.",
         "type": "object",
         "properties": {
            "xColumn": {
               "title": "Xcolumn",
               "type": "string"
            },
            "yColumns": {
               "title": "Ycolumns",
               "type": "array",
               "items": {
                  "type": "string"
               }
            },
            "intervals": {
               "title": "Intervals",
               "type": "integer"
            },
            "method": {
               "$ref": "#/definitions/DecimationMethod"
            }
         }
      }
   }
}
```

Fields

- *decimation*

field decimation: Optional[*DecimationOptions*] = None

The decimation options to use when querying data. If not specified, the default is to use *DecimationMethod.Lossy* with 1000 intervals over the table's index column.

pydantic model nisystemlink.clients.dataframe.models.**QueryTableDataRequest**

Contains the filtering and sorting options to use when querying table data.

```
{
   "title": "QueryTableDataRequest",
   "description": "Contains the filtering and sorting options to use when querying␣
↪table data.",
   "type": "object",
   "properties": {
      "continuationToken": {
         "title": "Continuationtoken",
         "type": "string"
      },
      "columns": {
         "title": "Columns",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "filters": {
         "title": "Filters",
         "type": "array",
         "items": {
            "$ref": "#/definitions/ColumnFilter"
         }
      },
      "orderBy": {
         "title": "Orderby",
         "type": "array",
         "items": {
            "$ref": "#/definitions/ColumnOrderBy"
         }
      },
      "take": {
         "title": "Take",
         "type": "integer"
      }
   },
   "definitions": {
      "FilterOperation": {
         "title": "FilterOperation",
         "description": "Represents the different operations that can be used in a␣
↪filter.",
         "enum": [
            "EQUALS",
            "NOT_EQUALS",
            "LESS_THAN",
            "LESS_THAN_EQUALS",
            "GREATER_THAN",
            "GREATER_THAN_EQUALS",
            "CONTAINS",
            "NOT_CONTAINS"
         ],
         "type": "string"
      },
```


(continues on next page)

(continued from previous page)


```
      "ColumnFilter": {
         "title": "ColumnFilter",
         "description": "A filter to apply to the table data.",
         "type": "object",
         "properties": {
            "column": {
               "title": "Column",
               "type": "string"
            },
            "operation": {
               "$ref": "#/definitions/FilterOperation"
            },
            "value": {
               "title": "Value",
               "type": "string"
            }
         },
         "required": [
            "column",
            "operation"
         ]
      },
      "ColumnOrderBy": {
         "title": "ColumnOrderBy",
         "description": "Specifies a column to order by and the ordering direction.
↪",
         "type": "object",
         "properties": {
            "column": {
               "title": "Column",
               "type": "string"
            },
            "descending": {
               "title": "Descending",
               "type": "boolean"
            }
         },
         "required": [
            "column"
         ]
      }
   }
}
```


Fields

- *order_by*
- *take*

field order_by: Optional[List[*ColumnOrderBy*]] = None

A list of columns to order the results by. Multiple columns may be specified to order rows that have the same value for prior columns. The columns used for sorting do not need to be included in the columns

list, in which case they are not returned. If `order_by` is not specified, then the order in which results are returned is undefined.

field take: Optional[int] = None

Limits the returned list to the specified number of results.

pydantic model nisystemlink.clients.dataframe.models.**QueryTablesRequest**

Request parameters for querying tables.


```
{
   "title": "QueryTablesRequest",
   "description": "Request parameters for querying tables.",
   "type": "object",
   "properties": {
      "continuationToken": {
         "title": "Continuationtoken",
         "type": "string"
      },
      "filter": {
         "title": "Filter",
         "type": "string"
      },
      "substitutions": {
         "title": "Substitutions",
         "type": "array",
         "items": {
            "anyOf": [
               {
                  "type": "integer"
               },
               {
                  "type": "boolean"
               },
               {
                  "type": "string"
               }
            ]
         }
      },
      "referenceTime": {
         "title": "Referencetime",
         "type": "string",
         "format": "date-time"
      },
      "take": {
         "title": "Take",
         "type": "integer"
      },
      "orderBy": {
         "title": "Orderby",
         "enum": [
            "CREATED_AT",
            "METADATA_MODIFIED_AT",
            "NAME",
```


(continues on next page)

(continued from previous page)

```
            "NUMBER_OF_ROWS",
            "ROWS_MODIFIED_AT"
         ],
         "type": "string"
      },
      "orderByDescending": {
         "title": "Orderbydescending",
         "type": "boolean"
      }
   },
   "required": [
      "filter"
   ]
}
```

Fields

- *filter*
- *order_by*
- *order_by_descending*
- *reference_time*
- *substitutions*
- *take*

field filter: str [Required]

The table query filter in Dynamic LINQ format.

Allowed properties in the filter are:

- `createdAt`: DateTime the table was created
- `createdWithin`: TimeSpan in which the table was created
- `id`: String value uniquely identifying the table
- `name`: String name for the table
- `metadataModifiedAt`: DateTime the table's metadata was last modified
- `metadataModifiedWithin`: TimeSpan in which the table's metadata was last modified
- `properties`: Dictionary with string keys and values representing table metadata
- `rowsModifiedAt`: DateTime rows were last appended to the table
- `rowsModifiedWithin`: TimeSpan within rows were last appended to the table
- `rowCount`: Int32 number of rows in the table
- `supportsAppend`: Boolean indicating whether or not the table supports appending additional rows of data
- `workspace`: String value ID of the workspace the table belongs to
- `workspaceName`: String value name of the workspace the table belongs to

Allowed constants in the filter are:

- `RelativeTime.CurrentDay`: TimeSpan representing the elapsed time between now and the start of the current day
- `RelativeTime.CurrentWeek`: TimeSpan representing the elapsed time between now and the start of the current week
- `RelativeTime.CurrentMonth`: TimeSpan representing the elapsed time between now and the start of the current month
- `RelativeTime.CurrentYear`: TimeSpan representing the elapsed time between now and the start of the current year

field order_by: Optional[Literal['CREATED_AT', 'METADATA_MODIFIED_AT', 'NAME', 'NUMBER_OF_ROWS', 'ROWS_MODIFIED_AT']] = None

The sort order of the returned list of tables.

field order_by_descending: Optional[bool] = None

Whether to sort descending instead of ascending.

The elements in the list are sorted ascending by default. If the orderByDescending parameter is specified, the elements in the list are sorted based on it's value. The orderByDescending value must be a boolean string. The elements in the list are sorted ascending if false and descending if true.

field reference_time: Optional[datetime] = None

The date and time to use as the reference point for *RelativeTime* filters, including time zone information. Defaults to the time on the server in UTC.

field substitutions: Optional[List[Union[StrictInt, StrictBool, str, None]]] = None

Make substitutions in the query filter expression.

Substitutions for the query expression are indicated by non-negative integers that are prefixed with the `@` symbol. Each substitution in the given expression will be replaced by the element at the corresponding index (zero-based) in this list. For example, `@0` in the filter expression will be replaced with the element at the zeroth index of the substitutions list.

field take: Optional[int] = None

Limits the returned list to the specified number of results.

pydantic model `nisystemlink.clients.dataframe.models.`**TableMetadata**

Contains information about a table, including its properties and column definitions.


```
{
   "title": "TableMetadata",
   "description": "Contains information about a table, including its properties and␣
↪column definitions.",
   "type": "object",
   "properties": {
      "columns": {
         "title": "Columns",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Column"
         }
      },
      "createdAt": {
         "title": "Createdat",
         "type": "string",
```


(continues on next page)

(continued from previous page)


```
            "format": "date-time"
        },
        "id": {
            "title": "Id",
            "type": "string"
        },
        "metadataModifiedAt": {
            "title": "Metadatamodifiedat",
            "type": "string",
            "format": "date-time"
        },
        "metadataRevision": {
            "title": "Metadatarevision",
            "type": "integer"
        },
        "name": {
            "title": "Name",
            "type": "string"
        },
        "properties": {
            "title": "Properties",
            "type": "object",
            "additionalProperties": {
                "type": "string"
            }
        },
        "rowCount": {
            "title": "Rowcount",
            "type": "integer"
        },
        "rowsModifiedAt": {
            "title": "Rowsmodifiedat",
            "type": "string",
            "format": "date-time"
        },
        "supportsAppend": {
            "title": "Supportsappend",
            "type": "boolean"
        },
        "workspace": {
            "title": "Workspace",
            "type": "string"
        }
    },
    "required": [
        "columns",
        "createdAt",
        "id",
        "metadataModifiedAt",
        "metadataRevision",
        "name",
        "properties",
```


(continues on next page)

(continued from previous page)

```
      "rowCount",
      "rowsModifiedAt",
      "supportsAppend",
      "workspace"
   ],
   "definitions": {
      "DataType": {
         "title": "DataType",
         "description": "Represents the different data types for a table column.",
         "enum": [
            "BOOL",
            "FLOAT32",
            "FLOAT64",
            "INT32",
            "INT64",
            "STRING",
            "TIMESTAMP"
         ],
         "type": "string"
      },
      "ColumnType": {
         "title": "ColumnType",
         "description": "Represents the different column types for a table column.",
         "enum": [
            "NORMAL",
            "INDEX",
            "NULLABLE"
         ],
         "type": "string"
      },
      "Column": {
         "title": "Column",
         "description": "Defines a single column in a table.",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "type": "string"
            },
            "dataType": {
               "$ref": "#/definitions/DataType"
            },
            "columnType": {
               "default": "NORMAL",
               "allOf": [
                  {
                     "$ref": "#/definitions/ColumnType"
                  }
               ]
            },
            "properties": {
               "title": "Properties",
```

(continues on next page)

(continued from previous page)

```
                "type": "object",
                "additionalProperties": {
                   "type": "string"
                }
             }
          },
          "required": [
             "name",
             "dataType"
          ]
       }
    }
}
```

Fields

- *columns*
- *created_at*
- *id*
- *metadata_modified_at*
- *metadata_revision*
- *name*
- *properties*
- *row_count*
- *rows_modified_at*
- *supports_append*
- *workspace*

field columns: List[*Column*] [Required]

The list of columns in the table.

field created_at: datetime [Required]

The date and time the table was created.

field id: str [Required]

The table's unique identifier.

field metadata_modified_at: datetime [Required]

The date and time the table's metadata was last modified.

field metadata_revision: int [Required]

The table's metadata revision number, incremented each time the metadata is modified.

field name: str [Required]

The name associated with the table.

field properties: Dict[str, str] [Required]

User-defined properties associated with the table.

field row_count: int [Required]

The number of rows in the table.

field rows_modified_at: datetime [Required]

The date and time the table's data was last modified.

field supports_append: bool [Required]

Whether the table supports appending additional rows of data.

field workspace: str [Required]

The workspace the table belongs to.

pydantic model nisystemlink.clients.dataframe.models.**TableMetadataModification**

Contains the metadata properties to modify. Values not included in the request or included with a None value will remain unchanged.


```
{
   "title": "TableMetadataModification",
   "description": "Contains the metadata properties to modify. Values not included␣
↪in the\nrequest or included with a ``None`` value will remain unchanged.",
   "type": "object",
   "properties": {
      "id": {
         "title": "Id",
         "type": "string"
      },
      "metadataRevision": {
         "title": "Metadatarevision",
         "type": "integer"
      },
      "name": {
         "title": "Name",
         "type": "string"
      },
      "workspace": {
         "title": "Workspace",
         "type": "string"
      },
      "properties": {
         "title": "Properties",
         "type": "object",
         "additionalProperties": {
            "type": "string"
         }
      }
   },
   "required": [
      "id"
   ]
}
```


Fields

- *id*
- *metadata_revision*

- *name*
- *properties*
- *workspace*

field id: str [Required]

The ID of the table to modify.

field metadata_revision: Optional[int] = None

When specified, this is an integer that must match the last known revision number of the table, incremented by one. If it doesn't match the current `metadataRevision` incremented by one at the time of execution, the modify request will be rejected with a conflict error. This is used to ensure that changes to this table's metadata are based on a known, previous state.

field name: Optional[str] = None

The new name of the table.

field properties: Optional[Dict[str, Optional[str]]] = None

The properties to modify. A map of key value properties containing the metadata to be added or modified. Setting a property value to `None` will delete the property. Existing properties not included in the map are unaffected unless replace is true in the top-level request object.

field workspace: Optional[str] = None

The new workspace for the table. Changing the workspace requires permission to delete the table in its current workspace and permission to create the table in its new workspace.

nisystemlink.clients.dataframe.models.**TableMetdataModification**

alias of *TableMetadataModification*

pydantic model nisystemlink.clients.dataframe.models.**TableRows**

Contains the result of a query for rows of decimated data.


```
{
   "title": "TableRows",
   "description": "Contains the result of a query for rows of decimated data.",
   "type": "object",
   "properties": {
      "frame": {
         "$ref": "#/definitions/DataFrame"
      }
   },
   "required": [
      "frame"
   ],
   "definitions": {
      "DataFrame": {
         "title": "DataFrame",
         "description": "Data read from or to be written to a table.\n\nValues may␣
↪be ``None`` (if the column is of type ``NULLABLE``) or encoded as\na string in a␣
↪format according to each column's datatype:\n\n* BOOL: One of ``\"true\"`` or ``\
↪"false\"``, case-insensitive.\n* INT32: Any integer number in the range [-
↪2147483648, 2147483647],\n  surrounded by quotes.\n* INT64: Any integer number in␣
↪the range [-9223372036854775808,\n  9223372036854775807], surrounded by quotes.\
↪n* FLOAT32: A decimal number using a period for the decimal point, optionally\n ␣
↪in scientific notation, in the range [-3.40282347E+38, 3.40282347E+38],\n ␣
```


(continues on next page)

(continued from previous page)

```
↪surrounded by quotes. Not all values within the range can be represented\n  with␣
↪32 bits. To preserve the exact binary encoding of the value when\n  converting to␣
↪a string, clients should serialize 9 digits after the\n  decimal. Instead of a␣
↪number, the value may be ``\"NaN\"`` (not a number),\n  ``\"Infinity\"``␣
↪(positive infinity), or ``\"-Infinity\"`` (negative\n  infinity), case-sensitive.\
↪n* FLOAT64: A decimal number using a period for the decimal point, optionally\n ␣
↪in scientific notation, in the range [-1.7976931348623157E+308,\n  1.
↪7976931348623157E+308], surrounded by quotes. Not all values within the\n  range␣
↪can be represented with 64 bits. To preserve the exact binary\n  encoding of the␣
↪value when converting to a string, clients should\n  serialize 17 digits after␣
↪the decimal. Instead of a number, the value may\n  be ``\"NaN\"`` (not a number),␣
↪``\"Infinity\"`` (positive infinity), or\n  ``\"-Infinity\"`` (negative infinity),
↪ case-sensitive.\n* STRING: Any quoted string.\n* TIMESTAMP: A date and time with␣
↪millisecond precision in ISO-8601 format\n  and time zone. For example: ``\"2022-
↪08-19T16:17:30.123Z\"``. If a time zone\n  is not provided, UTC is assumed. If a␣
↪time zone other than UTC is\n  provided, the value will be converted to UTC. If␣
↪more than three digits of\n  fractional seconds are provided, the time will be␣
↪truncated to three\n  digits (i.e. milliseconds).\n\nThe format is the same as a␣
↪serialized Pandas DataFrame with orient=\"split\"\nand index=False. See\nhttps://
↪pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_json.html.\n\nWhen␣
↪providing a DataFrame for appending rows, any of the table's columns\nnot␣
↪specified will receive a value of ``None``. If any such columns aren't\nnullable,␣
↪an error will be returned. If the entire columns property is left\nout, each row␣
↪is assumed to contain all columns in the order specified when\nthe table was␣
↪created.",
        "type": "object",
        "properties": {
          "columns": {
            "title": "Columns",
            "type": "array",
            "items": {
              "type": "string"
            }
          },
          "data": {
            "title": "Data",
            "type": "array",
            "items": {
              "type": "array",
              "items": {
                "type": "string"
              }
            }
          }
        },
        "required": [
          "data"
        ]
      }
    }
}
```

Fields

- *frame*

field frame: *DataFrame* [Required]

The data frame containing the rows of data.

1.2.4 nisystemlink.clients.spec

class nisystemlink.clients.spec.**SpecClient**(*configuration*)

__init__(*configuration*)

Initialize an instance.

Parameters

- **configuration** (Optional[*nisystemlink.clients.core.HttpConfiguration*]) – Defines the web server to connect to and information about how to connect.
- **base_path** – The base path for all API calls.

api_info()

Get information about available API operations.

Return type
nisystemlink.clients.spec.models.V1Operations

Returns
Information about available API operations.

Raises
ApiException – if unable to communicate with the DataFrame Service.

create_specs(*specs*)

Creates one or more specifications.

Parameters
specs (*nisystemlink.clients.spec.models.CreateSpecificationsRequest*) – A list of specifications to create.

Return type
nisystemlink.clients.spec.models.CreateSpecificationsPartialSuccess

Returns
A list of specs that were successfully created and ones that failed to be created.

Raises

- ***ApiException*** – if unable to communicate with the */nispec* service or if there are
- **invalid arguments.** –

delete_specs(*ids*)

Deletes one or more specifications by global id.

Parameters

- **ids** (List[str]) – a list of specification ids. Note that these are the global ids and not the
- **workspace.** (*specId that is local to a product and*) –

Return type
Optional[*nisystemlink.clients.spec.models.DeleteSpecificationsPartialSuccess*]

Returns
None if all deletes succeed otherwise a list of which ids failed and which succeeded.

Raises

- ***ApiException*** – if unable to communicate with the *nispec* service or if there are invalid
- **arguments.** –

query_specs(*query*)

Queries for specs that match the filters.

Parameters
query (*nisystemlink.clients.spec.models.QuerySpecificationsRequest*) – The query contains a product id as well as a filter for specs under that product.

Return type
nisystemlink.clients.spec.models.QuerySpecifications

Returns
A list of specifications that match the filter.

update_specs(*specs*)

Updates one or more specifications.

Update requires that the version field matches the version being updated from.

Parameters

- **specs** (*nisystemlink.clients.spec.models.UpdateSpecificationsRequest*) – a list of specifications that are to be updated. Must include the global id and
- **server.** (*each spec being updated must match the version currently on the*) –

Return type
Optional[*nisystemlink.clients.spec.models.UpdateSpecificationsPartialSuccess*]

Returns
A list of specs that were successfully updated and a list of ones that were not along with error messages for updates that failed.

pydantic model nisystemlink.clients.spec.models.**Condition**

A single condition.


```
{
   "title": "Condition",
   "description": "A single condition.",
   "type": "object",
   "properties": {
      "name": {
         "title": "Name",
         "type": "string"
      },
      "value": {
         "title": "Value",
         "anyOf": [
```


(continues on next page)

(continued from previous page)

```
            {
               "$ref": "#/definitions/NumericConditionValue"
            },
            {
               "$ref": "#/definitions/StringConditionValue"
            }
         ]
      }
   },
   "definitions": {
      "ConditionType": {
         "title": "ConditionType",
         "description": "Conditions are either numeric or string type.",
         "enum": [
            "NUMERIC",
            "STRING"
         ]
      },
      "ConditionRange": {
         "title": "ConditionRange",
         "description": "Specifies the range of values that the condition must␣
↪cover.",
         "type": "object",
         "properties": {
            "min": {
               "title": "Min",
               "type": "number"
            },
            "max": {
               "title": "Max",
               "type": "number"
            },
            "step": {
               "title": "Step",
               "type": "number"
            }
         }
      },
      "NumericConditionValue": {
         "title": "NumericConditionValue",
         "description": "A numeric condition.\n\nNumeric conditions can contain a␣
↪combination of ranges and discrete lists.",
         "type": "object",
         "properties": {
            "conditionType": {
               "$ref": "#/definitions/ConditionType"
            },
            "range": {
               "title": "Range",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/ConditionRange"
```

(continues on next page)

(continued from previous page)


```
                }
            },
            "discrete": {
                "title": "Discrete",
                "type": "array",
                "items": {
                    "type": "number"
                }
            },
            "unit": {
                "title": "Unit",
                "type": "string"
            }
        },
        "required": [
            "conditionType"
        ]
    },
    "StringConditionValue": {
        "title": "StringConditionValue",
        "description": "A string condition.\n\nString conditions may only contain␣
↪discrete lists of values.",
        "type": "object",
        "properties": {
            "conditionType": {
                "$ref": "#/definitions/ConditionType"
            },
            "discrete": {
                "title": "Discrete",
                "type": "array",
                "items": {
                    "type": "string"
                }
            }
        },
        "required": [
            "conditionType"
        ]
    }
  }
}
```


Fields

- *name*
- *value*

field name: Optional[str] = None

Name of the condition.

field value: Union[*NumericConditionValue*, *StringConditionValue*, None] = None

Value of the condition.

pydantic model nisystemlink.clients.spec.models.**ConditionRange**

Specifies the range of values that the condition must cover.


```
{
   "title": "ConditionRange",
   "description": "Specifies the range of values that the condition must cover.",
   "type": "object",
   "properties": {
      "min": {
         "title": "Min",
         "type": "number"
      },
      "max": {
         "title": "Max",
         "type": "number"
      },
      "step": {
         "title": "Step",
         "type": "number"
      }
   }
}
```


Fields

- *max*
- *min*
- *step*

field max: Optional[float] = None

Maximum value of the condition range.

field min: Optional[float] = None

Minimum value of the condition range.

field step: Optional[float] = None

Step value of the condition range.

class nisystemlink.clients.spec.models.**ConditionType**(*value*)

Conditions are either numeric or string type.

NUMERIC = 'NUMERIC'

Numeric condition.

STRING = 'STRING'

String condition.

pydantic model nisystemlink.clients.spec.models.**CreateSpecificationsPartialSuccess**

When some specs can not be created, this contains the list that was and was not created.


```
{
   "title": "CreateSpecificationsPartialSuccess",
   "description": "When some specs can not be created, this contains the list that␣
↪was and was not created.",
```


(continues on next page)

(continued from previous page)

```
   "type": "object",
   "properties": {
      "createdSpecs": {
         "title": "Createdspecs",
         "type": "array",
         "items": {
            "$ref": "#/definitions/CreatedSpecification"
         }
      },
      "failedSpecs": {
         "title": "Failedspecs",
         "type": "array",
         "items": {
            "$ref": "#/definitions/SpecificationDefinition"
         }
      },
      "error": {
         "$ref": "#/definitions/ApiError"
      }
   },
   "definitions": {
      "CreatedSpecification": {
         "title": "CreatedSpecification",
         "description": "A specification successfully created on the server.",
         "type": "object",
         "properties": {
            "createdAt": {
               "title": "Createdat",
               "type": "string",
               "format": "date-time"
            },
            "createdBy": {
               "title": "Createdby",
               "type": "string"
            },
            "productId": {
               "title": "Productid",
               "type": "string"
            },
            "specId": {
               "title": "Specid",
               "type": "string"
            },
            "workspace": {
               "title": "Workspace",
               "type": "string"
            },
            "id": {
               "title": "Id",
               "type": "string"
            },
            "version": {
```

(continues on next page)

(continued from previous page)

```
                "title": "Version",
                "type": "integer"
            }
        },
        "required": [
            "productId",
            "specId",
            "id",
            "version"
        ]
    },
    "SpecificationType": {
        "title": "SpecificationType",
        "description": "The overall type of the specification.",
        "enum": [
            "PARAMETRIC",
            "FUNCTIONAL"
        ]
    },
    "SpecificationLimit": {
        "title": "SpecificationLimit",
        "description": "A limit for a specification.\n\nThe limit is the value␣
↪that a measurement should be compared against during analysis to\ndetermine the␣
↪health or pass/fail status of that measurement.",
        "type": "object",
        "properties": {
            "min": {
                "title": "Min",
                "type": "number"
            },
            "typical": {
                "title": "Typical",
                "type": "number"
            },
            "max": {
                "title": "Max",
                "type": "number"
            }
        }
    },
    "ConditionType": {
        "title": "ConditionType",
        "description": "Conditions are either numeric or string type.",
        "enum": [
            "NUMERIC",
            "STRING"
        ]
    },
    "ConditionRange": {
        "title": "ConditionRange",
        "description": "Specifies the range of values that the condition must␣
↪cover.",
```

(continues on next page)

(continued from previous page)

```
            "type": "object",
            "properties": {
               "min": {
                  "title": "Min",
                  "type": "number"
               },
               "max": {
                  "title": "Max",
                  "type": "number"
               },
               "step": {
                  "title": "Step",
                  "type": "number"
               }
            }
         },
         "NumericConditionValue": {
            "title": "NumericConditionValue",
            "description": "A numeric condition.\n\nNumeric conditions can contain a␣
↪combination of ranges and discrete lists.",
            "type": "object",
            "properties": {
               "conditionType": {
                  "$ref": "#/definitions/ConditionType"
               },
               "range": {
                  "title": "Range",
                  "type": "array",
                  "items": {
                     "$ref": "#/definitions/ConditionRange"
                  }
               },
               "discrete": {
                  "title": "Discrete",
                  "type": "array",
                  "items": {
                     "type": "number"
                  }
               },
               "unit": {
                  "title": "Unit",
                  "type": "string"
               }
            },
            "required": [
               "conditionType"
            ]
         },
         "StringConditionValue": {
            "title": "StringConditionValue",
            "description": "A string condition.\n\nString conditions may only contain␣
↪discrete lists of values.",
```

(continues on next page)

(continued from previous page)

```
        "type": "object",
        "properties": {
          "conditionType": {
            "$ref": "#/definitions/ConditionType"
          },
          "discrete": {
            "title": "Discrete",
            "type": "array",
            "items": {
              "type": "string"
            }
          }
        },
        "required": [
          "conditionType"
        ]
      },
      "Condition": {
        "title": "Condition",
        "description": "A single condition.",
        "type": "object",
        "properties": {
          "name": {
            "title": "Name",
            "type": "string"
          },
          "value": {
            "title": "Value",
            "anyOf": [
              {
                "$ref": "#/definitions/NumericConditionValue"
              },
              {
                "$ref": "#/definitions/StringConditionValue"
              }
            ]
          }
        }
      },
      "SpecificationDefinition": {
        "title": "SpecificationDefinition",
        "description": "Base class for models that are serialized to and from JSON.
↪",
        "type": "object",
        "properties": {
          "productId": {
            "title": "Productid",
            "type": "string"
          },
          "specId": {
            "title": "Specid",
            "type": "string"
```

(continues on next page)

(continued from previous page)

```
            },
            "workspace": {
               "title": "Workspace",
               "type": "string"
            },
            "name": {
               "title": "Name",
               "type": "string"
            },
            "category": {
               "title": "Category",
               "type": "string"
            },
            "type": {
               "$ref": "#/definitions/SpecificationType"
            },
            "symbol": {
               "title": "Symbol",
               "type": "string"
            },
            "block": {
               "title": "Block",
               "type": "string"
            },
            "limit": {
               "$ref": "#/definitions/SpecificationLimit"
            },
            "unit": {
               "title": "Unit",
               "type": "string"
            },
            "conditions": {
               "title": "Conditions",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/Condition"
               }
            },
            "keywords": {
               "title": "Keywords",
               "type": "array",
               "items": {
                  "type": "string"
               }
            },
            "properties": {
               "title": "Properties",
               "type": "object",
               "additionalProperties": {
                  "type": "string"
               }
            }
```

(continues on next page)

(continued from previous page)

```
        },
        "required": [
           "productId",
           "specId",
           "type"
        ]
     },
     "ApiError": {
        "title": "ApiError",
        "description": "Represents the standard error structure for SystemLink API␣
↪responses.",
        "type": "object",
        "properties": {
           "name": {
              "title": "Name",
              "type": "string"
           },
           "code": {
              "title": "Code",
              "type": "integer"
           },
           "message": {
              "title": "Message",
              "type": "string"
           },
           "args": {
              "title": "Args",
              "default": [],
              "type": "array",
              "items": {
                 "type": "string"
              }
           },
           "resourceType": {
              "title": "Resourcetype",
              "type": "string"
           },
           "resourceId": {
              "title": "Resourceid",
              "type": "string"
           },
           "innerErrors": {
              "title": "Innererrors",
              "default": [],
              "type": "array",
              "items": {
                 "$ref": "#/definitions/ApiError"
              }
           }
        }
     }
  }
```

(continues on next page)

(continued from previous page)

```
}
```

Fields

- *created_specs*
- *error*
- *failed_specs*

field created_specs: Optional[List[*CreatedSpecification*]] = None

Information about the created specification(s)

field error: Optional[*ApiError*] = None

field failed_specs: Optional[List[*SpecificationDefinition*]] = None

List of specification requests that failed during creation.

pydantic model nisystemlink.clients.spec.models.**CreateSpecificationsRequest**

Create multiple specifications.

```
{
   "title": "CreateSpecificationsRequest",
   "description": "Create multiple specifications.",
   "type": "object",
   "properties": {
      "specs": {
         "title": "Specs",
         "type": "array",
         "items": {
            "$ref": "#/definitions/SpecificationDefinition"
         }
      }
   },
   "definitions": {
      "SpecificationType": {
         "title": "SpecificationType",
         "description": "The overall type of the specification.",
         "enum": [
            "PARAMETRIC",
            "FUNCTIONAL"
         ]
      },
      "SpecificationLimit": {
         "title": "SpecificationLimit",
         "description": "A limit for a specification.\n\nThe limit is the value␣
↪that a measurement should be compared against during analysis to\ndetermine the␣
↪health or pass/fail status of that measurement.",
         "type": "object",
         "properties": {
            "min": {
               "title": "Min",
               "type": "number"
            },
```

(continues on next page)

(continued from previous page)

```
            "typical": {
               "title": "Typical",
               "type": "number"
            },
            "max": {
               "title": "Max",
               "type": "number"
            }
         }
      },
      "ConditionType": {
         "title": "ConditionType",
         "description": "Conditions are either numeric or string type.",
         "enum": [
            "NUMERIC",
            "STRING"
         ]
      },
      "ConditionRange": {
         "title": "ConditionRange",
         "description": "Specifies the range of values that the condition must␣
↪cover.",
         "type": "object",
         "properties": {
            "min": {
               "title": "Min",
               "type": "number"
            },
            "max": {
               "title": "Max",
               "type": "number"
            },
            "step": {
               "title": "Step",
               "type": "number"
            }
         }
      },
      "NumericConditionValue": {
         "title": "NumericConditionValue",
         "description": "A numeric condition.\n\nNumeric conditions can contain a␣
↪combination of ranges and discrete lists.",
         "type": "object",
         "properties": {
            "conditionType": {
               "$ref": "#/definitions/ConditionType"
            },
            "range": {
               "title": "Range",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/ConditionRange"
```

(continues on next page)

(continued from previous page)

```
                }
            },
            "discrete": {
                "title": "Discrete",
                "type": "array",
                "items": {
                    "type": "number"
                }
            },
            "unit": {
                "title": "Unit",
                "type": "string"
            }
        },
        "required": [
            "conditionType"
        ]
    },
    "StringConditionValue": {
        "title": "StringConditionValue",
        "description": "A string condition.\n\nString conditions may only contain␣
↪discrete lists of values.",
        "type": "object",
        "properties": {
            "conditionType": {
                "$ref": "#/definitions/ConditionType"
            },
            "discrete": {
                "title": "Discrete",
                "type": "array",
                "items": {
                    "type": "string"
                }
            }
        },
        "required": [
            "conditionType"
        ]
    },
    "Condition": {
        "title": "Condition",
        "description": "A single condition.",
        "type": "object",
        "properties": {
            "name": {
                "title": "Name",
                "type": "string"
            },
            "value": {
                "title": "Value",
                "anyOf": [
                    {
```

(continues on next page)

(continued from previous page)


```
                    "$ref": "#/definitions/NumericConditionValue"
                },
                {
                    "$ref": "#/definitions/StringConditionValue"
                }
            ]
        }
    }
},
"SpecificationDefinition": {
    "title": "SpecificationDefinition",
    "description": "Base class for models that are serialized to and from JSON.
↪",
    "type": "object",
    "properties": {
        "productId": {
            "title": "Productid",
            "type": "string"
        },
        "specId": {
            "title": "Specid",
            "type": "string"
        },
        "workspace": {
            "title": "Workspace",
            "type": "string"
        },
        "name": {
            "title": "Name",
            "type": "string"
        },
        "category": {
            "title": "Category",
            "type": "string"
        },
        "type": {
            "$ref": "#/definitions/SpecificationType"
        },
        "symbol": {
            "title": "Symbol",
            "type": "string"
        },
        "block": {
            "title": "Block",
            "type": "string"
        },
        "limit": {
            "$ref": "#/definitions/SpecificationLimit"
        },
        "unit": {
            "title": "Unit",
            "type": "string"
```


(continues on next page)

(continued from previous page)


```
            },
            "conditions": {
               "title": "Conditions",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/Condition"
               }
            },
            "keywords": {
               "title": "Keywords",
               "type": "array",
               "items": {
                  "type": "string"
               }
            },
            "properties": {
               "title": "Properties",
               "type": "object",
               "additionalProperties": {
                  "type": "string"
               }
            }
         },
         "required": [
            "productId",
            "specId",
            "type"
         ]
      }
   }
}
```


Fields

- *specs*

field specs: Optional[List[*SpecificationDefinition*]] = None

List of specifications to be created.

pydantic model nisystemlink.clients.spec.models.**CreatedSpecification**

A specification successfully created on the server.


```
{
   "title": "CreatedSpecification",
   "description": "A specification successfully created on the server.",
   "type": "object",
   "properties": {
      "createdAt": {
         "title": "Createdat",
         "type": "string",
         "format": "date-time"
      },
```


(continues on next page)

(continued from previous page)


```
      "createdBy": {
         "title": "Createdby",
         "type": "string"
      },
      "productId": {
         "title": "Productid",
         "type": "string"
      },
      "specId": {
         "title": "Specid",
         "type": "string"
      },
      "workspace": {
         "title": "Workspace",
         "type": "string"
      },
      "id": {
         "title": "Id",
         "type": "string"
      },
      "version": {
         "title": "Version",
         "type": "integer"
      }
   },
   "required": [
      "productId",
      "specId",
      "id",
      "version"
   ]
}
```


Fields

- *id*
- *version*

field id: str [Required]

The global Id of the specification.

field version: int [Required]

Current version of the specification.

When an update is applied, the version is automatically incremented.

pydantic model nisystemlink.clients.spec.models.**DeleteSpecificationsPartialSuccess**

The results of deleting multiple specs when one or more of the specs could not be deleted.


```
{
   "title": "DeleteSpecificationsPartialSuccess",
   "description": "The results of deleting multiple specs when one or more of the␣
```


(continues on next page)

(continued from previous page)

```
↪specs could not be deleted.",
   "type": "object",
   "properties": {
      "deletedSpecIds": {
         "title": "Deletedspecids",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "failedSpecIds": {
         "title": "Failedspecids",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "error": {
         "$ref": "#/definitions/ApiError"
      }
   },
   "required": [
      "deletedSpecIds",
      "failedSpecIds",
      "error"
   ],
   "definitions": {
      "ApiError": {
         "title": "ApiError",
         "description": "Represents the standard error structure for SystemLink API␣
↪responses.",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "type": "string"
            },
            "code": {
               "title": "Code",
               "type": "integer"
            },
            "message": {
               "title": "Message",
               "type": "string"
            },
            "args": {
               "title": "Args",
               "default": [],
               "type": "array",
               "items": {
                  "type": "string"
               }
```

(continues on next page)

(continued from previous page)


```
            },
            "resourceType": {
               "title": "Resourcetype",
               "type": "string"
            },
            "resourceId": {
               "title": "Resourceid",
               "type": "string"
            },
            "innerErrors": {
               "title": "Innererrors",
               "default": [],
               "type": "array",
               "items": {
                  "$ref": "#/definitions/ApiError"
               }
            }
         }
      }
   }
}
```


Fields

- *deleted_spec_ids*
- *error*
- *failed_spec_ids*

field deleted_spec_ids: List[str] [Required]

Global IDs of the deleted specifications.

field error: *ApiError* [Required]

field failed_spec_ids: List[str] [Required]

Global IDs of the specifications that could not be deleted.

pydantic model nisystemlink.clients.spec.models.**NumericConditionValue**

A numeric condition.

Numeric conditions can contain a combination of ranges and discrete lists.


```
{
   "title": "NumericConditionValue",
   "description": "A numeric condition.\n\nNumeric conditions can contain a␣
↪combination of ranges and discrete lists.",
   "type": "object",
   "properties": {
      "conditionType": {
         "$ref": "#/definitions/ConditionType"
      },
      "range": {
         "title": "Range",
```


(continues on next page)

(continued from previous page)

```
            "type": "array",
            "items": {
               "$ref": "#/definitions/ConditionRange"
            }
         },
         "discrete": {
            "title": "Discrete",
            "type": "array",
            "items": {
               "type": "number"
            }
         },
         "unit": {
            "title": "Unit",
            "type": "string"
         }
      },
      "required": [
         "conditionType"
      ],
      "definitions": {
         "ConditionType": {
            "title": "ConditionType",
            "description": "Conditions are either numeric or string type.",
            "enum": [
               "NUMERIC",
               "STRING"
            ]
         },
         "ConditionRange": {
            "title": "ConditionRange",
            "description": "Specifies the range of values that the condition must␣
↪cover.",
            "type": "object",
            "properties": {
               "min": {
                  "title": "Min",
                  "type": "number"
               },
               "max": {
                  "title": "Max",
                  "type": "number"
               },
               "step": {
                  "title": "Step",
                  "type": "number"
               }
            }
         }
      }
}
```

Fields

- *discrete*
- *range*
- *unit*

field discrete: Optional[List[float]] = None

List of condition discrete values.

field range: Optional[List[*ConditionRange*]] = None

List of condition range values.

field unit: Optional[str] = None

Unit of the condition.

pydantic model nisystemlink.clients.spec.models.**Operation**

Represents an operation that can be performed on a data frame.


```
{
   "title": "Operation",
   "description": "Represents an operation that can be performed on a data frame.",
   "type": "object",
   "properties": {
      "available": {
         "title": "Available",
         "type": "boolean"
      },
      "version": {
         "title": "Version",
         "type": "integer"
      }
   },
   "required": [
      "available",
      "version"
   ]
}
```


Fields

- *available*
- *version*

field available: bool [Required]

Whether or not the operation is available to the caller (e.g. due to permissions).

field version: int [Required]

The version of the available operation.

pydantic model nisystemlink.clients.spec.models.**QuerySpecifications**

The list of matching specifications and a continuation token to get the next items.

```
{
   "title": "QuerySpecifications",
   "description": "The list of matching specifications and a continuation token to␣
↪get the next items.",
   "type": "object",
   "properties": {
      "continuationToken": {
         "title": "Continuationtoken",
         "type": "string"
      },
      "specs": {
         "title": "Specs",
         "type": "array",
         "items": {
            "$ref": "#/definitions/SpecificationWithHistory"
         }
      }
   },
   "definitions": {
      "SpecificationType": {
         "title": "SpecificationType",
         "description": "The overall type of the specification.",
         "enum": [
            "PARAMETRIC",
            "FUNCTIONAL"
         ]
      },
      "SpecificationLimit": {
         "title": "SpecificationLimit",
         "description": "A limit for a specification.\n\nThe limit is the value␣
↪that a measurement should be compared against during analysis to\ndetermine the␣
↪health or pass/fail status of that measurement.",
         "type": "object",
         "properties": {
            "min": {
               "title": "Min",
               "type": "number"
            },
            "typical": {
               "title": "Typical",
               "type": "number"
            },
            "max": {
               "title": "Max",
               "type": "number"
            }
         }
      },
      "ConditionType": {
         "title": "ConditionType",
         "description": "Conditions are either numeric or string type.",
         "enum": [
            "NUMERIC",
```


(continues on next page)

(continued from previous page)

```
            "STRING"
         ]
      },
      "ConditionRange": {
         "title": "ConditionRange",
         "description": "Specifies the range of values that the condition must␣
↪cover.",
         "type": "object",
         "properties": {
            "min": {
               "title": "Min",
               "type": "number"
            },
            "max": {
               "title": "Max",
               "type": "number"
            },
            "step": {
               "title": "Step",
               "type": "number"
            }
         }
      },
      "NumericConditionValue": {
         "title": "NumericConditionValue",
         "description": "A numeric condition.\n\nNumeric conditions can contain a␣
↪combination of ranges and discrete lists.",
         "type": "object",
         "properties": {
            "conditionType": {
               "$ref": "#/definitions/ConditionType"
            },
            "range": {
               "title": "Range",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/ConditionRange"
               }
            },
            "discrete": {
               "title": "Discrete",
               "type": "array",
               "items": {
                  "type": "number"
               }
            },
            "unit": {
               "title": "Unit",
               "type": "string"
            }
         },
         "required": [
```

(continues on next page)

(continued from previous page)

```
            "conditionType"
         ]
      },
      "StringConditionValue": {
         "title": "StringConditionValue",
         "description": "A string condition.\n\nString conditions may only contain␣
↪discrete lists of values.",
         "type": "object",
         "properties": {
            "conditionType": {
               "$ref": "#/definitions/ConditionType"
            },
            "discrete": {
               "title": "Discrete",
               "type": "array",
               "items": {
                  "type": "string"
               }
            }
         },
         "required": [
            "conditionType"
         ]
      },
      "Condition": {
         "title": "Condition",
         "description": "A single condition.",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "type": "string"
            },
            "value": {
               "title": "Value",
               "anyOf": [
                  {
                     "$ref": "#/definitions/NumericConditionValue"
                  },
                  {
                     "$ref": "#/definitions/StringConditionValue"
                  }
               ]
            }
         }
      },
      "SpecificationWithHistory": {
         "title": "SpecificationWithHistory",
         "description": "A full specification with update and create history.",
         "type": "object",
         "properties": {
            "updatedAt": {
```

(continues on next page)

(continued from previous page)

```
            "title": "Updatedat",
            "type": "string",
            "format": "date-time"
        },
        "updatedBy": {
            "title": "Updatedby",
            "type": "string"
        },
        "createdAt": {
            "title": "Createdat",
            "type": "string",
            "format": "date-time"
        },
        "createdBy": {
            "title": "Createdby",
            "type": "string"
        },
        "id": {
            "title": "Id",
            "type": "string"
        },
        "version": {
            "title": "Version",
            "type": "integer"
        },
        "productId": {
            "title": "Productid",
            "type": "string"
        },
        "specId": {
            "title": "Specid",
            "type": "string"
        },
        "workspace": {
            "title": "Workspace",
            "type": "string"
        },
        "name": {
            "title": "Name",
            "type": "string"
        },
        "category": {
            "title": "Category",
            "type": "string"
        },
        "type": {
            "$ref": "#/definitions/SpecificationType"
        },
        "symbol": {
            "title": "Symbol",
            "type": "string"
        },
```

(continues on next page)

(continued from previous page)


```
            "block": {
               "title": "Block",
               "type": "string"
            },
            "limit": {
               "$ref": "#/definitions/SpecificationLimit"
            },
            "unit": {
               "title": "Unit",
               "type": "string"
            },
            "conditions": {
               "title": "Conditions",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/Condition"
               }
            },
            "keywords": {
               "title": "Keywords",
               "type": "array",
               "items": {
                  "type": "string"
               }
            },
            "properties": {
               "title": "Properties",
               "type": "object",
               "additionalProperties": {
                  "type": "string"
               }
            }
         },
         "required": [
            "id",
            "version",
            "productId",
            "specId",
            "type"
         ]
      }
   }
}
```


Fields

- *specs*

field specs: Optional[List[*SpecificationWithHistory*]] = None

List of queried specifications.

An empty list indicates that there are no specifications meeting the criteria provided in the request.

pydantic model nisystemlink.clients.spec.models.**QuerySpecificationsRequest**

The request to query specifications.


```
{
   "title": "QuerySpecificationsRequest",
   "description": "The request to query specifications.",
   "type": "object",
   "properties": {
      "productIds": {
         "title": "Productids",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "take": {
         "title": "Take",
         "type": "integer"
      },
      "continuationToken": {
         "title": "Continuationtoken",
         "type": "string"
      },
      "filter": {
         "title": "Filter",
         "type": "string"
      },
      "projection": {
         "type": "array",
         "items": {
            "$ref": "#/definitions/Projection"
         }
      },
      "orderBy": {
         "$ref": "#/definitions/OrderBy"
      },
      "orderByDescending": {
         "title": "Orderbydescending",
         "type": "boolean"
      }
   },
   "required": [
      "productIds"
   ],
   "definitions": {
      "Projection": {
         "title": "Projection",
         "description": "The allowed projections for query.\n\nWhen using␣
↪projection, only the fields specified by the projection element will be included␣
↪in\nthe response.",
         "enum": [
            "ID",
            "PRODUCT_ID",
```


(continues on next page)

(continued from previous page)

```
            "SPEC_ID",
            "NAME",
            "CATEGORY",
            "TYPE",
            "SYMBOL",
            "BLOCK",
            "LIMIT",
            "UNIT",
            "CONDITION_NAME",
            "CONDITION_VALUES",
            "CONDITION_UNIT",
            "CONDITION_TYPE",
            "KEYWORDS",
            "PROPERTIES",
            "WORKSPACE",
            "CREATED_AT",
            "CREATED_BY"
         ],
         "type": "string"
      },
      "OrderBy": {
         "title": "OrderBy",
         "description": "The valid ways to order the response to a spec query.",
         "enum": [
            "ID",
            "SPEC_ID"
         ]
      }
   }
}
```

Fields

- *continuation_token*
- *filter*
- *order_by*
- *order_by_descending*
- *product_ids*
- *projection*
- *take*

field continuation_token: Optional[str] = None

Allows users to continue the query at the next specification that matches the given criteria.

To retrieve the next batch of specifications, pass the continuation token from the previous batch in the next request. The service responds with the next batch of data and provides a new continuation token. To paginate results, continue sending requests with the newest continuation token provided in each response.

field filter: Optional[str] = None

The specification query filter in Dynamic Linq format.

Allowed properties in the filter are: - *specId*: String representing the SpecID of a specification. - *name*: String representing the name of a specification. - *category*: String representing the category of a specification. - *type*: String enumeration representing the type of the specification.

Possible values are : PARAMETRIC, FUNCTIONAL

- *block*: String representing the block of a specification.
- *symbol*: String representing the symbol of a specification.
- *unit*: String representing the unit of a specification.
- *workspace*: String representing the ID of the workspace the specification belongs to.
- *createdBy*: String representing the ID of the user who created the specification.
- *createdAt*: ISO-8601 formatted UTC timestamp indicating when the specification was created.
- *updatedBy*: String representing the ID of the user who updated the specification.
- *updatedAt*: ISO-8601 formatted UTC timestamp indicating when the specification was updated.

See [Dynamic Linq](https://github.com/ni/systemlink-OpenAPI-documents/wiki/Dynamic-Linq-Query-Language) documentation for more details.

field order_by: Optional[*OrderBy*] = None

Specifies the field to use to sort specifications.

By default, specifications are sorted by *ID*.

field order_by_descending: Optional[bool] = None

Specifies whether to return the specifications in descending order.

By default, this value is *false* and specifications are sorted in ascending order.

field product_ids: List[str] [Required]

IDs of the products to query the specifications for.

field projection: Optional[List[*Projection*]] = None

Specifies the fields to include in the returned specifications.

Fields you do not specify are excluded. Returns all fields if no value is specified.

field take: Optional[int] = None

Maximum number of specifications to return in the current API response.

Uses the default if the specified value is negative. The default value is *1000* specs.

pydantic model nisystemlink.clients.spec.models.**Specification**

The complete definition of a specification.


```
{
   "title": "Specification",
   "description": "The complete definition of a specification.",
   "type": "object",
   "properties": {
      "id": {
         "title": "Id",
         "type": "string"
      },
      "version": {
```


(continues on next page)

(continued from previous page)

```
            "title": "Version",
            "type": "integer"
        },
        "productId": {
            "title": "Productid",
            "type": "string"
        },
        "specId": {
            "title": "Specid",
            "type": "string"
        },
        "workspace": {
            "title": "Workspace",
            "type": "string"
        },
        "name": {
            "title": "Name",
            "type": "string"
        },
        "category": {
            "title": "Category",
            "type": "string"
        },
        "type": {
            "$ref": "#/definitions/SpecificationType"
        },
        "symbol": {
            "title": "Symbol",
            "type": "string"
        },
        "block": {
            "title": "Block",
            "type": "string"
        },
        "limit": {
            "$ref": "#/definitions/SpecificationLimit"
        },
        "unit": {
            "title": "Unit",
            "type": "string"
        },
        "conditions": {
            "title": "Conditions",
            "type": "array",
            "items": {
                "$ref": "#/definitions/Condition"
            }
        },
        "keywords": {
            "title": "Keywords",
            "type": "array",
            "items": {
```

(continues on next page)

(continued from previous page)

```
            "type": "string"
         }
      },
      "properties": {
         "title": "Properties",
         "type": "object",
         "additionalProperties": {
            "type": "string"
         }
      }
   },
   "required": [
      "id",
      "version",
      "productId",
      "specId",
      "type"
   ],
   "definitions": {
      "SpecificationType": {
         "title": "SpecificationType",
         "description": "The overall type of the specification.",
         "enum": [
            "PARAMETRIC",
            "FUNCTIONAL"
         ]
      },
      "SpecificationLimit": {
         "title": "SpecificationLimit",
         "description": "A limit for a specification.\n\nThe limit is the value␣
↪that a measurement should be compared against during analysis to\ndetermine the␣
↪health or pass/fail status of that measurement.",
         "type": "object",
         "properties": {
            "min": {
               "title": "Min",
               "type": "number"
            },
            "typical": {
               "title": "Typical",
               "type": "number"
            },
            "max": {
               "title": "Max",
               "type": "number"
            }
         }
      },
      "ConditionType": {
         "title": "ConditionType",
         "description": "Conditions are either numeric or string type.",
         "enum": [
```

(continues on next page)

(continued from previous page)

```
            "NUMERIC",
            "STRING"
         ]
      },
      "ConditionRange": {
         "title": "ConditionRange",
         "description": "Specifies the range of values that the condition must␣
↪cover.",
         "type": "object",
         "properties": {
            "min": {
               "title": "Min",
               "type": "number"
            },
            "max": {
               "title": "Max",
               "type": "number"
            },
            "step": {
               "title": "Step",
               "type": "number"
            }
         }
      },
      "NumericConditionValue": {
         "title": "NumericConditionValue",
         "description": "A numeric condition.\n\nNumeric conditions can contain a␣
↪combination of ranges and discrete lists.",
         "type": "object",
         "properties": {
            "conditionType": {
               "$ref": "#/definitions/ConditionType"
            },
            "range": {
               "title": "Range",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/ConditionRange"
               }
            },
            "discrete": {
               "title": "Discrete",
               "type": "array",
               "items": {
                  "type": "number"
               }
            },
            "unit": {
               "title": "Unit",
               "type": "string"
            }
         },
```

(continues on next page)

(continued from previous page)

```
            "required": [
               "conditionType"
            ]
         },
         "StringConditionValue": {
            "title": "StringConditionValue",
            "description": "A string condition.\n\nString conditions may only contain␣
↪discrete lists of values.",
            "type": "object",
            "properties": {
               "conditionType": {
                  "$ref": "#/definitions/ConditionType"
               },
               "discrete": {
                  "title": "Discrete",
                  "type": "array",
                  "items": {
                     "type": "string"
                  }
               }
            },
            "required": [
               "conditionType"
            ]
         },
         "Condition": {
            "title": "Condition",
            "description": "A single condition.",
            "type": "object",
            "properties": {
               "name": {
                  "title": "Name",
                  "type": "string"
               },
               "value": {
                  "title": "Value",
                  "anyOf": [
                     {
                        "$ref": "#/definitions/NumericConditionValue"
                     },
                     {
                        "$ref": "#/definitions/StringConditionValue"
                     }
                  ]
               }
            }
         }
      }
}
```

Fields

- *block*

- *category*
- *conditions*
- *keywords*
- *limit*
- *name*
- *properties*
- *symbol*
- *type*
- *unit*

field block: Optional[str] = None

Block name of the specification.

Typically a block is one of the subsystems of the overall product being specified.

field category: Optional[str] = None

Category of the specification.

field conditions: Optional[List[*Condition*]] = None

Conditions associated with the specification.

field keywords: Optional[List[str]] = None

Keywords or phrases associated with the specification.

field limit: Optional[*SpecificationLimit*] = None

The limits for this spec.

field name: Optional[str] = None

Name of the specification.

field properties: Optional[Dict[str, str]] = None

Additional properties associated with the specification.

field symbol: Optional[str] = None

Short form identifier of the specification.

field type: *SpecificationType* [Required]

Type of the specification.

field unit: Optional[str] = None

Unit of the specification.

pydantic model nisystemlink.clients.spec.models.**SpecificationCreation**

When the spec was created and when.


```
{
   "title": "SpecificationCreation",
   "description": "When the spec was created and when.",
   "type": "object",
   "properties": {
      "createdAt": {
         "title": "Createdat",
```


(continues on next page)

(continued from previous page)

```
            "type": "string",
            "format": "date-time"
        },
        "createdBy": {
            "title": "Createdby",
            "type": "string"
        }
    }
}
```

Fields

- *created_at*
- *created_by*

field created_at: Optional[datetime] = None

ISO-8601 formatted timestamp indicating when the specification was created.

field created_by: Optional[str] = None

Id of the user who created the specification.

pydantic model nisystemlink.clients.spec.models.**SpecificationDefinition**

```
{
   "title": "SpecificationDefinition",
   "description": "Base class for models that are serialized to and from JSON.",
   "type": "object",
   "properties": {
      "productId": {
         "title": "Productid",
         "type": "string"
      },
      "specId": {
         "title": "Specid",
         "type": "string"
      },
      "workspace": {
         "title": "Workspace",
         "type": "string"
      },
      "name": {
         "title": "Name",
         "type": "string"
      },
      "category": {
         "title": "Category",
         "type": "string"
      },
      "type": {
         "$ref": "#/definitions/SpecificationType"
      },
      "symbol": {
```

(continues on next page)

(continued from previous page)

```
            "title": "Symbol",
            "type": "string"
        },
        "block": {
            "title": "Block",
            "type": "string"
        },
        "limit": {
            "$ref": "#/definitions/SpecificationLimit"
        },
        "unit": {
            "title": "Unit",
            "type": "string"
        },
        "conditions": {
            "title": "Conditions",
            "type": "array",
            "items": {
                "$ref": "#/definitions/Condition"
            }
        },
        "keywords": {
            "title": "Keywords",
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "properties": {
            "title": "Properties",
            "type": "object",
            "additionalProperties": {
                "type": "string"
            }
        }
    },
    "required": [
        "productId",
        "specId",
        "type"
    ],
    "definitions": {
        "SpecificationType": {
            "title": "SpecificationType",
            "description": "The overall type of the specification.",
            "enum": [
                "PARAMETRIC",
                "FUNCTIONAL"
            ]
        },
        "SpecificationLimit": {
            "title": "SpecificationLimit",
```

(continues on next page)

(continued from previous page)

```
        "description": "A limit for a specification.\n\nThe limit is the value␣
↪that a measurement should be compared against during analysis to\ndetermine the␣
↪health or pass/fail status of that measurement.",
        "type": "object",
        "properties": {
           "min": {
              "title": "Min",
              "type": "number"
           },
           "typical": {
              "title": "Typical",
              "type": "number"
           },
           "max": {
              "title": "Max",
              "type": "number"
           }
        }
     },
     "ConditionType": {
        "title": "ConditionType",
        "description": "Conditions are either numeric or string type.",
        "enum": [
           "NUMERIC",
           "STRING"
        ]
     },
     "ConditionRange": {
        "title": "ConditionRange",
        "description": "Specifies the range of values that the condition must␣
↪cover.",
        "type": "object",
        "properties": {
           "min": {
              "title": "Min",
              "type": "number"
           },
           "max": {
              "title": "Max",
              "type": "number"
           },
           "step": {
              "title": "Step",
              "type": "number"
           }
        }
     },
     "NumericConditionValue": {
        "title": "NumericConditionValue",
        "description": "A numeric condition.\n\nNumeric conditions can contain a␣
↪combination of ranges and discrete lists.",
        "type": "object",
```

(continues on next page)

(continued from previous page)

```
         "properties": {
            "conditionType": {
               "$ref": "#/definitions/ConditionType"
            },
            "range": {
               "title": "Range",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/ConditionRange"
               }
            },
            "discrete": {
               "title": "Discrete",
               "type": "array",
               "items": {
                  "type": "number"
               }
            },
            "unit": {
               "title": "Unit",
               "type": "string"
            }
         },
         "required": [
            "conditionType"
         ]
      },
      "StringConditionValue": {
         "title": "StringConditionValue",
         "description": "A string condition.\n\nString conditions may only contain␣
↪discrete lists of values.",
         "type": "object",
         "properties": {
            "conditionType": {
               "$ref": "#/definitions/ConditionType"
            },
            "discrete": {
               "title": "Discrete",
               "type": "array",
               "items": {
                  "type": "string"
               }
            }
         },
         "required": [
            "conditionType"
         ]
      },
      "Condition": {
         "title": "Condition",
         "description": "A single condition.",
         "type": "object",
```

(continues on next page)

(continued from previous page)

```
        "properties": {
           "name": {
              "title": "Name",
              "type": "string"
           },
           "value": {
              "title": "Value",
              "anyOf": [
                 {
                    "$ref": "#/definitions/NumericConditionValue"
                 },
                 {
                    "$ref": "#/definitions/StringConditionValue"
                 }
              ]
           }
        }
     }
  }
}
```

Fields

- *block*
- *category*
- *conditions*
- *keywords*
- *limit*
- *name*
- *properties*
- *symbol*
- *type*
- *unit*

field block: Optional[str] = None

Block name of the specification.

Typically a block is one of the subsystems of the overall product being specified.

field category: Optional[str] = None

Category of the specification.

field conditions: Optional[List[*Condition*]] = None

Conditions associated with the specification.

field keywords: Optional[List[str]] = None

Keywords or phrases associated with the specification.

field limit: Optional[*SpecificationLimit*] = None

The limits for this spec.

field name: Optional[str] = None

Name of the specification.

field properties: Optional[Dict[str, str]] = None

Additional properties associated with the specification.

field symbol: Optional[str] = None

Short form identifier of the specification.

field type: *SpecificationType* [Required]

Type of the specification.

field unit: Optional[str] = None

Unit of the specification.

pydantic model nisystemlink.clients.spec.models.**SpecificationLimit**

A limit for a specification.

The limit is the value that a measurement should be compared against during analysis to determine the health or pass/fail status of that measurement.

```
{
   "title": "SpecificationLimit",
   "description": "A limit for a specification.\n\nThe limit is the value that a␣
↪measurement should be compared against during analysis to\ndetermine the health␣
↪or pass/fail status of that measurement.",
   "type": "object",
   "properties": {
      "min": {
         "title": "Min",
         "type": "number"
      },
      "typical": {
         "title": "Typical",
         "type": "number"
      },
      "max": {
         "title": "Max",
         "type": "number"
      }
   }
}
```

Fields

- *max*
- *min*
- *typical*

field max: Optional[float] = None

Maximum value of the specification.

All measurements that map to this specification should be < this limit.

field min: Optional[float] = None

Minimum limit of the specification.

All measurements that map to this specification should be > this limit.

field typical: Optional[float] = None

Typical value of the specification.

pydantic model nisystemlink.clients.spec.models.**SpecificationServerManaged**


```
{
   "title": "SpecificationServerManaged",
   "description": "Base class for models that are serialized to and from JSON.",
   "type": "object",
   "properties": {
      "id": {
         "title": "Id",
         "type": "string"
      },
      "version": {
         "title": "Version",
         "type": "integer"
      }
   },
   "required": [
      "id",
      "version"
   ]
}
```


Fields

- *id*
- *version*

field id: str [Required]

The global Id of the specification.

field version: int [Required]

Current version of the specification.

When an update is applied, the version is automatically incremented.

class nisystemlink.clients.spec.models.**SpecificationType**(*value*)

The overall type of the specification.

FUNCTIONAL = 'FUNCTIONAL'

Functional specs only have pass/fail status.

PARAMETRIC = 'PARAMETRIC'

Parametric specs have limits.

pydantic model nisystemlink.clients.spec.models.**SpecificationUpdated**

When the spec was updated and when.

```
{
   "title": "SpecificationUpdated",
   "description": "When the spec was updated and when.",
   "type": "object",
   "properties": {
      "updatedAt": {
         "title": "Updatedat",
         "type": "string",
         "format": "date-time"
      },
      "updatedBy": {
         "title": "Updatedby",
         "type": "string"
      }
   }
}
```


Fields

- *updated_at*
- *updated_by*

field updated_at: Optional[datetime] = None

ISO-8601 formatted timestamp indicating when the specification was last updated.

field updated_by: Optional[str] = None

Id of the user who last updated the specification.

pydantic model nisystemlink.clients.spec.models.**SpecificationUserManaged**


```
{
   "title": "SpecificationUserManaged",
   "description": "Base class for models that are serialized to and from JSON.",
   "type": "object",
   "properties": {
      "productId": {
         "title": "Productid",
         "type": "string"
      },
      "specId": {
         "title": "Specid",
         "type": "string"
      },
      "workspace": {
         "title": "Workspace",
         "type": "string"
      }
   },
   "required": [
      "productId",
      "specId"
   ]
}
```

Fields

- *product_id*
- *spec_id*
- *workspace*

field product_id: str [Required]

Id of the product to which the specification will be associated.

field spec_id: str [Required]

User provided value using which the specification will be identified.

This should be unique for a product and workspace combination.

field workspace: Optional[str] = None

Id of the workspace to which the specification will be associated.

Default workspace will be taken if the value is not given.

pydantic model nisystemlink.clients.spec.models.**SpecificationWithHistory**

A full specification with update and create history.


```
{
   "title": "SpecificationWithHistory",
   "description": "A full specification with update and create history.",
   "type": "object",
   "properties": {
      "updatedAt": {
         "title": "Updatedat",
         "type": "string",
         "format": "date-time"
      },
      "updatedBy": {
         "title": "Updatedby",
         "type": "string"
      },
      "createdAt": {
         "title": "Createdat",
         "type": "string",
         "format": "date-time"
      },
      "createdBy": {
         "title": "Createdby",
         "type": "string"
      },
      "id": {
         "title": "Id",
         "type": "string"
      },
      "version": {
         "title": "Version",
         "type": "integer"
      },
      "productId": {
```


(continues on next page)

(continued from previous page)

```
            "title": "Productid",
            "type": "string"
        },
        "specId": {
            "title": "Specid",
            "type": "string"
        },
        "workspace": {
            "title": "Workspace",
            "type": "string"
        },
        "name": {
            "title": "Name",
            "type": "string"
        },
        "category": {
            "title": "Category",
            "type": "string"
        },
        "type": {
            "$ref": "#/definitions/SpecificationType"
        },
        "symbol": {
            "title": "Symbol",
            "type": "string"
        },
        "block": {
            "title": "Block",
            "type": "string"
        },
        "limit": {
            "$ref": "#/definitions/SpecificationLimit"
        },
        "unit": {
            "title": "Unit",
            "type": "string"
        },
        "conditions": {
            "title": "Conditions",
            "type": "array",
            "items": {
                "$ref": "#/definitions/Condition"
            }
        },
        "keywords": {
            "title": "Keywords",
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "properties": {
```

(continues on next page)

(continued from previous page)

```
        "title": "Properties",
        "type": "object",
        "additionalProperties": {
           "type": "string"
        }
     }
  },
  "required": [
     "id",
     "version",
     "productId",
     "specId",
     "type"
  ],
  "definitions": {
     "SpecificationType": {
        "title": "SpecificationType",
        "description": "The overall type of the specification.",
        "enum": [
           "PARAMETRIC",
           "FUNCTIONAL"
        ]
     },
     "SpecificationLimit": {
        "title": "SpecificationLimit",
        "description": "A limit for a specification.\n\nThe limit is the value␣
↪that a measurement should be compared against during analysis to\ndetermine the␣
↪health or pass/fail status of that measurement.",
        "type": "object",
        "properties": {
           "min": {
              "title": "Min",
              "type": "number"
           },
           "typical": {
              "title": "Typical",
              "type": "number"
           },
           "max": {
              "title": "Max",
              "type": "number"
           }
        }
     },
     "ConditionType": {
        "title": "ConditionType",
        "description": "Conditions are either numeric or string type.",
        "enum": [
           "NUMERIC",
           "STRING"
        ]
     },
```

(continues on next page)

(continued from previous page)

```
      "ConditionRange": {
         "title": "ConditionRange",
         "description": "Specifies the range of values that the condition must␣
↪cover.",
         "type": "object",
         "properties": {
            "min": {
               "title": "Min",
               "type": "number"
            },
            "max": {
               "title": "Max",
               "type": "number"
            },
            "step": {
               "title": "Step",
               "type": "number"
            }
         }
      },
      "NumericConditionValue": {
         "title": "NumericConditionValue",
         "description": "A numeric condition.\n\nNumeric conditions can contain a␣
↪combination of ranges and discrete lists.",
         "type": "object",
         "properties": {
            "conditionType": {
               "$ref": "#/definitions/ConditionType"
            },
            "range": {
               "title": "Range",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/ConditionRange"
               }
            },
            "discrete": {
               "title": "Discrete",
               "type": "array",
               "items": {
                  "type": "number"
               }
            },
            "unit": {
               "title": "Unit",
               "type": "string"
            }
         },
         "required": [
            "conditionType"
         ]
      },
```

(continues on next page)

(continued from previous page)

```
      "StringConditionValue": {
         "title": "StringConditionValue",
         "description": "A string condition.\n\nString conditions may only contain␣
↪discrete lists of values.",
         "type": "object",
         "properties": {
            "conditionType": {
               "$ref": "#/definitions/ConditionType"
            },
            "discrete": {
               "title": "Discrete",
               "type": "array",
               "items": {
                  "type": "string"
               }
            }
         },
         "required": [
            "conditionType"
         ]
      },
      "Condition": {
         "title": "Condition",
         "description": "A single condition.",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "type": "string"
            },
            "value": {
               "title": "Value",
               "anyOf": [
                  {
                     "$ref": "#/definitions/NumericConditionValue"
                  },
                  {
                     "$ref": "#/definitions/StringConditionValue"
                  }
               ]
            }
         }
      }
   }
}
```

Fields

- *block*
- *category*
- *conditions*

- *id*
- *keywords*
- *limit*
- *name*
- *product_id*
- *properties*
- *spec_id*
- *symbol*
- *type*
- *unit*
- *version*
- *workspace*

field block: Optional[str] = None

Block name of the specification.

Typically a block is one of the subsystems of the overall product being specified.

field category: Optional[str] = None

Category of the specification.

field conditions: Optional[List[*Condition*]] = None

Conditions associated with the specification.

field id: str [Required]

The global Id of the specification.

field keywords: Optional[List[str]] = None

Keywords or phrases associated with the specification.

field limit: Optional[*SpecificationLimit*] = None

The limits for this spec.

field name: Optional[str] = None

Name of the specification.

field product_id: str [Required]

Id of the product to which the specification will be associated.

field properties: Optional[Dict[str, str]] = None

Additional properties associated with the specification.

field spec_id: str [Required]

User provided value using which the specification will be identified.

This should be unique for a product and workspace combination.

field symbol: Optional[str] = None

Short form identifier of the specification.

field type: *SpecificationType* [Required]

Type of the specification.

field unit: Optional[str] = None

Unit of the specification.

field version: int [Required]

Current version of the specification.

When an update is applied, the version is automatically incremented.

field workspace: Optional[str] = None

Id of the workspace to which the specification will be associated.

Default workspace will be taken if the value is not given.

pydantic model nisystemlink.clients.spec.models.**StringConditionValue**

A string condition.

String conditions may only contain discrete lists of values.


```
{
   "title": "StringConditionValue",
   "description": "A string condition.\n\nString conditions may only contain␣
↪discrete lists of values.",
   "type": "object",
   "properties": {
      "conditionType": {
         "$ref": "#/definitions/ConditionType"
      },
      "discrete": {
         "title": "Discrete",
         "type": "array",
         "items": {
            "type": "string"
         }
      }
   },
   "required": [
      "conditionType"
   ],
   "definitions": {
      "ConditionType": {
         "title": "ConditionType",
         "description": "Conditions are either numeric or string type.",
         "enum": [
            "NUMERIC",
            "STRING"
         ]
      }
   }
}
```


Fields

- *discrete*

field discrete: Optional[List[str]] = None

List of condition discrete values.

pydantic model nisystemlink.clients.spec.models.**UpdateSpecificationsPartialSuccess**


```
{
   "title": "UpdateSpecificationsPartialSuccess",
   "description": "Base class for models that are serialized to and from JSON.",
   "type": "object",
   "properties": {
      "updatedSpecs": {
         "title": "Updatedspecs",
         "type": "array",
         "items": {
            "$ref": "#/definitions/UpdatedSpecification"
         }
      },
      "failedSpecs": {
         "title": "Failedspecs",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Specification"
         }
      },
      "error": {
         "$ref": "#/definitions/ApiError"
      }
   },
   "definitions": {
      "UpdatedSpecification": {
         "title": "UpdatedSpecification",
         "description": "A specification that was updated on the server.",
         "type": "object",
         "properties": {
            "updatedAt": {
               "title": "Updatedat",
               "type": "string",
               "format": "date-time"
            },
            "updatedBy": {
               "title": "Updatedby",
               "type": "string"
            },
            "id": {
               "title": "Id",
               "type": "string"
            },
            "version": {
               "title": "Version",
               "type": "integer"
            },
            "productId": {
               "title": "Productid",
               "type": "string"
            },
            "specId": {
```


(continues on next page)

(continued from previous page)

```
            "title": "Specid",
            "type": "string"
         },
         "workspace": {
            "title": "Workspace",
            "type": "string"
         }
      },
      "required": [
         "id",
         "version",
         "productId",
         "specId"
      ]
   },
   "SpecificationType": {
      "title": "SpecificationType",
      "description": "The overall type of the specification.",
      "enum": [
         "PARAMETRIC",
         "FUNCTIONAL"
      ]
   },
   "SpecificationLimit": {
      "title": "SpecificationLimit",
      "description": "A limit for a specification.\n\nThe limit is the value␣
↪that a measurement should be compared against during analysis to\ndetermine the␣
↪health or pass/fail status of that measurement.",
      "type": "object",
      "properties": {
         "min": {
            "title": "Min",
            "type": "number"
         },
         "typical": {
            "title": "Typical",
            "type": "number"
         },
         "max": {
            "title": "Max",
            "type": "number"
         }
      }
   },
   "ConditionType": {
      "title": "ConditionType",
      "description": "Conditions are either numeric or string type.",
      "enum": [
         "NUMERIC",
         "STRING"
      ]
   },
```

(continues on next page)

(continued from previous page)

```
      "ConditionRange": {
         "title": "ConditionRange",
         "description": "Specifies the range of values that the condition must␣
↪cover.",
         "type": "object",
         "properties": {
            "min": {
               "title": "Min",
               "type": "number"
            },
            "max": {
               "title": "Max",
               "type": "number"
            },
            "step": {
               "title": "Step",
               "type": "number"
            }
         }
      },
      "NumericConditionValue": {
         "title": "NumericConditionValue",
         "description": "A numeric condition.\n\nNumeric conditions can contain a␣
↪combination of ranges and discrete lists.",
         "type": "object",
         "properties": {
            "conditionType": {
               "$ref": "#/definitions/ConditionType"
            },
            "range": {
               "title": "Range",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/ConditionRange"
               }
            },
            "discrete": {
               "title": "Discrete",
               "type": "array",
               "items": {
                  "type": "number"
               }
            },
            "unit": {
               "title": "Unit",
               "type": "string"
            }
         },
         "required": [
            "conditionType"
         ]
      },
```

(continues on next page)

(continued from previous page)


```
      "StringConditionValue": {
         "title": "StringConditionValue",
         "description": "A string condition.\n\nString conditions may only contain␣
↪discrete lists of values.",
         "type": "object",
         "properties": {
            "conditionType": {
               "$ref": "#/definitions/ConditionType"
            },
            "discrete": {
               "title": "Discrete",
               "type": "array",
               "items": {
                  "type": "string"
               }
            }
         },
         "required": [
            "conditionType"
         ]
      },
      "Condition": {
         "title": "Condition",
         "description": "A single condition.",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "type": "string"
            },
            "value": {
               "title": "Value",
               "anyOf": [
                  {
                     "$ref": "#/definitions/NumericConditionValue"
                  },
                  {
                     "$ref": "#/definitions/StringConditionValue"
                  }
               ]
            }
         }
      },
      "Specification": {
         "title": "Specification",
         "description": "The complete definition of a specification.",
         "type": "object",
         "properties": {
            "id": {
               "title": "Id",
               "type": "string"
            },
```


(continues on next page)

(continued from previous page)

```
            "version": {
               "title": "Version",
               "type": "integer"
            },
            "productId": {
               "title": "Productid",
               "type": "string"
            },
            "specId": {
               "title": "Specid",
               "type": "string"
            },
            "workspace": {
               "title": "Workspace",
               "type": "string"
            },
            "name": {
               "title": "Name",
               "type": "string"
            },
            "category": {
               "title": "Category",
               "type": "string"
            },
            "type": {
               "$ref": "#/definitions/SpecificationType"
            },
            "symbol": {
               "title": "Symbol",
               "type": "string"
            },
            "block": {
               "title": "Block",
               "type": "string"
            },
            "limit": {
               "$ref": "#/definitions/SpecificationLimit"
            },
            "unit": {
               "title": "Unit",
               "type": "string"
            },
            "conditions": {
               "title": "Conditions",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/Condition"
               }
            },
            "keywords": {
               "title": "Keywords",
               "type": "array",
```

(continues on next page)

(continued from previous page)

```
                "items": {
                    "type": "string"
                }
            },
            "properties": {
                "title": "Properties",
                "type": "object",
                "additionalProperties": {
                    "type": "string"
                }
            }
        },
        "required": [
            "id",
            "version",
            "productId",
            "specId",
            "type"
        ]
    },
    "ApiError": {
        "title": "ApiError",
        "description": "Represents the standard error structure for SystemLink API␣
↪responses.",
        "type": "object",
        "properties": {
            "name": {
                "title": "Name",
                "type": "string"
            },
            "code": {
                "title": "Code",
                "type": "integer"
            },
            "message": {
                "title": "Message",
                "type": "string"
            },
            "args": {
                "title": "Args",
                "default": [],
                "type": "array",
                "items": {
                    "type": "string"
                }
            },
            "resourceType": {
                "title": "Resourcetype",
                "type": "string"
            },
            "resourceId": {
                "title": "Resourceid",
```

(continues on next page)

(continued from previous page)


```
                "type": "string"
            },
            "innerErrors": {
                "title": "Innererrors",
                "default": [],
                "type": "array",
                "items": {
                    "$ref": "#/definitions/ApiError"
                }
            }
        }
    }
}
```


Fields

- *error*
- *failed_specs*
- *updated_specs*

field error: Optional[*ApiError*] = None

Any errors that occurred.

field failed_specs: Optional[List[*Specification*]] = None

Information about each of the specification request(s) that failed during the update.

field updated_specs: Optional[List[*UpdatedSpecification*]] = None

Information about each of the updated specification(s).

pydantic model nisystemlink.clients.spec.models.**UpdateSpecificationsRequest**


```
{
    "title": "UpdateSpecificationsRequest",
    "description": "Base class for models that are serialized to and from JSON.",
    "type": "object",
    "properties": {
        "specs": {
            "title": "Specs",
            "type": "array",
            "items": {
                "$ref": "#/definitions/Specification"
            }
        }
    },
    "definitions": {
        "SpecificationType": {
            "title": "SpecificationType",
            "description": "The overall type of the specification.",
            "enum": [
                "PARAMETRIC",
                "FUNCTIONAL"
```


(continues on next page)

(continued from previous page)

```
            ]
         },
         "SpecificationLimit": {
            "title": "SpecificationLimit",
            "description": "A limit for a specification.\n\nThe limit is the value␣
↪that a measurement should be compared against during analysis to\ndetermine the␣
↪health or pass/fail status of that measurement.",
            "type": "object",
            "properties": {
               "min": {
                  "title": "Min",
                  "type": "number"
               },
               "typical": {
                  "title": "Typical",
                  "type": "number"
               },
               "max": {
                  "title": "Max",
                  "type": "number"
               }
            }
         },
         "ConditionType": {
            "title": "ConditionType",
            "description": "Conditions are either numeric or string type.",
            "enum": [
               "NUMERIC",
               "STRING"
            ]
         },
         "ConditionRange": {
            "title": "ConditionRange",
            "description": "Specifies the range of values that the condition must␣
↪cover.",
            "type": "object",
            "properties": {
               "min": {
                  "title": "Min",
                  "type": "number"
               },
               "max": {
                  "title": "Max",
                  "type": "number"
               },
               "step": {
                  "title": "Step",
                  "type": "number"
               }
            }
         },
         "NumericConditionValue": {
```

(continues on next page)

(continued from previous page)

```
        "title": "NumericConditionValue",
        "description": "A numeric condition.\n\nNumeric conditions can contain a␣
↪combination of ranges and discrete lists.",
        "type": "object",
        "properties": {
           "conditionType": {
              "$ref": "#/definitions/ConditionType"
           },
           "range": {
              "title": "Range",
              "type": "array",
              "items": {
                 "$ref": "#/definitions/ConditionRange"
              }
           },
           "discrete": {
              "title": "Discrete",
              "type": "array",
              "items": {
                 "type": "number"
              }
           },
           "unit": {
              "title": "Unit",
              "type": "string"
           }
        },
        "required": [
           "conditionType"
        ]
     },
     "StringConditionValue": {
        "title": "StringConditionValue",
        "description": "A string condition.\n\nString conditions may only contain␣
↪discrete lists of values.",
        "type": "object",
        "properties": {
           "conditionType": {
              "$ref": "#/definitions/ConditionType"
           },
           "discrete": {
              "title": "Discrete",
              "type": "array",
              "items": {
                 "type": "string"
              }
           }
        },
        "required": [
           "conditionType"
        ]
     },
```

(continues on next page)

(continued from previous page)


```
    "Condition": {
      "title": "Condition",
      "description": "A single condition.",
      "type": "object",
      "properties": {
        "name": {
          "title": "Name",
          "type": "string"
        },
        "value": {
          "title": "Value",
          "anyOf": [
            {
              "$ref": "#/definitions/NumericConditionValue"
            },
            {
              "$ref": "#/definitions/StringConditionValue"
            }
          ]
        }
      }
    },
    "Specification": {
      "title": "Specification",
      "description": "The complete definition of a specification.",
      "type": "object",
      "properties": {
        "id": {
          "title": "Id",
          "type": "string"
        },
        "version": {
          "title": "Version",
          "type": "integer"
        },
        "productId": {
          "title": "Productid",
          "type": "string"
        },
        "specId": {
          "title": "Specid",
          "type": "string"
        },
        "workspace": {
          "title": "Workspace",
          "type": "string"
        },
        "name": {
          "title": "Name",
          "type": "string"
        },
        "category": {
```


(continues on next page)

(continued from previous page)

```
            "title": "Category",
            "type": "string"
         },
         "type": {
            "$ref": "#/definitions/SpecificationType"
         },
         "symbol": {
            "title": "Symbol",
            "type": "string"
         },
         "block": {
            "title": "Block",
            "type": "string"
         },
         "limit": {
            "$ref": "#/definitions/SpecificationLimit"
         },
         "unit": {
            "title": "Unit",
            "type": "string"
         },
         "conditions": {
            "title": "Conditions",
            "type": "array",
            "items": {
               "$ref": "#/definitions/Condition"
            }
         },
         "keywords": {
            "title": "Keywords",
            "type": "array",
            "items": {
               "type": "string"
            }
         },
         "properties": {
            "title": "Properties",
            "type": "object",
            "additionalProperties": {
               "type": "string"
            }
         }
      },
      "required": [
         "id",
         "version",
         "productId",
         "specId",
         "type"
      ]
   }
}
```

(continues on next page)

(continued from previous page)

```
}
```

Fields

- *specs*

field specs: Optional[List[*Specification*]] = None

List of specifications to be updated.

pydantic model nisystemlink.clients.spec.models.**UpdatedSpecification**

A specification that was updated on the server.


```
{
   "title": "UpdatedSpecification",
   "description": "A specification that was updated on the server.",
   "type": "object",
   "properties": {
      "updatedAt": {
         "title": "Updatedat",
         "type": "string",
         "format": "date-time"
      },
      "updatedBy": {
         "title": "Updatedby",
         "type": "string"
      },
      "id": {
         "title": "Id",
         "type": "string"
      },
      "version": {
         "title": "Version",
         "type": "integer"
      },
      "productId": {
         "title": "Productid",
         "type": "string"
      },
      "specId": {
         "title": "Specid",
         "type": "string"
      },
      "workspace": {
         "title": "Workspace",
         "type": "string"
      }
   },
   "required": [
      "id",
      "version",
      "productId",
      "specId"
```


(continues on next page)

(continued from previous page)

```
   ]
}
```

Fields

- *product_id*
- *spec_id*
- *workspace*

field product_id: str [Required]

Id of the product to which the specification will be associated.

field spec_id: str [Required]

User provided value using which the specification will be identified.

This should be unique for a product and workspace combination.

field workspace: Optional[str] = None

Id of the workspace to which the specification will be associated.

Default workspace will be taken if the value is not given.

pydantic model nisystemlink.clients.spec.models.**V1Operations**

The operations available in the routes provided by the v1 HTTP API.

```
{
   "title": "V1Operations",
   "description": "The operations available in the routes provided by the v1 HTTP␣
↪API.",
   "type": "object",
   "properties": {
      "createSpecifications": {
         "$ref": "#/definitions/Operation"
      },
      "querySpecifications": {
         "$ref": "#/definitions/Operation"
      },
      "updateSpecifications": {
         "$ref": "#/definitions/Operation"
      },
      "deleteSpecifications": {
         "$ref": "#/definitions/Operation"
      }
   },
   "required": [
      "createSpecifications",
      "querySpecifications",
      "updateSpecifications",
      "deleteSpecifications"
   ],
   "definitions": {
      "Operation": {
         "title": "Operation",
```

(continues on next page)

(continued from previous page)

```
        "description": "Represents an operation that can be performed on a data␣
↪frame.",
        "type": "object",
        "properties": {
           "available": {
              "title": "Available",
              "type": "boolean"
           },
           "version": {
              "title": "Version",
              "type": "integer"
           }
        },
        "required": [
           "available",
           "version"
        ]
     }
  }
}
```

Fields

- *create_specifications*
- *delete_specifications*
- *query_specifications*
- *update_specifications*

field create_specifications: *Operation* **[Required]**

The ability to create new specifications.

field delete_specifications: *Operation* **[Required]**

The ability to delete specifications.

field query_specifications: *Operation* **[Required]**

The ability to query specifications.

field update_specifications: *Operation* **[Required]**

The ability to update specifications.

1.2.5 Indices and tables

- genindex
- modindex

CHAPTER

TWO

PACKAGE INFO

Info	NI SystemLink API Clients for Python
Author	National Instruments

2.1 About

The **nisystemlink-clients** package contains an API (Application Programming Interface) for SystemLink that uses HTTP to interact with a SystemLink Server. The package is implemented in Python. NI created and supports this package.

2.2 Requirements

nisystemlink-clients has the following requirements:

- A SystemLink Server installation or a SystemLink Cloud account to connect to
- CPython 3.8+

2.3 Installation

To install **nisystemlink-clients**, use one of the following methods:

1. pip:

```
$ python -m pip install nisystemlink-clients
```

2. **easy_install** from setuptools:

```
$ python -m easy_install nisystemlink-clients
```

2.4 Usage

Refer to the documentation for detailed information on how to use **nisystemlink-clients**.

2.5 Support / Feedback

The **nisystemlink-clients** package is supported by NI. For support for **nisystemlink-clients**, open a request through the NI support portal at ni.com.

2.6 Bugs / Feature Requests

To report a bug or submit a feature request, please use the GitHub issues page.

2.7 Documentation

To view the documentation, visit the nisystemlink.clients Documentation Page.

2.8 Changelog

To view the changelog, visit the GitHub repository CHANGELOG.

2.9 License

nisystemlink-clients is licensed under an MIT-style license (see LICENSE). Other incorporated projects may be licensed under different licenses. All licenses allow for non-commercial and commercial use.

PYTHON MODULE INDEX

n

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y