nisystemlink

Release 0.1.3

National Instruments

Nov 15, 2022

CONTENTS

- *User Documentation*
- *Package Info*
  - *About*
  - *Requirements*
  - *Installation*
  - *Usage*
  - *Support / Feedback*
  - *Bugs / Feature Requests*
  - *Documentation*
  - *License*

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

```python
from datetime import timedelta

from systemlink.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

```python
from contextlib import ExitStack
from time import sleep

from systemlink.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)
```

```python
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
```

1.2 API Reference

1.2.1 systemlink.clients.core

class systemlink.clients.core.**ApiError**

Represents the standard error structure for SystemLink API responses.

    property args: List[str]

        The list of positional arguments formatted into the error.

            Return type

                List[str]

property code: Optional[int]

> The numeric code associated with the error.
>
> > **Return type**
> >
> > > Optional[int]

copy()

> Get a copy of this object.
>
> > **Return type**
> >
> > > *ApiError*

property inner_errors: List[*ApiError*]

> The list of inner errors.
>
> > **Return type**
> >
> > > List[*ApiError*]

property message: Optional[str]

> The error message.
>
> > **Return type**
> >
> > > Optional[str]

property name: Optional[str]

> The name of the error.
>
> > **Return type**
> >
> > > Optional[str]

property resource_id: Optional[str]

> The ID of the resource associated with the error, if any.
>
> Set *resource_type* when setting this property.
>
> > **Return type**
> >
> > > Optional[str]

property resource_type: Optional[str]

> The type of resource associated with the error, if any.
>
> Set this when setting *resource_id*.
>
> > **Return type**
> >
> > > Optional[str]

exception systemlink.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[*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.

> **Return type**
>> Optional[*ApiError*]

property http_status_code: Optional[int]

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

> **Return type**
>> Optional[int]

property inner_exception: Optional[Exception]

The exception that caused this failure, if any.

> **Return type**
>> Optional[Exception]

property message: Optional[str]

The error message.

> **Return type**
>> Optional[str]

with_traceback()

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

class systemlink.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.

> **Return type**
>> Optional[Dict[str, str]]

property cert_path: Optional[Path]

Local path to an SSL certificate file.

> **Return type**
>> Optional[pathlib.Path]

property password: Optional[str]

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

> **Return type**
>> Optional[str]

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.

> **Return type**
>
> str

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.

> **Return type**
>
> int

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.

> **Return type**
>
> Optional[str]

property username: Optional[str]

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

> **Return type**
>
> Optional[str]

class systemlink.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.
>
> > **Return type**
> >   Optional[Dict[str, str]]

property cert_path: Optional[Path]

> Local path to an SSL certificate file.
>
> > **Return type**
> >   Optional[pathlib.Path]

property password: Optional[str]

> The password to use for HTTP authentication, or None if none was provided.
>
> > **Return type**
> >   Optional[str]

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.
>
> > **Return type**
> >   str

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.
>
> > **Return type**
> >   int

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.
>
> > **Return type**
> >   Optional[str]

property username: Optional[str]

> The username to use for HTTP authentication, or None if none was provided.
>
> > **Return type**
> >   Optional[str]

class systemlink.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**
> > *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.

1.2.2 systemlink.clients.tag

class systemlink.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.
> >
> > > **Return type**
> > > Optional[List[*TagData*]]
>
> **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[*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[*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.

Return type

int

class systemlink.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** (*DataType*) – The data type of the tag to write.

Return type

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** (*[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** (*[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 systemlink.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.

> **Return type**
> > str

class systemlink.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** (*[DataType](#)*) – The data type of the tag to read.
>
> **Return type**
> > *[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[*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[*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 systemlink.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** (*DataType*) – The data type of the tag to write.
>
> **Return type**
> *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** (*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** (*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 systemlink.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 systemlink.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[`*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*.

Return type

bool

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.

> **Return type**
>> [*DataType*](#)

property keywords: List[str]

The list of keywords associated with the tag.

> **Return type**
>> List[str]

property path: str

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

> **Return type**
>> str

property properties: Dict[str, str]

The properties associated with the tag.

> **Return type**
>> Dict[str, str]

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.

> **Return type**
>> Optional[int]

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.

> **Return type**
>> Optional[int]

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.

> **Return type**
> > *RetentionType*

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** (*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 systemlink.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** (*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 re-
> > >   placed 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.
> >
> > > **Return type**
> > > Optional[bool]
>
> **property data_type: *DataType***
>
> > The data type for the tag's values.
> >
> > > **Return type**
> > > *DataType*
>
> **classmethod from_tagdata**(*data*, *fields*)
>
> > Create an update by taking one or more fields from a *TagData*.
> >
> > > **Parameters**
> > >
> > > - **data** (*TagData*) – The metadata to send to the server in the update.
> > >
> > > - **fields** (*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**
> > > *TagDataUpdate*

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

> The list of keywords to send with the update, or None to not send any keywords.
>
> > **Return type**
> > Optional[Tuple[str, ...]]

property path: str

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

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

> The properties send with the update, or None to not send any properties.
>
> > **Return type**
> > Optional[Dict[str, str]]

class systemlink.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[*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[*TagData*]) – The tags to include in the selection.
> >
> > **Return type**
> > > *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**
>> [*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[[*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[[*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** ([*DataType*](#)) – The data type of the tag to read.
>
> **Return type**
>> [*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[*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**
>   >   *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[*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**
>   >   *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**
>> `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[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
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
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[*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[*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[`*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[*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[*TagData*], Sequence[*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[*TagData*], Sequence[*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 systemlink.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 systemlink.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.
>
> > **Return type**
> > > int

class systemlink.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[*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**
> [*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[[*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** (*DataType*) – The data type of the tag to read.
>
> **Return type**
>> *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.
>
> **Return type**
>> Dict[str, *TagData*]

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.
>
> > **Return type**
> >
> > > Tuple[str, ...]

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[*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[*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[*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.
>>
>> > **Return type**
>> > Dict[str, *TagValueReader*]

class systemlink.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:

```python
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 systemlink.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 systemlink.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** (*ITagReader*) – The *ITagReader* to use when reading values.
> >
> > - **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.
>
> > **Return type**
> > *DataType*

property path: **str**

> The path of the tag associated with the value.
>
> > **Return type**
> > str

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[*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[*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 systemlink.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** (*`ITagWriter`*) – The *`ITagWriter`* to use when writing values.
> > >
> > > - **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.
> >
> > **Return type**
> > > *`DataType`*

> **property path:** str
>
> > The path of the tag associated with the value.
> >
> > **Return type**
> > > str

> **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 systemlink.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.

Return type

Optional[int]

property data_type: *DataType*

The data type of the value.

Return type

DataType

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.

Return type

Union[int, float, None]

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.

Return type

Optional[float]

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.

Return type

Union[int, float, None]

property path: str

The path of the tag associated with the value.

Return type

str

property timestamp: Optional[datetime]

The timestamp associated with the value, if available.

Return type

Optional[datetime.datetime]

property value: _Any_

The value of the tag.

> **Return type**
> TypeVar(typing.Any)

1.2.3 Indices and tables

- genindex

- modindex

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 3.5+

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
```

3. Download the project source and run:

```
$ python setup.py install
```

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 systemlink.clients Documentation Page.

2.8 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

S