QConnectBase

v. 1.1.0

Nguyen Huynh Tri Cuong

05.07.2022

Contents

Chapter 1

Introduction

QConnectBase
TO BE CONTINUED

Chapter 2

Description

QConnectBase
TO BE CONTINUED

Chapter 3

__init__.py

3.1 Class: ConnectionManager

```
QConnectBase.__init__
```

Class to manage all connections.

Chapter 4

connection_base.py

4.1 Class: BrokenConnError

```
QConnectBase.connection_base
```

4.2 Class: ConnectionBase

```
QConnectBase.connection_base
```

Base class for all connection classes. Method: is_supported_platform ----------
Check if current platform is supported.

Returns: True if platform is supported.
False if platform is not supported.

4.2.1 Method: is_precondition_pass

Check for precondition.

Returns: True if passing the precondition.
False if failing the precondition.

4.2.2 Method: error_instruction

Get the error instruction.

Returns: Error instruction string.

4.2.3 Method: quit

>> This method MUST be overridden in derived class << Abstract method for quiting the connection.

Args: is_disconnect_all: Determine if it's necessary to disconnect all connections.

Returns: None.

4.2.4 Method: connect

>> This method MUST be overridden in derived class << Abstract method for quiting the connection.

Args: device: Determine if it's necessary to disconnect all connections.

files: Determine if it's necessary to disconnect all connections.

test_connection: Determine if it's necessary to disconnect all connections.

Returns: None.

4.2.5 Method: disconnect

>> This method MUST be overridden in derived class << Abstract method for disconnecting connection.

Args: device: Device name.

Returns: None.

4.2.6 Method: send_obj

Wrapper method to send message to a tcp connection.

Args: obj: Data to be sent. cr: Determine if it's necessary to add newline character at the end of command.

Returns: None

4.2.7 Method: read_obj

Wrapper method to get the response from connection.

Returns: Responded message.

4.2.8 Method: wait_4_trace

Suspend the control flow until a Trace message is received which matches to a specified regular expression.

Args: search_obj : Regular expression all received trace messages are compare to. Can be passed either as a string or a regular expression object. Refer to Python documentation for module 're'.

use_fetch_block : Determine if 'fetch block' feature is used.

end_of_block_pattern : The end of block pattern.

filter_pattern : Regular expression object to filter message line by line.

timeout : Optional timeout parameter specified as a floating point number in the unit 'seconds'.

fct_args: Optional list of function arguments passed to be sent.

Returns: None : If no trace message matched to the specified regular expression and a timeout occurred.

<match> : If a trace message has matched to the specified regular expression, a match object is returned as the result. The complete trace message can be accessed by the 'string' attribute of the match object. For access to groups within the regular expression, use the group() method. For more information, refer to Python documentation for module 're'.

4.2.9 Method: wait_4_trace_continuously

Getting trace log continuously without creating a new trace queue.

Args: trace_queue: Queue to store the traces.

timeout: Timeout for waiting a matched log.

fct_args: Arguments to be sent to connection.

Returns: None : If no trace message matched to the specified regular expression and a timeout occurred.

match object : If a trace message has matched to the specified regular expression, a match object is returned as the result. The complete trace message can be accessed by the 'string' attribute of the match object. For access to groups within the regular expression, use the group() method. For more information, refer to Python documentation for module 're'.

4.2.10 Method: create_and_activate_trace_queue

Create Queue and assign it to _trace_queue object and activate the queue with the search element.

Args: search_element : Regular expression all received trace messages are compare to. Can be passed either as a string or a regular expression object. Refer to Python documentation for module 're'.#

use_fetch_block : Determine if 'fetch block' feature is used.

end_of_block_pattern : The end of block pattern.

regex_line_filter_pattern : Regular expression object to filter message line by line.

Returns: trq_handle, trace_queue: the handle and search object

4.2.11 Method: deactivate_and_delete_trace_queue

Deactivate trace queue and delete.

Args: trq_handle: Trace queue handle.

trace_queue: Trace queue object.

Returns: None.

4.2.12 Method: activate_trace_queue

Activates a trace message filter specified as a regular expression. All matching trace messages are put in the specified queue object.

Args: search_obj : Regular expression all received trace messages are compare to. Can be passed either as a string or a regular expression object. Refer to Python documentation for module 're'.#

trace_queue : A queue object all trace message which matches the regular expression are put in. The using application must assure, that the queue is emptied or deleted.

use_fetch_block : Determine if 'fetch block' feature is used

end_of_block_pattern : The end of block pattern

line_filter_pattern : Regular expression object to filter message line by line.

Returns: <int> : Handle to deactivate the message filter.

4.2.13 Method: deactivate_trace_queue

Deactivates a trace message filter previously activated by ActivateTraceQ() method.

Args: handle : Integer object returned by ActivateTraceQ() method.

Returns: False : No trace message filter active with the specified handle (i.e. handle is not in use).
True : Trace message filter successfully deleted.

4.2.14 Method: check_timeout

>> This method will be override in derived class << Check if responded message come in cls._RESPOND_TIMEOUT or we will raise a timeout event.

Args: msg: Responded message for checking.

Returns: None.

4.2.15 Method: pre_msg_check

>> This method will be override in derived class << Pre-checking message when receiving it from connection.

Args: msg: received message to be checked.

Returns: None.

4.2.16 Method: post_msg_check

>> This method will be override in derived class << Post-checking message when receiving it from connection.

Args: msg: received message to be checked.

Returns: None.

Chapter 5

connection_manager.py

5.1 Class: InputParam

```
QConnectBase.connection_manager
```

5.1.1 Method: get_attr_list

5.2 Class: ConnectParam

```
QConnectBase.connection_manager
```

Class for storing parameters for connect action. Class: SendCommandParam ==========

```
QConnectBase.connection_manager
```

Class for storing parameters for send command action. Class: VerifyParam ==========

```
QConnectBase.connection_manager
```

Class for storing parameters for verify action. Class: ConnectionManager ==========

```
QConnectBase.connection_manager
```

Class to manage all connections. Method: quit ----------
Quit connection manager.

Returns: None.

5.2.1 Method: add_connection

Add a connection to managed dictionary.

Args: name: connection's name.
conn: connection object.

Returns: None.

5.2.2 Method: remove_connection

Remove a connection by name.

Args: connection_name: connection name.

Returns: None.

5.2.3 Method: get_connection_by_name

Get an exist connection by name.

Args: connection_name: connection's name.

Returns: Connection object.

5.2.4 Method: disconnect

Keyword for disconnecting a connection by name.

Args: connection_name: Name of connection.

Returns: None.

5.2.5 Method: connect

Keyword for making a connection.

Args: args: Non-Keyword Arguments.
kwargs: Keyword Arguments.

Returns: None.

5.2.6 Method: connect_named_args

Making a connection with name arguments.

Args: kwargs: Dictionary of arguments.

Returns: None.

5.2.7 Method: connect_unnamed_args

Making a connection.

Args: connection_name: Name of connection.
connection_type: Type of connection.
mode: Connection mode.
config: Configuration for connection.

Returns: None.

5.2.8 Method: send_command

Keyword for sending command to a connection.

Args: args: Non-Keyword Arguments.
kwargs: Keyword Arguments.

Returns: None.

5.2.9 Method: send_command_named_args

Send command to a connection with name arguments.

Args: args: Dictionary of arguments.

Returns: None.

5.2.10 Method: send_command_unnamed_args

Send command to a connection.

Args: connection_name: connection's name.
command: command.

Returns: None.

5.2.11 Method: verify

Keyword uses to verify a pattern from connection response after sending a command.

Args: args: Non-Keyword Arguments.
kwargs: Keyword Arguments.

Returns: match_res: matched string.

5.2.12 Method: verify_named_args

Verify a pattern from connection response after sending a command with named arguments.

Args: kwargs: Dictionary of arguments.

Returns: match_res: matched string.

5.2.13 Method: verify_unnamed_args

Verify a pattern from connection response after sending a command.

Args: connection_name: connection's name.
search_obj: search pattern.
timeout: timeout for waiting result.
fetch_block: use fetch block feature.
end_of_block_pattern: pattern for detecting the end of block.
filter_pattern: line filter pattern.
fct_args: command to be sent.

Returns: match_res: matched string.

5.3 Class: TestOption

```
QConnectBase.connection_manager
```

Chapter 6

constants.py

6.1 Class: SocketType

```
QConnectBase.constants
```

6.2 Class: String

```
QConnectBase.constants
```

Chapter 7

qlogger.py

7.1 Class: ColorFormatter

```
QConnectBase.qlogger
```

Custom formatter class for setting log color. Method: format ----------
Set the color format for the log.

Args: record: log record.

Returns: Log with color formatter.

7.2 Class: QFileHandler

```
QConnectBase.qlogger
```

Handler class for user defined file in config. Method: get_log_path ----------
Get the log file path for this handler.

Args: config: connection configurations.

Returns: Log file path.

7.2.1 Method: get_config_supported

Check if the connection config is supported by this handler.

Args: config: connection configurations.

Returns: True if the config is supported.
False if the config is not supported.

7.3 Class: QDefaultFileHandler

```
QConnectBase.qlogger
```

Handler class for default log file path. Method: get_log_path ----------
Get the log file path for this handler.

Args: logger_name: name of the logger.

Returns: Log file path.

7.3.1 Method: get_config_supported

Check if the connection config is supported by this handler.

Args: config: connection configurations.

Returns: True if the config is supported.
False if the config is not supported.

7.4 Class: QConsoleHandler

```
QConnectBase.qlogger
```

Handler class for console log. Method: get_config_supported ----------
Check if the connection config is supported by this handler.

Args: config: connection configurations.

Returns: True if the config is supported.
False if the config is not supported.

7.5 Class: QLogger

```
QConnectBase.qlogger
```

Logger class for QConnect Libraries. Method: get_logger ----------
Get the logger object.

Args: logger_name: Name of the logger.

Returns: Logger object.

7.5.1 Method: set_handler

Set handler for logger.

Args: config: connection configurations.

Returns: None if no handler is set. Handler object.

Chapter 8

serial_base.py

8.1 Class: SerialConfig

```
QConnectBase.serialclient.serial_base
```

Class to store the configuration for Serial connection. Class: SerialSocket ==========

```
QConnectBase.serialclient.serial_base
```

Class for handling serial connection. Method: connect ----------
Connect to serial port.

Returns: None.

8.1.1 Method: disconnect

Disconnect serial port.

Args: _device: unused.

Returns: None.

8.1.2 Method: quit

Quit serial connection.

Returns: None

8.2 Class: SerialClient

```
QConnectBase.serialclient.serial_base
```

Serial client class. Method: connect ----------
Connect to the Serial port.

Returns: None.

Chapter 9

raw_tcp.py

9.1 Class: RawTCPBase

```
QConnectBase.tcp/raw.raw_tcp
```

Base class for a raw tcp connection. Class: RawTCPServer ==========

```
QConnectBase.tcp/raw.raw_tcp
```

Class for a raw tcp connection server. Class: RawTCPClient ==========

```
QConnectBase.tcp/raw.raw_tcp
```

Class for a raw tcp connection client.

Chapter 10

ssh_client.py

10.1 Class: AuthenticationType

```
QConnectBase.tcp/ssh.ssh_client
```

10.2 Class: SSHConfig

```
QConnectBase.tcp/ssh.ssh_client
```

Class to store the configuration for SSH connection. Class: SSHClient ==========

```
QConnectBase.tcp/ssh.ssh_client
```

SSH client connection class. Method: connect ----------
Implementation for creating a SSH connection.

Returns: None.

10.2.1 Method: close

Close SSH connection.
Returns: Method: quit ----------
Quit and stop receiver thread.

Returns: None.

Chapter 11

tcp_base.py

11.1 Class: TCPConfig

```
QConnectBase.tcp.tcp_base
```

Class to store configurations for TCP connection. Class: TCPBase ==========

```
QConnectBase.tcp.tcp_base
```

Base class for a tcp connection. Method: close ----------
Close connection.

Returns: None.

11.1.1 Method: quit

Quit connection.

Args: is_disconnect_all: Determine if it's necessary for disconnect all connection.

Returns: None.

11.1.2 Method: connect

>> Should be override in derived class. Establish the connection.

Returns: None.

11.1.3 Method: disconnect

>> Should be override in derived class. Disconnect the connection.

Returns: None.

11.2 Class: TCPBaseServer

```
QConnectBase.tcp.tcp_base
```

Base class for TCP server. Method: accept_connection ----------
Wrapper method for handling accept action of TCP Server.

Returns: None.

11.2.1 Method: connect

11.2.2 Method: disconnect

11.3 Class: TCPBaseClient

```
QConnectBase.tcp.tcp_base
```

Base class for TCP client. Method: connect ----------

11.3.1 Method: disconnect

Chapter 12

utils.py

12.1 Class: Singleton

```
QConnectBase.utils
```

Class to implement Singleton Design Pattern. This class is used to derive the TTFisClientReal as only a single instance of this class is allowed.

Disabled pyLint Messages: R0903: Too few public methods (%s/%s) Used when class has too few public methods, so be sure it's really worth it.

> This base class implements the Singleton Design Pattern required for the TTFisClientReal. Adding further methods does not make sense.

12.2 Class: DictToClass

```
QConnectBase.utils
```

Class for converting dictionary to class object. Method: validate ----------

12.3 Class: Utils

```
QConnectBase.utils
```

Class to implement utilities for supporting development. Method: get_all_descendant_classes ----------

Get all descendant classes of a class

Args: cls: Input class for finding descendants.

Returns: Array of descendant classes.

12.3.1 Method: get_all_sub_classes

Get all children classes of a class

Args: cls: Input class for finding children.

Returns: Array of children classes.

12.3.2 Method: is_valid_host

12.3.3 Method: execute_command

12.3.4 Method: kill_process

12.3.5 Method: caller_name

Get a name of a caller in the format module.class.method

Args: skip: specifies how many levels of stack to skip while getting caller name. skip=1 means "who calls me", skip=2 "who calls my caller" etc.

Returns: An empty string is returned if skipped levels exceed stack height

12.3.6 Method: load_library

Load native library depend on the calling convention.

Args: path: library path. is_stdcall: determine if the library's calling convention is stdcall or cdecl.

Returns: Loaded library object.

12.3.7 Method: is_ascii_or_unicode

Check if the string is ascii or unicode

Args: str_check: string for checking codecs: encoding type list

Returns: True : if checked string is ascii or unicode False : if checked string is not ascii or unicode

12.4 Class: Job

```
QConnectBase.utils
```

12.4.1 Method: stop

12.4.2 Method: run

12.5 Class: ResultType

```
QConnectBase.utils
```

Result Types. Class: ResponseMessage ==========

```
QConnectBase.utils
```

Response message class Method: get_json ----------

Convert response message to json Returns: Response message in json format Method: get_data ----------
---

Get string data result Returns: String result Method: create_from_string ----------

Chapter 13

Appendix

About this package:

Table 13.1: Package setup

Setup parameter	Value
Name	QConnectBase
Version	1.1.0
Date	05.07.2022
Description	Robot Framework test library for TCP, SSH, serial connection
Package URL	robotframework-qconnect-base
Author	Nguyen Huynh Tri Cuong
Email	cuong.nguyenhuynhtri@vn.bosch.com
Language	Programming Language :: Python :: 3
License	License :: OSI Approved :: Apache Software License
OS	Operating System :: OS Independent
Python required	>=3.0
Development status	Development Status :: 4 - Beta
Intended audience	Intended Audience :: Developers
Topic	Topic :: Software Development

Chapter 14

History

1.0.0	2022
Initial version	