Verification Continuum™ Verdi® Python-Based NPI Transaction Waveform Model

Version V-2023.12-SP1, March 2024

SYNOPSYS®

Copyright and Proprietary Information Notice

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Feedback

Contents

Customer Support 4

Synopsys Statement on Inclusivity and Diversity 5

1. Introduction to Python-Based NPI 6

Packages and Modules 6

Module Functions and Class Objects 7

User Interface and Use Flow 7

Environment and Library Setting 7

2. Python-Based NPI Transaction Waveform Model 9

Overview 9

Quick Start 9

Enums 10

TrScope Enum 10

Trt Enums 12

Val/VCT Enums 12

L0 APIs 15

File 15

Transaction Scope 29

Stream 35

Transaction Traverse 39

Relation 49

Preface

The Python-Based NPI Transaction Waveform Model User Guide provides information to let you read the waveform file and can get signal values in the waveform with this model.

Customer Support

For any online access to the self-help resources, you can refer to the documentation and searchable knowledge base available in SolvNetPlus.

To obtain support for your Verdi product, choose one of the following:

- Open a case through SolvNetPlus.

  Go to https://solvnetplus.synopsys.com/s/contactsupport and provide the requested information, including:

  - Product L1 as Verdi
  - Case Type

  Fill in the remaining fields according to your environment and issue.

- Send an e-mail message to verdi_support@synopsys.com.

  Include product name (L1), sub-product name/technology (L2), and product version in your e-mail, so it can be routed correctly.

  Your e-mail will be acknowledged by automatic reply and assigned a Case number along with Case reference ID in the subject (ref:_...:ref).

  For any further communication on this Case via e-mail, send e-mail to verdi_support@synopsys.com and ensure to have the same Case ref ID in the subject header or else it will open duplicate cases.

- You can call for support at:

  https://www.synopsys.com/support/global-support-centers.html

Note:

In general, we need to be able to reproduce the problem in order to fix it, so a simple model demonstrating the error is the most effective way for us to identify the bug. If that is not possible, then provide a detailed explanation of the problem along with complete error and corresponding code, if any/permissible.

Synopsys Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. are reviewing and removing exclusionary language from products and supporting customer-facing collateral. effort also includes internal initiatives to remove biased language from engineering and working environment, including terms that are embedded in and IPs. At the same time, are working to ensure that web content and applications are usable to people of varying abilities. You may still find examples of non-inclusive language in or documentation as IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

1

Introduction to Python-Based NPI

Python-Based NPI APIs support seven models:

- Coverage
- Language
- Netlist
- Text
- Transaction Waveform Model
- Waveform
- Waveform Writer

Each model have their own APIs to let you be able to traverse data objects and obtain objects' properties like the existing C-Based or Tcl-Based NPI APIs.

In this guide, the environment setting for using **Python-Based NPI Transaction Waveform Model** is demonstrated.

Packages and Modules

Packages

The Python-based NPI package name is "pynpi", and it is placed at `$VERDI_HOME/share/NPI/python`.

Modules

There are seven modules inside the "pynpi" package: npisys, lang, netlist, text, cov waveform and waveformw. The first module, npisys, is the system model for initialization, loading design and exit. The other modules represent language model, netlist model, text model, coverage model, wave model and waveform writer model respectively

Module Functions and Class Objects

L0 Module Functions

Every module provides some L0 (level 0) functions to let you get the class objects. These functions return a class object or a list of class objects, and they follow the specification of the existing L0 APIs provided in C or Tcl.

L1 Module Functions

Similar to L0 module functions, every module also provides some L1 (level 1) functions to let you get advanced information based on the results obtained by L0 module functions. These functions follows the specification of the existing L1 APIs provided in C or Tcl.

Class Objects

The class object is similar to the so-called handle in NPI C APIs. The most difference is that some basic L0 APIs in C and Tcl will become class method function. These L0 APIs are usually to get integer value, string value, 1-to-1 method to get a handle, and 1-to-many method to get handle iterator.

User Interface and Use Flow

Environment and Library Setting

The python library setting flow of using Python-Based NPI APIs contains four parts:

1. Check your Python's version:

   Python-Based NPI APIs need the Python version greater than 3.6.0.

2. Environment setting for "VERDI_HOME" is required for Python-based NPI. Ensure that you set it up before running program.

3. Add python library path into your python code before loading Python-Based NPI by the following commands:

   ```
   rel_lib_path = os.environ['VERDI_HOME'] + '/share/NPI/python'
   sys.path.append(os.path.abspath(rel_lib_path))
   ```

4. Import module "npisys" for using the function of NPI initialization and exit from pynpi package.

   ```
   from pynpi import npisys
   ```

5. Import the module you need from pynpi package. For example, if you want to use Coverage model, you can import module like the following:

```
from pynpi import cov
```

6. Note that initialization function `npisys.init()` must be called before writing your code by using any other modules. Also, you must call `npisys.end()` after finishing your code. Following is a simple example to demonstrate how to use coverage model by Python-Based NPI APIs.

   Python program to use NPI models: (`demo.py`)

```
#!/global/freeware/Linux/2.X/python-3.6.0/bin/python
import sys, os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import cov
# Initialize NPI
if not npisys.init(sys.argv):
print("Error: Fail to initialize NPI")
assert 0
# Load design (if needed, depends on models)
if not npisys.load_design(sys.argv):
print("Error: Fail to load design")
assert 0
# Beginning of your code here --------------------
#
# Example code can be found in later chapters
#
# End of your code -------------------------------
# End NPI
npisys.end()
```

   C shell script example to setup environment and execute Python program: (`run_demo`)

```
#!/bin/csh -f
# Setup your $VERDI_HOME here
setenv VERDI_HOME [YOUR_VERDI_HOME_PATH]
# run the python program
# - Input arguments depend on your program design
# - If loading design is required, you can pass the options like
./demo.py -sv demo.v
```

   To run the files, put the above files in the same directory and execute the run_demo C shell script.

```
./run_demo
```

2

Python-Based NPI Transaction Waveform Model

This chapter includes the following topics:

- Overview
- Quick Start
- Enums
- L0 APIs

Overview

The NPI Transaction Waveform Model allows reading the waveform file. You can get transaction information in the waveform with this model. Database accessing performance and the programming usability are both considered in this model.

Quick Start

The following are the Environment and library setting:

1. Add python library path by the following commands:

   ```
   rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"

   sys.path.append(os.path.abspath(rel_lib_path))
   ```

2. Import `npisys` to use the function of NPI initialization and exit.

   Import `waveform` to use the APIs of Waveform Model.

   ```
   from pynpi import npisys

   from pynpi import waveform
   ```

3. If there exists any error in `LD_LIBRARY_PATH`, please add `$VERDI_HOME/share/NPI/lib/linux64` and `$VERDI_HOME/platform/linux64/bin` to `LD_LIBRARY_PATH`:

   ```
   os.environ['LD_LIBRARY_PATH'] =
   os.environ['VERDI_HOME']+'/share/NPI/lib/
   ```

Feedback

```
linux64:'+os.environ['VERDI_HOME']+'/platform/linux64/
bin:'+os.environ['LD_LIBRARY_PATH']
```

Enums

- Enum list
- TrScope Enum

ScopeType_e	Waveform scope type

- Trt Enums

RelationDirType_e	Apply to: Transaction Fsdb Relation Direction Type
CallStackType_e	Apply to: Transaction Fsdb Call Stack Type
TrtType_e	Apply to: Transaction Fsdb Transaction Traverse Type

- Val/VCT Enums

VctFormat_e	`VctFormat_e` used in returning vct format and the input of vct value format.
ValFormat_e	`ValFormat_e` used in returning value format and the input of waveform value (vct value, attribute value) format.

TrScope Enum

ScopeType_e

class waveform.**ScopeType_e**

Waveform scope type

Apply to:

System Verilog type Scope

SvModule = 0

SvTask = 1

SvFunction = 2

SvBegin = 3

SvFork = 4

SvGenerate = 5

SvInterface = 6

SvInterfacePort = 7

SvModport = 8

SvModportPort = 9

Apply to – VHDL type Scope

VhArchitecture = 10

VhProcedure = 11

VhFunction = 12

VhProcess = 13

VhBlock = 14

VhGenerate = 15

Apply to – System C type Scope

ScModule = 16

Apply to – spice type Scope

Spice = 17

Apply to – Power Type Scope

PwScope = 18

PwDomain = 19

PwSupplySet = 20

PwStateTable = 21

PwStateGroup = 22

PwSwitch = 23

PwIsoStrategy = 24

PwRetStrategy = 25

PwLsStrategy = 26

Apply to – Unknown type Scope

Unknown = 27

Trt Enums

RelationDirType_e

class waveform.**RelationDirType_e**

Apply to: Transaction Fsdb Relation Direction Type

Master = 0

Slave = 1

CallStackType_e

class waveform.**CallStackType_e**

Apply to: Transaction Fsdb Call Stack Type

Begin = 0

End = 1

TrtType_e

class waveform**.TrtType_**e

Apply to: Transaction Fsdb Transaction Traverse Type

Message = 0

Transaction = 1

Action = 2

Group = 3

Val/VCT Enums

VctFormat_e

class waveform.**VctFormat_e**

`VctFormat_e` is used in returning vct format and the input of vct value format.

BinStrVal: string of binary format (e.g., “1111” for 4’d15)

OctStrVal: string of octal format (e.g., “17” for 4’d15)

DecStrVal: string of decimal format (e.g., “15” for 4’d15)

HexStrVal: string of hex format (e.g., “f” 10/29/23for 4’d15)

SintVal: signed integer type (e.g., -1 for 4’d15)

UintVal: unsigned integer type (e.g., 15 for 4’d15)

RealVal: double type (e.g., -1.234E+01)

StringVal: ASCII string type (e.g., Synopsys)

EnumStrVal:string of enum literal (e.g., “R” for 0 in enum {R, G, B})

Sint64Val: signed 64-bit integer type (e.g., -1 for 64’d15)

Uint64Val: unsigned 64-bit integer type (e.g., 15 for 64’d15)

ObjTypeVal:use the given vct value’s default format.

BinStrVal = 0

OctStrVal = 1

DecStrVal = 2

HexStrVal = 3

SintVal = 4

UintVal = 5

RealVal = 6

StringVal = 7

EnumStrVal = 8

Sint64Val = 9

Uint64Val = 10

ObjTypeVal = 11

ValFormat_e

class waveform.**ValFormat_e**

`ValFormat_e` used in returning value format and the input of waveform value(vct value, attribute value) format.

BinStrVal: string of binary format (e.g., “1111” for 4’d15)

OctStrVal: string of octal format (e.g., “17” for 4’d15)

DecStrVal: string of decimal format (e.g., “15” for 4’d15)

HexStrVal: string of hex format (e.g., “f” for 4’d15)

SintVal: signed integer type (e.g., -1 for 4’d15)

UintVal: unsigned integer type (e.g., 15 for 4’d15)

RealVal: double type (e.g., -1.234E+01)

StringVal: ASCII string type (e.g., Synopsys)

EnumStrVal:string of enum literal (e.g., “R” for 0 in enum {R, G, B})

Sint64Val: signed 64-bit integer type (e.g., -1 for 64’d15)

Uint64Val: unsigned 64-bit integer type (e.g., 15 for 64’d15)

ObjTypeVal:use the given waveform value’s default format.

BinStrVal = 0

OctStrVal = 1

DecStrVal = 2

HexStrVal = 3

SintVal = 4

UintVal = 5

RealVal = 6

StringVal = 7

EnumStrVal = 8

Sint64Val = 9

Uint64Val = 10

ObjTypeVal = 11

Feedback

L0 APIs

File

Function list

open()	Open Wavefscopeorm file.
close()	Close the Waveform file.
is_fsdb()	Check if the given file is FSDB file.

Example:

Following is an example showing how to init and start with a fsdb file named `CPU.fsdb`.

example.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform

npisys.init(sys.argv)

fileName = "CPU.fsdb"
fileHandle = waveform.open(fileName)
if fileHandle is None:
  print("open file failed")
ret = waveform.is_fsdb(fileName)
if ret is True:
  print("this is FSDB")

waveform.close(fileHandle)
npisys.end()
```

Result:

```
this is FSDB
```

Waveform Function

waveform.open(*name*)

Open Waveform file

Parameters:

- name – file name

Returns:

- File object if success.
- None if fail.

Return type: FileHandle

waveform.close(*file*)

Close waveform file

Parameters:

- file – FileHandle

Returns:

- True if success.
- False if fail.

Return type: bool

Examples

```
>>> waveform.close(file)
```

waveform.is_fsdb(*name*)

Check if the given file is FSDB file

Parameters:

- name – file name

Returns:

- True if it is FSDB.
- False if it is not FSDB.

Return type: bool

Examples

```
>>> print(waveform.is_fsdb("CPU.fsdb"))
True
```

Feedback

class waveform.FileHandle(*fileObj*)

FileHandle Function list

min_time()	Get minimum time of file object.
max_time()	Get maximum time of file object.
name()	Get name of file object.
scale_unit()	Get scale unit of file object.
dump_off_range()	Get dump off range of file object.
has_seq_num()	Check if file object has sequence number.
is_completed()	Check if file object is completed.
has_glitch()	Check if file object has glitch.
has_assertion()	Check if file object has assertion type signal.
has_force_tag()	Check if file object has force tag.
has_reason_code()	Check if file object has reason code.
has_power_info()	Check if file object has power information.
version()	Get the file version.
sim_date()	Get the simulation date.
has_gate_tech()	Check if file object has gate technology.
top_scope_list()	Get top scope list.
top_sig_list()	Get top signal list.
add_to_sig_list(signal)	Add a signal of interest into the pending load list.
reset_sig_list()	Reset the pending load list.
load_vc_by_range(start, end)	For those signals in the signal list, load their value changes in the specified time range into memory.
unload_vc()	Unload value changes from memory that are already loaded.
scope_by_name(name[, scope])	Get a scope object with the specified name.

Feedback

sig_by_name(name[, scope])	Get a signal object with the specified name.
update()	Update current file object.
top_tr_scope_list()	Get top transaction scope list.
tr_scope_by_name(name[, trScope])	Get a transaction scope object with the specified name.
stream_by_name(name[, trScope])	Get a stream object wwith the specified name.
add_to_stream_list(stream)	Add a stream of interest into the pending load list.
reset_stream_list()	Reset the pending load list.
load_trans()	For those streams in the pending load list, load their transactions into memory.
unload_trans()	Unload transactions from memory that are already loaded.
trt_by_id(id)	Create a transaction traverse object by a transaction ID.
relation_list()	Get relation list.

Example:

file.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
# file L0 API
def test():
fileName = "CPU.fsdb"
fileHandle = waveform.open(fileName)
      ret = waveform.is_fsdb(fileName)
      if ret:
      print("this is FSDB")
      time = fileHandle.min_time()
          print("min time:"+str(time))
      time = fileHandle.max_time()
          print("max time:"+str(time))
          print("scale unit: " + fileHandle.scale_unit())
      if fileHandle.dump_off_range() is None:
          print("No dump off range")
```

```
else:
    print("dump_off_range: " + fileHandle.dump_off_range())
hasSeq = fileHandle.has_seq_num()
    print("has_seq_num: " + str(hasSeq))
boolVal = fileHandle.is_completed()
    print("is_completed: " + str(boolVal))
boolVal = fileHandle.has_glitch()
    print("has_glitch: " + str(boolVal))
boolVal = fileHandle.has_assertion()
    print("has_assertion: " + str(boolVal))
boolVal = fileHandle.has_force_tag()
    print("has_force_tag: " + str(boolVal))
boolVal = fileHandle.has_reason_code()
    print("has_reason_code: " + str(boolVal))
boolVal = fileHandle.has_power_info()
    print("has_power_info: " + str(boolVal))
    print("version: " + fileHandle.version())
    print("sim_date: " + fileHandle.sim_date())
boolVal = fileHandle.has_gate_tech()
    print("has_gate_tech: " + str(boolVal))
waveform.close(fileHandle)
if __name__ == '__main__':
orig_stdout = sys.stdout
f = open('file.log', 'w')
sys.stdout = f
npisys.init(sys.argv)
test()
npisys.end()
sys.stdout = orig_stdout
f.close()
```

Result: file.log

```
this is FSDB
min time:0
max time:14000
scale unit: 1ns
No dump off range
has_seq_num: True
is_completed: True
has_glitch: True
has_assertion: False
has_force_tag: False
has_reason_code: False
has_power_info: False
version: 4.3
sim_date: Tue Jun 8 17:40:56 2010
has_gate_tech: False
```

update()

Update current file object.

Returns: True if success. False if fail.

Return type: bool

Examples

```
>>> print(file.update())
True
```

min_time()

Get minimum time of file object.

Returns: Time if success. None if fail.

Return type: int

Examples

```
>>> print(file.min_time())
0
```

max_time()

Get maximum time of file object.

Returns: Time if success. None if fail.

Return type: int

Examples

```
>>> print(file.max_time())
14000
```

name()

Get name of file object.

Returns: File name if success. None if fail.

Return type: str

Examples

```
>>> print(file.name())
./myFolder/CPU.fsdb
```

scale_unit()

Get scale unit of file object.

Returns: Scale unit if success. None if fail.

Return type: str

Examples

```
>>> print(file.scale_unit())
1ns
```

dump_off_range()

Get dump off range of file object.

Returns: Dump off range if success. None if fail.

Return type: str

Examples

```
>>> print(file.dump_off_range())
None
```

has_seq_num()

Check if file object has sequence number.

Returns: True if it has sequence number. False if it does not have sequence number. None if fail.

Return type: bool

Examples

```
>>> print(file.has_seq_num())
True
```

is_completed()

Check if file object is completed.

Returns: True if it is completed. False if it is not completed. None if fail.

Return type: bool

Examples

```
>>> print(file.is_completed())
True
```

has_glitch()

Check if file object has glitch.

Returns: True if it has glitch. False if it does not have glitch. None if fail.

Return type: bool

Examples

```
>>> print(file.has_glitch())
True
```

has_assertion()

Check if file object has assertion type signal.

Returns: True if it has assertion type signal. False if it does not have assertion type signal. None if fail.

Return type: bool

Examples

```
>>> print(file.has_assertion())
False
```

has_force_tag()

Check if file object has force tag.

Returns: True if it has force tag. False if it does not have force tag. None if fail.

Return type: bool

Examples

```
>>> print(file.has_force_tag())
False
```

has_reason_code()

Check if file object has reason code.

Returns: True if it has reason code. False if it does not have reason code. None if fail.

Return type: bool

Examples

```
>>> print(file.has_reason_code())
False
```

has_power_info()

Check if file object has power information.

Returns: True if it has power information. False if it does not have power information. None if fail.

Return type: bool

Examples

```
>>> print(file.has_power_info())
False
```

version()

Get the file version.

Returns: File version if success. None if fail.

Return type: str

Examples

```
>>> print(file.version())
4.3
```

sim_date()

Get the simulation date.

Returns:Simulation date if success. None if fail.

Return type: str

Examples

```
>>> print(file.sim_date())
Tue Jun 8 17:40:56 2010
```

has_gate_tech()

Check if file object has gate technology.

Returns:True if it has gate technology. False if it does not have gate technology. None if fail.

Return type: bool

Examples

```
>>> print(file.has_gate_tech())
False
```

top_scope_list()

Get top scope list.

Returns:List of top scope if success. Empty list if fail.

Return type: ScopeHandle list

Examples

```
scope_list = file.top_scope_list()
for scope in scope_list:
```

```
  print(scope.name())
tb_CPUsystem
dump_fsdb
```

top_sig_list()

Get top signal list.

Returns:List of top signal if success. Empty list if fail.

Return type: list

Examples

```
topSigFileHandle = waveform.open("top_sig.fsdb")
signalList = topSigFileHandle.top_sig_list()
for signal in signalList:
  print(signal.name())
realSig
```

add_to_sig_list(*signal*)

Add a signal of interest into the pending load list.

Parameters: **signal** - The target signal object.

Returns:True if success. False if fail.

Return type: bool

Examples

```
>>> print(file.add_to_sig_list(sig))
True
```

reset_sig_list()

Reset the pending load list.

Returns:True if success. False if fail.

Return type: bool

Examples

```
>>> print(file.reset_sig_list())
True
```

load_vc_by_range(*start*, *end*)

For those signals in the signal list, load their value changes in the specified time range into memory.

Parameters: **start** - The start time to load vc.

end - The end time to load vc.

Returns:True if success. False if fail.

Return type: bool

Examples

```
>>> print(file.load_vc_by_range(10, 2000))
True
```

unload_vc()

Unload value changes from memory that are already loaded.

Returns:True if success. False if fail.

Return type: bool

Examples

```
>>> print(file.unload_vc())
True
```

scope_by_name(*name*, *scope=None*)

Unload value changes from memory that are already loaded.

Parameters: **name** - The string representing the scope name (e.g. top.subscope1.subscope2). (The scope delimiter is fixed to ".")

scope - A scope object for localizing the search space. (If the scope object is null, this function searches the scope name from the root space.)

Returns:Scope object if success. None if fail.

Return type: ScopeHandle

Examples

```
>>> scope = fileHandle.scope_by_name("tb_CPUsystem.i_BJsource")
print(scope.name())
i_BJsource
```

sig_by_name(*name*, *scope=None*)

Get a signal object with the specified name.

Parameters: **name** - The string representing the signal name (e.g. top.subscope1.sig). (The scope delimiter is fixed to ".")

scope - A scope object for localizing the search space. (If the scope object is null, this function searches the scope name from the root space.)

Returns:Signal object if success. None if fail.

Return type: SigHandle

Examples

```
>>> sigName = "tb_CPUsystem.i_BJsource.Card_temp"
signal = fileHandle.sig_by_name(sigName)
print(signal.name())
Card_temp
```

top_tr_scope_list()

Get top transaction scope list.

Returns:List of top transaction scope if success. Empty list if fail.

Return type: TrScopeHandle list

Examples

```
>>> fileHandle = waveform.open("attr.fsdb")
tr_scope_list = fileHandle.top_tr_scope_list()
for trScope in tr_scope_list:
print(trScope.name())
```

tr_scope_by_name(*name*, *trScope=None*)

Get a transaction scope object with the specified name.

Parameters: **name** - The string representing the transaction scope name (e.g. top.subscope1.subscope2). (The transaction scope delimiter is fixed to ".")

trScope - A transaction scope object for localizing the search space. (If the transaction scope object is null, this function searches the scope name from the root space.)

Returns:Transaction Scope object if success. None if fail.

Return type: TrScopeHandle

Examples

```
>>> fileHandle = waveform.open("attr.fsdb")
trScope = fileHandle.tr_scope_by_name("$trans_root.scopeName")
print(trScope.name())
scopeName
```

stream_by_name(*name*, *trScope=None*)

Get a stream object wwith the specified name.

Parameters: **name** - The string representing the stream name (e.g. top.subscope1.stream). (The scope delimiter is fixed to ".")

trScope - A transaction scope object for localizing the search space. (If the transaction scope object is null, this function searches the scope name from the root space.)

Returns:Stream object if success. None if fail.

Return type: StreamHandle

Examples

```
>>> fileHandle = waveform.open("attr.fsdb")
stream = fileHandle.stream_by_name("$trans_root.streamName")
print(stream.name())
streamName
```

add_to_stream_list(*stream*)

Add a stream of interest into the pending load list.

Parameters: **stream** - The target stream object.

Returns:True if success. False if fail.

Return type: bool

Examples

```
>>> fileHandle = waveform.open("attr.fsdb")
stream = fileHandle.stream_by_name("$trans_root.streamName")
fileHandle.add_to_stream_list(stream)
```

reset_stream_list()

Reset the pending load list.

Returns:True if success. False if fail.

Return type: bool

Examples

```
>>> fileHandle = waveform.open("attr.fsdb")
stream = fileHandle.stream_by_name("$trans_root.streamName")
fileHandle.add_to_stream_list(stream)
fileHandle.reset_stream_list()
```

load_trans()

For those streams in the pending load list, load their transactions into memory.

Returns:True if success. False if fail.

Return type: bool

Examples

Feedback

```
>>> fileHandle = waveform.open("attr.fsdb")
stream = fileHandle.stream_by_name("$trans_root.streamName")
fileHandle.add_to_stream_list(stream)
fileHandle.load_trans()
```

unload_trans()

Unload transactions from memory that are already loaded.

Returns:True if success. False if fail.

Return type: bool

Examples

```
>>> fileHandle = waveform.open("attr.fsdb")
stream = fileHandle.stream_by_name("$trans_root.streamName")
fileHandle.add_to_stream_list(stream)
fileHandle.load_trans()
fileHandle.unload_trans()
fileHandle.reset_stream_list()
```

trt_by_id(*id*)

Create a transaction traverse object by a transaction ID.

Parameters: **id -** The target transaction ID.

Returns:Transaction traverse object if success. None if fail.

Return type: TrtHandle

Examples

```
>>> fileHandle = waveform.open("attr.fsdb")
trt = fileHandle.trt_by_id( 5 )
```

relation_list()

Get relation list.

Returns:List of relation if success. Empty list if fail.

Return type: RelationHandle list

Examples

```
>>> fileHandle = waveform.open("attr.fsdb")
rel_list = fileHandle.relation_list()
for rel in rel_list:
  print(rel.name())
parent-child
belong-to
succ-predecessor
annotate-to
```

Feedback

Transaction Scope

class waveform.TrScopeHandle(*trScopeObj*)

TrScopeHandle Function list

name()	Get the name of transaction scope object.
full_name()	Get the full name of transaction scope object.
def_name()	Get the defined name of transaction scope object.
type([isEnum])	Get the scope type of transaction scope object.
parent()	Get the parent transaction scope.
child_tr_scope_list()	Get the child transaction scope list.
stream_list()	Get the stream list.
file()	Get the file object.
attr_count()	Get the attribute count of corresponding transaction scope.
attr_value(ith[, format])	Get the i-th attribute value of corresponding transaction scope.
attr_name(ith)	Get the i-th attribute name of corresponding transaction scope.
attr_is_hidden(ith)	Get the i-th attribute is hidden or not of corresponding transaction scope.
attr_is_tag(ith)	Get the i-th attribute is tag or not of corresponding transaction scope.
attr_value_format(ith)	Get the i-th attribute's value default format of corresponding transaction scope.

Example:

trScope.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisy
from pynpi import waveform
npisys.init(sys.argv)
def iter_sub_tr_scope_to_list(all_list, tr_scope):
```

Feedback

```
    child_list = tr_scope.child_tr_scope_list()
    all_list.extend( child_list )
    for child_tr_scope in child_list:
    iter_sub_tr_scope_to_list( all_list , child_tr_scope )
def print_tr_scope( scope ):
    print("tr scope file name: ", (scope.file()).name())
    print("tr scope name: ", scope.name())
    print("tr scope full name: ",scope.full_name())
    print("tr scope parent name: ", scope.parent().name())
    print("tr scope type: ", scope.type(False))
    num = scope.attr_count()
    print("tr_scope attr_count: ", num)
    for ith in range(num):
        print(ith, " scope attr name: ", scope.attr_name(ith))
        print("Property: Hidden? ", scope.attr_is_hidden(ith),
 " Tag? " , scope.attr_is_tag(ith), " default format: " ,
 scope.attr_value_format(ith) )
    if __name__ == '__main__':
    orig_stdout = sys.stdout
    f = open('tr.log', 'w')
    sys.stdout = f
    fileHandle = waveform.open("attr.fsdb")
    if not fileHandle:
        print(" open waveform fail." )
        npisys.end()
        sys.stdout = orig_stdout
        f.close()
        quit()
    tr_scope_list = fileHandle.top_tr_scope_list()
    all_list = []
    all_list.extend(tr_scope_list)
    for tr_scope in tr_scope_list:
    iter_sub_tr_scope_to_list( all_list , tr_scope )
    tr_scope = fileHandle.tr_scope_by_name("$trans_root.scopeName")
    if tr_scope in all_list:
        print_tr_scope(tr_scope)
    waveform.close(fileHandle)
    npisys.end()
    sys.stdout = orig_stdout
    f.close()
```

Result: tr.log

```
tr scope file name: ./myFolder/attr.fsdb
tr scope name: scopeName
tr scope full name: $trans_root.scopeName
tr scope parent name: $trans_root
tr scope type: npiFsdbScopeSvModule
tr_scope attr_count: 24
0 scope attr name: attrByte8(Bin)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
1 scope attr name: attrByte8(Oct)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
```

```
2 scope attr name: attrByte8(Dec)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
3 scope attr name: attrByte8(Hex)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
4 scope attr name: attrByte8(Uns)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
5 scope attr name: attrByte8(no radix)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
6 scope attr name: attrInt16(Bin)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
7 scope attr name: attrInt16(Oct)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
8 scope attr name: attrInt16(Dec)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
9 scope attr name: attrInt16(Hex)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
10 scope attr name: attrInt16(Uns)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
11 scope attr name: attrInt16(no radix)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
12 scope attr name: attrInt32(Bin)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
13 scope attr name: attrInt32(Oct)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
14 scope attr name: attrInt32(Dec)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
15 scope attr name: attrInt32(Hex)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
16 scope attr name: attrInt32(Uns)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
17 scope attr name: attrInt32(no radix)
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
18 scope attr name: attrInt64(Bin)
Property: Hidden? False Tag? False default format: ValFormat_e.Sint64Val
19 scope attr name: attrInt64(Oct)
Property: Hidden? False Tag? False default format: ValFormat_e.Sint64Val
20 scope attr name: attrInt64(Dec)
Property: Hidden? False Tag? False default format: ValFormat_e.Sint64Val
21 scope attr name: attrInt64(Hex)
Property: Hidden? False Tag? False default format: ValFormat_e.Sint64Val
22 scope attr name: attrInt64(Uns)
Property: Hidden? False Tag? False default format: ValFormat_e.Sint64Val
23 scope attr name: attrInt64(no radix)
Property: Hidden? False Tag? False default format: ValFormat_e.Sint64Val
```

name()

Get the name of transaction scope object.

Returns: The transaction scope name if success. None if fail.

Return type: str

Examples

```
>>> As trScope.py shown
```

full_name()

Get the full name of transaction scope object.

Returns: The full name of transaction scope object if success. None if fail.

Return type: str

Examples

```
>>> As trScope.py shown
```

def_name()

Get the defined name of transaction scope object.

Returns: The defined name of transaction scope object if success. None if fail.

Return type: str

Examples

```
>>> As trScope.py shown
```

type(*isEnum=True*)

Get the scope type of transaction scope object.

Parameters: **isEnum** - Specify the type in enum or string.

Returns: If isEnum is True: The scope type if success. None if fail.

If isEnum is False: The string_value if success. None if fail.

Return type: ScopeType_e/str

Examples

```
>>> As trScope.py shown
```

parent()

Get the parent transaction scope.

Returns: The parent transaction scope if success. None if fail.

Return type: TrScopeHandle

Examples

```
>>> As trScope.py shown
```

Feedback

child_tr_scope_list()

Get the child transaction scope list.

Returns: The list of child transaction scope if success. Empty list if fail.

Return type: TrScopeHandle list

Examples

```
>>> As trScope.py shown
```

stream_list()

Get the stream list.

Returns: The stream list if success. Empty list if fail.

Return type: StreamHandle list

Examples

```
>>> As trScope.py shown
```

file()

Get the file object.

Returns: The file object if success. None if fail.

Return type: FileHandle

Examples

```
>>> As trScope.py shown
```

attr_count()

Get the attribute count of corresponding transaction scope.

Returns: The attribute count of corresponding transaction scope.

Return type: int

Examples

```
>>> As trScope.py shown
```

attr_value(*ith*, *format=<ValFormat_e.ObjTypeVal: 11>*)

Get the i-th attribute value of corresponding transaction scope.

Parameters: *ith* - The attribute index.

format - **ValFormat_e**.

Returns: Value with the specified format.

Examples

```
>>> As trScope.py shown
```

attr_name(*ith*)

Get the i-th attribute name of corresponding transaction scope.

Parameters: **ith** - The attribute index.

Returns: Transaction scope attribute name if success. None if fail.

Return type: str

Examples

```
>>> As trScope.py shown
```

attr_is_hidden(*ith*)

Get the i-th attribute is hidden or not of corresponding transaction scope.

Parameters: *ith* - The attribute index.

Returns: True if i-th attribute is hidden of corresponding transaction scope. False if i-th attribute is not hidden of corresponding transaction scope. None if fail.

Return type: bool

Examples

```
>>> As trScope.py shown
```

attr_is_tag(*ith*)

Get the i-th attribute is tag or not of corresponding transaction scope.

Parameters: **ith** - The attribute index.

Returns: True if i-th attribute is tag of corresponding transaction scope. False if i-th attribute is not tag of corresponding transaction scope. None if fail.

Return type: bool

Examples

```
>>> As trScope.py shown
```

attr_value_format(*ith*)

Get the i-th attribute's value default format of corresponding transaction scope.

Parameters: **ith** - The attribute index.

Returns: Format ValFormat_e if success. None if fail.

Return type: ValFormat_e

Examples

```
>>> As trScope.py shown
```

Stream

class waveform.StreamHandle(*streamObj*)

StreamHandle Function list

name()	Get the name of stream object.
full_name()	Get the full name of stream object.
tr_scope()	Get the transaction scope object whose corresponding transaction scope contains this stream.
attr_count()	Get the attribute count of corresponding stream.
attr_value(ith[, format])	Get the i-th attribute value of corresponding stream.
attr_name(ith)	Get the i-th attribute name of corresponding stream.
attr_is_hidden(ith)	Get the i-th attribute is hidden or not of corresponding stream.
attr_is_tag(ith)	Get the i-th attribute is tag or not of corresponding stream.
attr_value_format(ith)	Get the i-th attribute's value default format of corresponding stream.
create_trt()	Create a transaction traverse object for a specific stream.

Example:

Stream.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(sys.argv)
def iter_sub_tr_scope_to_list(all_list, tr_scope):
    child_list = tr_scope.child_tr_scope_list()
```

Feedback

```
    all_list.extend( child_list )
    for child_tr_scope in child_list:
    iter_sub_tr_scope_to_list( all_list , child_tr_scope )
def iter_stream_to_list(all_stream_list, tr_scope):
    all_stream_list.extend( tr_scope.stream_list() )
def print_stream( stream ):
    print("stream full name: ", stream.full_name()," stream scope is: ",
 stream.tr_scope().name() )
    num = stream.attr_count()
    print("stream attr_count: ", num)
    for ith in range(num):
        print(ith, " stream attr name: ", stream.attr_name(ith), " val:
 ", stream.attr_value(ith))
        print("Property: Hidden? ", stream.attr_is_hidden(ith),
 " Tag? " , stream.attr_is_tag(ith), " default format: " ,
 stream.attr_value_format(ith) )
    if __name__ == '__main__':
    orig_stdout = sys.stdout
    f = open('stream.log', 'w')
    sys.stdout = f
    fileHandle = waveform.open("tr_wave/attr.fsdb")
    if fileHandle is None:
        print(" open waveform fail." )
        npisys.end()
        sys.stdout = orig_stdout
        f.close()
        quit()
    tr_scope_list = fileHandle.top_tr_scope_list()
    all_list = []
    all_list.extend(tr_scope_list)
    for tr_scope in tr_scope_list:
    iter_sub_tr_scope_to_list( all_list , tr_scope )
    tr_scope_name_list = []
    all_stream_list = []
    for tr_scope in all_list:
    iter_stream_to_list(all_stream_list, tr_scope)
    stream = fileHandle.stream_by_name("$trans_root.streamName")
    if stream in all_stream_list:
        print( stream.name() , " in stream list ")
    print_stream(stream)
    fileHandle.add_to_stream_list(stream)
    fileHandle.load_trans()
    trt = stream.create_trt()
    fileHandle.unload_trans()
    fileHandle.reset_stream_list()
    waveform.close(fileHandle)
    npisys.end()
    sys.stdout = orig_stdout
    f.close()
```

Result: stream.log

```
streamName in stream list
stream full name: $trans_root.streamName stream scope is: $trans_root
stream attr_count: 7
0 stream attr name: streamAttrName val: 10
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
1 stream attr name: streamAttrName val: 10
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
2 stream attr name: streamAttrName val: Attr_value_str2
Property: Hidden? False Tag? False default format: ValFormat_e.StringVal
3 stream attr name: streamAttrName8 val: 1
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
4 stream attr name: streamAttrName16 val: 19
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
5 stream attr name: streamAttrName32 val: 20
Property: Hidden? False Tag? False default format: ValFormat_e.SintVal
6 stream attr name: streamAttrName64 val: 30
Property: Hidden? False Tag? False default format: ValFormat_e.Sint64Val
```

name()

Get the name of stream object.

Returns: The stream name if success. None if fail.

Return type: str

Examples

```
>>> As Stream.py shown
```

full_name()

Get the full name of stream object.

Returns: The stream full name if success. None if fail.

Return type: str

Examples

```
>>> As Stream.py shown
```

tr_scope()

Get the transaction scope object whose corresponding transaction scope contains this stream.

Returns: The transaction scope object if success. None if fail.

Return type: TrScopeHandle

Examples

```
>>> As Stream.py shown
```

Feedback

attr_count()

Get the attribute count of corresponding stream.

Returns: The attribute count of corresponding stream.

Return type: int

Examples

```
>>> As Stream.py shown
```

attr_value(*ith*,*format*=<*ValFormat_e.ObjTypeVal: 11*>)

Get the i-th attribute value of corresponding stream.

Parameters: *ith* - The attribute index.

format - ValFormat_e.

Returns: Value with the specified format.

Examples

```
>>> As Stream.py shown
```

attr_name(*ith*)

Get the i-th attribute name of corresponding stream.

Parameters: **ith** - The attribute index.

Returns: Stream attribute name if success. None if fail.

Return type: str

Examples

```
>>> As Stream.py shown
```

attr_is_hidden(*ith*)

Get the i-th attribute is hidden or not of corresponding stream.

Parameters: *ith* - The attribute index.

Returns: True if i-th attribute is hidden of corresponding stream. False if i-th attribute is not hidden of corresponding stream. None if fail.

Return type: bool

Examples

```
>>> As Stream.py shown
```

Feedback

attr_is_tag(*ith*)

Get the i-th attribute is tag or not of corresponding stream.

Parameters: **ith** - The attribute index.

Returns: True if i-th attribute is tag of corresponding stream. False if i-th attribute is not tag of corresponding stream. None if fail.

Return type: bool

Examples

```
>>> As Stream.py shown
```

attr_value_format(*ith*)

Get the i-th attribute's value default format of corresponding stream.

Parameters: **ith** - The attribute index.

Returns: Format ValFormat_e if success. None if fail.

Return type: ValFormat_e

Examples

```
>>> As Stream.py shown
```

create_trt()

Create a transaction traverse object for a specific stream.

Parameters: **ith** - The attribute index.

Returns: Transaction traverse object if success. None if fail.

Return type: TrtHandle

Examples

```
>>> As Stream.py shown
```

Transaction Traverse

class waveform.TrtHandle(*trtObj*)

TrtHandle Function list

time()	Get the begin and end time of corresponding transaction traverse object.

id()	Get the id of corresponding transaction traverse object.(Note: The transaction id is unique in an FSDB file, so users can use it to check if two transaction traverse objects pointing to the same transaction.)
name()	Get the name of corresponding transaction traverse object.
goto_next()	Increase the index of the transaction traverse object if possible.
goto_prev()	Decrease the index of the transaction traverse object if possible.
goto_first()	Move the index of the transaction traverse object to the first transaction if possible.
goto_time(time)	Change the index of the transaction traverse object to the last transaction at the specified time (according to begin time of transactions).
stream()	Get the corresponding stream object of the transaction traverse object.
type()	Get the transaction traverse type of the transaction traverse object.
attr_count()	Get the attribute count of corresponding transaction.
attr_value(ith[, format])	Get the i-th attribute value of corresponding transaction.
attr_name(ith: int)	Get the i-th attribute name of corresponding transaction.
attr_is_hidden(ith)	Get the i-th attribute is hidden or not of corresponding transaction.
attr_is_tag(ith)	Get the i-th attribute is tag or not of corresponding transaction.
attr_value_format(ith)	Get the i-th attribute's value default format of corresponding transaction.
expected_attr(ith)	Get the expected attribute index of corresponding attribute.
is_unfinished()	Get the transaction traverse object is unfinished of corresponding transaction.
release()	Free the transaction traverse handle.
related_trt_list(relation[, dir])	Get transaction traverse list (pointing to related transactions).
call_stack_count([type])	Get the number of call stack under current transaction.

Feedback

call_stack(ith[, type])	Get the i-th call stack value under current transaction.

Example:

trt.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(sys.argv)
def stream_trt_info( trt ):
print("\nstream_trt_info.")
    if trt.goto_first():
    while True:
        print("trt name: ", trt.name(), "trt id: ", trt.id(), "trt time:
 ", trt.time() )
    num = trt.attr_count()
    if 36 < num: #use the 36th's attr as example
        print("trt attr name: ", trt.attr_name(36), " attr value is : ",
 trt.attr_value(36))
        print("Property: Hidden? ", trt.attr_is_hidden(36), " Tag? " ,
 trt.attr_is_tag(36), " default format: " , trt.attr_value_format(36) )
    exnum = trt.expected_attr(36)
    if exnum:
        print("trt expected attr name: ", trt.attr_name(exnum), "
 expected attr value is : ", trt.attr_value(exnum))
    ret = trt.goto_next()
    if not ret:
    break
def trt_related_info( trt , rel_list ):
    print("\ntrt_related_info.")
    if trt.goto_time(5500000):
    while True:
    for rel in rel_list:
    trt_list = trt.related_trt_list( rel ,
 waveform.RelationDirType_e.Master)
    for rtrt in trt_list:
        print( "Dir Master" , trt.name()," related_trt is ", rtrt.name(),
 " relation: " , rel.name() )
    trt_list = trt.related_trt_list( rel ,
 waveform.RelationDirType_e.Slave)
    for rtrt in trt_list:
        print( "Dir Slave" , trt.name()," related_trt is ", rtrt.name(),
 " relation: " , rel.name() )
    ret = trt.goto_prev()
    if not ret:
    break
def dump_tr_call_stack( trt ):
```

```
    print("\ndump_tr_call_stack.")
    type_name = ["Message","Transaction","Action","Group"]
    if trt.goto_time(28000000):
    while True:
    type = trt.type()
    if type != None:
        print("trt type: ", type_name[type] )
        print("time: ", trt.time() ," unfinished? ",
trt.is_unfinished() )
    num = trt.call_stack_count()
    if num: # use num-1 as example
        print("call stack(begin): ", trt.call_stack(num-1) )
    num = trt.call_stack_count(waveform.CallStackType_e.End)
    if num: # use num-1 as example
        print("call stack(end): ", trt.call_stack(num-1,
waveform.CallStackType_e.End) )
    ret = trt.goto_next()
    if not ret:
    break
    if __name__ == '__main__':
    orig_stdout = sys.stdout
    f = open('trt.log', 'w')
    sys.stdout = f
    fileHandle = waveform.open("tr_wave/begin_end_call_stack.fsdb")
    if fileHandle is None:
        print(" open waveform fail." )
        npisys.end()
        sys.stdout = orig_stdout
        f.close()
        quit()
    rel_list = fileHandle.relation_list()
    trt = fileHandle.trt_by_id( 998 ) #use trt's id: 998 as example.
    stream = trt.stream()
    fileHandle.add_to_stream_list(stream)
    fileHandle.load_trans()
        print("trt stream name:" , stream.name())
    stream_trt_info(trt)
    trt_related_info(trt, rel_list)
    dump_tr_call_stack(trt)
    trt.release()
    fileHandle.unload_trans()
    fileHandle.reset_stream_list()
    waveform.close(fileHandle)
    npisys.end()
    sys.stdout = orig_stdout
    f.close()
```

Result: trt.log

```
trt stream name: svt_chi_request_wr_writenosnpptl_transaction
stream_trt_info.
trt name: svt_chi_request_wr_writenosnpptl_transaction trt id: 635 trt
 time: [1750000, 4500000]
```

Feedback

```
trt attr name: ccid attr value is : CCID_DATA_383_DOWN_TO_256
Property: Hidden? False Tag? False default format: ValFormat_e.StringVal
trt name: svt_chi_request_wr_writenosnpptl_transaction trt id: 888 trt
 time: [5500000, 12050000]
trt attr name: ccid attr value is : CCID_DATA_127_DOWN_TO_0
Property: Hidden? False Tag? False default format: ValFormat_e.StringVal
trt name: svt_chi_request_wr_writenosnpptl_transaction trt id: 998 trt
 time: [7900000, 15050000]
trt attr name: ccid attr value is : CCID_DATA_255_DOWN_TO_128
Property: Hidden? False Tag? False default format: ValFormat_e.StringVal
trt name: svt_chi_request_wr_writenosnpptl_transaction trt id: 1628 trt
 time: [21600000, 26300000]
trt attr name: ccid attr value is : CCID_DATA_511_DOWN_TO_384
Property: Hidden? False Tag? False default format: ValFormat_e.StringVal
trt name: svt_chi_request_wr_writenosnpptl_transaction trt id: 1702 trt
 time: [23550000, 27500000]
trt attr name: ccid attr value is : CCID_DATA_511_DOWN_TO_384
Property: Hidden? False Tag? False default format: ValFormat_e.StringVal
trt name: svt_chi_request_wr_writenosnpptl_transaction trt id: 1737 trt
 time: [23900000, 28450000]
trt attr name: ccid attr value is : CCID_DATA_511_DOWN_TO_384
Property: Hidden? False Tag? False default format: ValFormat_e.StringVal
trt name: svt_chi_request_wr_writenosnpptl_transaction trt id: 1958 trt
 time: [28400000, 32550000]
trt attr name: ccid attr value is : CCID_DATA_511_DOWN_TO_384
Property: Hidden? False Tag? False default format: ValFormat_e.StringVal
trt name: svt_chi_request_wr_writenosnpptl_transaction trt id: 2015 trt
 time: [29000000, 34650000]
trt attr name: ccid attr value is : CCID_DATA_511_DOWN_TO_384
Property: Hidden? False Tag? False default format: ValFormat_e.StringVal
trt_related_info.
Dir Slave svt_chi_request_wr_writenosnpptl_transaction related_trt is
 svt_chi_protocol_req_flit relation: parent-child
Dir Slave svt_chi_request_wr_writenosnpptl_transaction related_trt is
 svt_chi_protocol_dat_flit relation: parent-child
Dir Slave svt_chi_request_wr_writenosnpptl_transaction related_trt is
 svt_chi_protocol_rsp_flit relation: parent-child
Dir Slave svt_chi_request_wr_writenosnpptl_transaction related_trt is
 svt_chi_protocol_req_flit relation: parent-child
Dir Slave svt_chi_request_wr_writenosnpptl_transaction related_trt is
 svt_chi_protocol_dat_flit relation: parent-child
Dir Slave svt_chi_request_wr_writenosnpptl_transaction related_trt is
 svt_chi_protocol_dat_flit relation: parent-child
Dir Slave svt_chi_request_wr_writenosnpptl_transaction related_trt is
 svt_chi_protocol_dat_flit relation: parent-child
Dir Slave svt_chi_request_wr_writenosnpptl_transaction related_trt is
 svt_chi_protocol_dat_flit relation: parent-child
Dir Slave svt_chi_request_wr_writenosnpptl_transaction related_trt is
 svt_chi_protocol_rsp_flit relation: parent-child
dump_tr_call_stack.
trt type: Transaction
time: [23900000, 28450000] unfinished? False
call stack(begin): ['top.sv', 235]
```

```
call stack(end): ['top.sv', 235]
trt type: Transaction
time: [28400000, 32550000] unfinished? False
call stack(begin): ['top.sv', 235]
call stack(end): ['top.sv', 235]
trt type: Transaction
time: [29000000, 34650000] unfinished? False
call stack(begin): ['top.sv', 235]
call stack(end): ['top.sv', 235]
```

time()

Get the begin and end time of corresponding transaction traverse object.

Returns: [begin_time, end_time] if success. None if fail.

Return type: int list

Examples

```
>>> As try.py shown
```

id()

Get the ID of corresponding transaction traverse object. (Note: The transaction ID is unique in an FSDB file, so users can use it to check if two transaction traverse objects pointing to the same transaction.)

Returns: The ID of the corresponding transaction; -1 if fail.

Return type: int

Examples

```
>>> As try.py shown
```

name()

Get the name of corresponding transaction traverse object.

Returns: Transaction name if success. None if fail.

Return type: str

Examples

```
>>> As try.py shown
```

goto_next()

Increase the index of the transaction traverse object if possible.

Returns: True if success. False if fail.

Return type: bool

Examples

```
>>> As try.py shown
```

goto_prev()

Decrease the index of the transaction traverse object if possible.

Returns: True if success. False if fail.

Return type: bool

Examples

```
>>> As try.py shown
```

goto_first()

Move the index of the transaction traverse object to the first transaction if possible.

Returns: True if success. False if fail.

Return type: bool

Examples

```
>>> As try.py shown
```

goto_time(*time*)

Change the index of the transaction traverse object to the last transaction at the specified time (according to begin time of transactions).

Parameters: **time** - The target time.

Returns: True if success. False if fail.

Return type: bool

Examples

```
>>> As try.py shown
```

stream()

Get the corresponding stream object of the transaction traverse object.

Returns: The stream object if success. None if fail.

Return type: StreamHandle

Examples

```
>>> As try.py shown
```

type()

Get the transaction traverse type of the transaction traverse object.

Returns: The transaction traverse type if success. None if fail.

Return type: TrtType_e

Examples

```
>>> As try.py shown
```

attr_count()

Get the attribute count of corresponding transaction.

Returns: The attribute count of corresponding transaction.

Return type: int

Examples

```
>>> As try.py shown
```

attr_value(*ith*, *format=<ValFormat_e.ObjTypeVal: 11>*)

Get the i-th attribute value of corresponding transaction.

Parameters: *ith* - The attribute index.

format - ValFormat_e.

Returns: Value with the specified format.

Examples

```
>>> As try.py shown
```

attr_name(*ith: int*)

Get the i-th attribute name of corresponding transaction.

Parameters: *ith* - The attribute index.

Returns: Transaction attribute name if success. None if fail.

Return type: str

Examples

```
>>> As try.py shown
```

attr_is_hidden(*ith*)

Get the i-th attribute is hidden or not of corresponding transaction.

Parameters: **ith** - The attribute index.

Returns: True if i-th attribute is hidden of corresponding transaction. False if i-th attribute is not hidden of corresponding transaction. None if fail.

Return type: bool

Examples

```
>>> As try.py shown
```

attr_is_tag(*ith*)

Get the i-th attribute is tag or not of corresponding transaction.

Parameters: **ith** - The attribute index.

Returns: True if i-th attribute is tag of corresponding transaction. False if i-th attribute is not tag of corresponding transaction. None if fail.

Return type: bool

Examples

```
>>> As try.py shown
```

attr_value_format(*ith*)

Get the i-th attribute's value default format of corresponding transaction.

Parameters: *ith* - The attribute index.

Returns: Format ValFormat_e if success. None if fail.

Return type: **ValFormat_e**

Examples

```
>>> As try.py shown
```

expected_attr(*ith*)

Get the expected attribute index of corresponding attribute.

Parameters: **ith** - The attribute index.

Returns: The expected attribute index of the corresponding attribute if success. None if fail.

Return type: int

Examples

```
>>> As try.py shown
```

Feedback

is_unfinished()

Get the transaction traverse object is unfinished of corresponding transaction.

Returns: True if the transaction is unfinished. False if the transaction is finished.

Return type: bool

Examples

```
>>> As try.py shown
```

release()

Free the transaction traverse handle.

Examples

```
>>> As try.py shown
```

related_trt_list(*relation*, *dir=<RelationDirType_e.Master: 0>*)

Get transaction traverse list (pointing to related transactions).

Parameters: **relation** - The target relation.

dir - RelationDirType_e.

Returns: The list of related transaction if success. Empty list if fail.

Return type: TrtHandle list

Examples

```
>>> As try.py shown
```

call_stack_count(*type=<CallStackType_e.Begin: 0>*)

Get the number of call stack under current transaction.

Parameters: **type - CallStackType_e**.

Returns: The number of call stack under current transaction.

Return type: int

Examples

```
>>> As try.py shown
```

call_stack(*ith*, *type=<CallStackType_e.Begin: 0>*)

Get the i-th call stack value under current transaction.

Parameters: **ith** - The index of call stack value under current transaction.

Feedback

type - CallStackType_e.

Returns: [filename(string), lineNum(int)] if success. None if fail.

Return type: list

Examples

```
>>> As try.py shown
```

Relation

class waveform.RelationHandle(*relationObj*)

RelationHandle Function list

name()	Get the name of relation.

Example:

rel.py:

```
import sys
import os
rel_lib_path = os.environ["VERDI_HOME"] + "/share/NPI/python"
sys.path.append(os.path.abspath(rel_lib_path))
from pynpi import npisys
from pynpi import waveform
npisys.init(sys.argv)
    if __name__ == '__main__':
    orig_stdout = sys.stdout
    f = open('rel.log', 'w')
    sys.stdout = f
    fileHandle = waveform.open("relation.fsdb")
    if not fileHandle:
        print(" open waveform fail." )
        npisys.end()
        sys.stdout = orig_stdout
        f.close()
        quit()
    rel_list = fileHandle.relation_list()
    for rel in rel_list:
        print("rel name: ", rel.name())
    waveform.close(fileHandle)
    npisys.end()
    sys.stdout = orig_stdout
    f.close()
```

Result:

```
rel name: parent-child
rel name: belong-to
rel name: succ-predecessor
rel name: annotate-to
rel name: couple
```

name()

Get the name of relation.

Returns: Relation name if success. None if fail.

Return type: str

Examples

```
>>> As rel.py shown
```