Nanostream Documentation

Release 0.1

Zachary Ernst

CONTENTS:

1	Overviev	Overview					
	1.1 Wh	hy?	1				
	1.2 Na	nnoStream pipelines	1				
	1.3 Us	sing built-in NanoNode classes	3				
	1.4 Ro	olling your own NanoNode class	5				
	1.5 Co	omposing and configuring NanoNode objects	6				
2	Requirer	ments	9				
3 Implementation							
	3.1 The	e data journey	11				
4	API Doc	umentation 1	13				
	4.1 No	ode module	13				
	4.2 Da	ata structures module	19				
	4.3 Ne	etwork nodes module	30				
	4.4 Na	nnoStreamMessage module	31				
	4.5 Poi	isonPill module	31				
	4.6 Tri	igger module	31				
	4.7 Ba	tch module	31				
	4.8 Na	unoStreamQueue module	31				
5	License	License					
6 Indices and tables							
Ру	ython Mod	lule Index	37				
In	ıdex	3	39				

CHAPTER

ONE

OVERVIEW

1.1 Why?

Tolstoy said that every happy family is the same, but every unhappy family is unhappy in its own way. ETL pipelines are unhappy families.

Why are they so unhappy? Every engineer who does more than one project involving ETL eventually goes through the same stages of ETL grief. First, they thing it's not so bad. Then they do another project and discover that they have to rewrite very similar code. Then they think, "Surely, I could have just written a few library functions and reused that code, saving lots of time." But when they try to do this, they discover that although their ETL projects are very similar, they are just different enough that their code isn't reusable. So they resign themselves to rewriting code over and over again. The code is unreliable, difficult to maintain, and usually poorly tested and documented because it's such a pain to write in the first place. The task of writing ETL pipelines is so lousy that engineering best practices tend to go out the window because the engineer has beter things to do.

NanoStream is an ETL framework written in Python that tries to thread the ETL needle. It is opinionated about how ETL pipelines ought to be written; it provides a number of highly reusable and well-documented modules that are useful in ETL tasks; and it has just enough flexibility to allow an engineer to accommodate the strange idiosyncratic requirements of real-world ETL tasks.

The overall idea of NanoStream is simple. It is a streaming framework that resembles stream-processing patterns such as those found in Spark, Storm, and others. But unlike those tools, it requires no special infrastructure or server because it runs entirely within one process. It also differs from high-powered stream processing tools in that it is designed from the ground up with ETL in mind.

Despite its superficial similarity to Spark and Storm, it is not intended to compete with them in any way. NanoStream is not suitable for huge analytic pipelines that require massive amounts of computation. You won't want to analyze the Twitter firehose with NanoStream.

1.2 NanoStream pipelines

An ETL pipeline in NanoStream is a series of nodes connected by queues. Data is generated or processed in each node, and the output is placed on a queue to be picked up by downstream nodes.

For the sake of convenience, we distinguish between three types of nodes (although there's no real difference in their use or implementation):

- 1. Source nodes. These are nodes that generate data and send it to the rest of the pipeline. They might, for example, read data from an external data source such as an API endpoint or a database.
- 2. Worker nodes. The workers process data by picking up messages from their incoming queues. Their output is placed onto any number of outgoing queues to be further processed by downstream nodes.

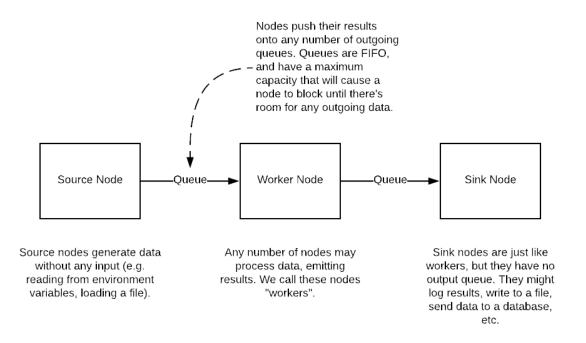


Fig. 1: Very high-level view of a NanoStream pipeline

3. Sink nodes. These are worker nodes with no outgoing queue. They will typically perform tasks such as inserting data into a database or generating statistics to be sent somewhere outside the pipeline.

All pipelines are implemented in pure Python (version >=3.5). Each node is instantiated from a class that inherits from the NanoNode class. Queues are never instantiated directly by the user; they are created automatically whenever two nodes are linked together.

There is a large (and growing) number of specialized NanoNode subclasses, each geared toward a specific task. Such tasks include:

- 1. Querying a table in a SQL database and sending the results downstream.
- 2. Making a request to a REST API, paging through the responses until there are no more results.
- 3. Ingesting individual messages from an upstream node and batching them together into a single message, or doing the reverse.
- 4. Reading environment variables.
- 5. Watching a directory for new files and sending the names of those files down the pipeline when they appear.
- 6. Filtering messages, letting them through the pipeline only if a particular test is passed.

All results and messages passed among the nodes must be dictionary-like objects. By default, messages hold the entire history of all the earlier messages that led to its being generated.

The goal is for NanoStream to be fully "batteries included", with built-in NanoNode subclasses for every necessary ETL task. But because this is actually impossible, we try to make it easy to roll your own node classes.

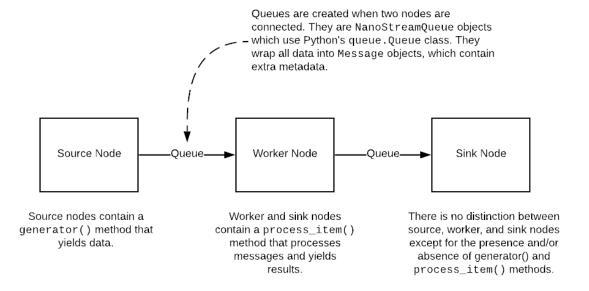


Fig. 2: Somewhat high-level view of a NanoStream pipeline

1.3 Using built-in NanoNode classes

The most straightforward use of NanoStream actually requires no coding, per se. You write a simple configuration file in YAML, and run the pipeline with the built-in command line tool, nanostream_cli. The tool reads the configuration file, instantiates the pipeline, and runs it.

The configuration file has two parts. The first specifies the nodes; the second specifies how they are linked together. Here is an example of a very simple configuration file:

(continues on next page)

(continued from previous page)

Let's look at this config file one part at a time.

Other than the optional pipeline_name and pipeline_description fields, there are two top-level keys: nodes and paths. Each entry in the node section corresponds to a specific node in the pipeline. Their top-level key is whatever name you would like to use to refer to that node – it should be something short, descriptive, and Python-y. We recommend naming these as simple verb-noun phrases such as get_environment_variable, print_variables, light_candle, curse_darkness, etc.

There are three keys under each node configuration: class, summary, and options. The class key is the only one that's required. It is the name of the node's class. The summary key is just an optional arbitrary string.

All nodes share a certain number of common options that are important for any node. Depending on the node, there may also be one or more options specific to it. For example, a node that reads from a SQL database will likely have the table name as an option. In the above configuration, the <code>GetEnvironmentVariables</code> node has (reasonably enough) an option <code>environment_variables</code>, which contains a list of the environment variables that are to be retrieved. The <code>PrinterOfThings</code> node has an optional <code>prepend</code> value, which is a string that will be prepended to anything the node prints to the terminal.

The structure of the pipeline is given in the paths section, which contains a list of lists. Each list is a set of nodes that are to be linked together in order. In our example, the paths value says that get_environment_variables will send its output to print_variables. Paths can be arbitrarily long.

If you wanted to send the environment variables down two different execution paths, you add another list to the paths, like so:

With this set of paths, the pipeline looks like a very simple tree, with get_environment_variables at the root, which branches to print_variables and do_something_else.

When you have written the configuration file, you're ready to use the NanoStream CLI. It accepts a command, followed by some options. As of now, the commands it accepts are run, which executes the pipeline, and draw, which generates a diagram of the pipeline. The relevant command(s) are:

```
python nanostream_cli.py [run | draw] --filename my_sample_config.yaml
```

It is also possible to skip using the configuration file and define your pipelines directly in code. In general, it's better to use the configuration file for a variety of reasons, but you always have the option of doing this in Python.

Nodes are defined in code by instantiating classes that inherit from NanoNode. Upon instantiation, the constructor takes the same set of keyword arguments as you see in the configuration. Nodes are linked together by the > operator, as in node_1 > node_2. After the pipeline has been built in this way, it is started by calling node. global_start() on any of the nodes in the pipeline.

The code corresponding to the configuration file above would look like this:

```
# Define the nodes using the various subclasses of NanoNode
get_environment_variables =
GetEnvironmentVariables(
    environment_variables=['API_KEY', 'API_USER_ID'])
print_variables = PrinterOfThings(prepend='Environment variables: ')

# The '>' operator can also be chained, as in:
# node_1 > node_2 > node_3 > ...
get_environment_variables > print_variables

# Run the pipeline. This command will not block.
get_environment_variables.global_start()
```

1.4 Rolling your own NanoNode class

If there are no built-in NanoNode classes suitable for your ETL pipeline, it is easy to write your own.

For example, suppose you want to create a source node for your pipeline that simply emits a user-defined string every few seconds forever. The user would be able to specify the string and the number of seconds to pause after each message has been sent. The class could be defined like so:

Let's look at each part of this class.

The first thing to note is that the class inherits from NanoNode – this is the mix-in class that gives the node all of its functionality within the NanoStream framework.

The __init__ method should take only keyword arguments, not positional arguments. This restriction is to guarantee that the configuration files have names for any options that are specified in the pipeline. In the __init__ function, you should also be sure to accept **kwargs, because options that are common to all NanoNode objects are expected to be there.

After any attributes have been defined, the __init__ method **must** invoke the parent class's constructor through the use of the super function. Be sure to pass the **kwargs argument into the function as shown in the example.

If the node class is intended to be used as a source node, then you need to define a generator method. This method can be virtually anything, so long as it sends its output via a yield statement.

If you need to define a worker node (that is, a node that accepts input from a queue), you will provide a process_item method instead of a generator. But the structure of that method is the same, with the single exception that you will have access to a __message__ attribute which contains the incoming message data. The structure of a typical process_item method is shown in the figure.

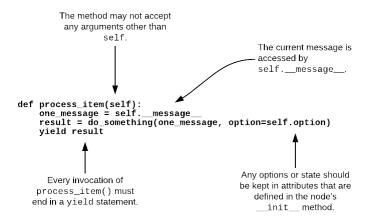


Fig. 3: A typical process_item method for NanoNode objects

For example, let's suppose you want to create a node that is passed a string as a message, and returns True if the message has an even number of characters, False otherwise. The class definition would look like this:

```
class MessageLengthTester(NanoNode):
    def __init__(self):
        # No particular initialization required in this example
        super(MessageLengthTester, self).__init__()

def process_item(self):
    if len(self.__message__) % 2 == 0:
        yield True
    else:
        yield False
```

1.5 Composing and configuring NanoNode objects

(in-progress... code is not guaranteed to be stable...)

Let's suppose you've worked very hard to create the pipeline from the last example. Now, your boss says that another engineering team wants to use it, but they want to rename parameters and "freeze" the values of certain other parameters to specific values. Once that's done, they want to use it as just one part of a more complicated NanoStream pipeline.

This can be accomplished using a configuration file. When NanoStream parses the configuration file, it will dynamically create the desired class, which can be instantiated and used as if it were a single node in another pipeline.

The configuration file is written in YAML, and it would look like this:

```
name: FooMessageTester

nodes:
    - name: foo_generator
    class FooEmitter
    frozen_arguments:
```

(continues on next page)

(continued from previous page)

```
message: foobar
arg_mapping:
   interval: foo_interval
- name: length_tester
   class: MessageLengthTester
   arg_mapping: null
```

With this file saved as (e.g.) foo_message.yaml, the following code will create a FooMessageTester class and instantiate it:

```
foo_message_config = yaml.load(open('./foo_message.yaml', 'r').read())
class_factory(foo_message_config)
# At this point, there is now a `FooMessageTester` class
foo_node = FooMessageTester(foo_interval=1)
```

You can now use foo_node just as you would any other node. So in order to run it, you just do:

```
foo_node.global_start()
```

Because foo_node is just another node, you can insert it into a larger pipeline and reuse it. For example, suppose that other engineering team wants to add a PrinterOfThings to the end of the pipeline. They'd do this:

```
printer = PrinterOfThings()
foo_node > printer
```

CHAPTER

TWO

REQUIREMENTS

NanoStream is written in Python 3.5. All requirements are pip-installable and are listed n the requirements.txt file:

```
` pip install -r requirements.txt `
```

The documentation is written using Sphinx, so if you want to rebuild the docs, you'll do:

```
` make [html | latexpdf | whatever] `
```

That ought to be everything.

CHAPTER

THREE

IMPLEMENTATION

This section describes what's happening under the hood in a NanoStream data pipeline. Most people won't need to read this section.

3.1 The data journey

NanoStream pipelines are sets of NanoNode objects connected by NanoStreamQueue objects. Think of each NanoNode as a vertex in a directed graph, and each NanoStreamQueue as a directed edge.

There are two types of NanoNode objects. A "source" is a NanoNode that does not accept incoming data from another NanoNode. A "processor" is any NanoNode that is not a "source". Note that there is nothing in the class definition or object that distinguishes between these two – the only difference is that processors have a process_item method, and sources have a generator method. Other than that, they are identical.

The data journey begins with one or more source nodes. When a source node is started (by calling its start method), a new thread is created and the node's generator method is executed inside the thread. As results from the generator method are yielded, they are placed on each outgoing NanoStreamQueue to be picked up by one or more processors downstream.

The data from the source's generator is handled by the NanoStreamQueue object. At its heart, the NanoStreamQueue is simply a class which has a Python Queue object as an attribute. The reason we don't simply use Python Queue objects is because the NanoStreamQueue contains some logic that's useful. In particular:

- 1. It wraps the data into a NanoStreamMessage object, which also holds useful metadata including a UUID, the ID of the node that generated the data, and a timestamp.
- 2. If the NanoStreamQueue receives data that is simply a None object, then it is skipped.

API DOCUMENTATION

4.1 Node module

The node module contains the NanoNode class, which is the foundation for NanoStream.

```
class nanostream.node.AggregateValues(values=False, tail_path=None, **kwargs)
    Bases: nanostream.node.NanoNode
    Does that.
    process_item()
         Default no-op for nodes.
class nanostream.node.BatchMessages (batch_size=None, batch_list=None, counter=0, time-
                                          out=5, **kwargs)
    Bases: nanostream.node.NanoNode
    cleanup()
    process_item()
        Default no-op for nodes.
class nanostream.node.CSVReader(*args, **kwargs)
    Bases: nanostream.node.NanoNode
    process_item()
         Default no-op for nodes.
class nanostream.node.CSVToDictionaryList(**kwargs)
    Bases: nanostream.node.NanoNode
    process item()
         Default no-op for nodes.
class nanostream.node.ConstantEmitter(thing=None, delay=2, **kwargs)
    Bases: nanostream.node.NanoNode
    Send a thing every n seconds
    generator()
```

```
class nanostream.node.CounterOfThings(*args, batch=False, get_runtime_attrs=<function
                                                                  get_runtime_attrs_args=None,
                                               no\_op>,
                                               get runtime attrs kwargs=None,
                                              time_attrs_destinations=None,
                                                                                         in-
                                              put mapping=None,
                                                                   retain input=True,
                                                                                       throt-
                                               tle=0,
                                                          keep alive=True,
                                                                               max errors=0,
                                              name=None.
                                                                 input message keypath=None,
                                              key=None, messages_received_counter=0,
                                              sages_sent_counter=0, post_process_function=None,
                                              post_process_keypath=None,
                                                                                summary=",
                                              post_process_function_kwargs=None,
                                                                                        out-
                                              put_key=None, **kwargs)
     Bases: nanostream.node.NanoNode
     foo init (*args, start=0, end=None, **kwargs)
     generator()
         Just start counting integers
class nanostream.node.DynamicClassMediator(*args, **kwargs)
     Bases: nanostream.node.NanoNode
     get_sink()
     get_source()
     hi()
     sink_list()
     source_list()
class nanostream.node.Filter(test=None, key=None, value=True, *args, **kwargs)
     Bases: nanostream.node.NanoNode
     Applies tests to each message and filters out messages that don't pass
     Built-in tests: key exists value is true value is not none
     Example:
         {'test': 'key_exists', 'key': mykey}
     process_item()
         Default no-op for nodes.
class nanostream.node.GetEnvironmentVariables (mappings=None,
                                                                                    environ-
                                                        ment_variables=None, **kwargs)
     Bases: nanostream.node.NanoNode
     generator()
     process item()
         Default no-op for nodes.
class nanostream.node.InsertData (overwrite=True, overwrite_if_null=True, value_dict=None,
                                         **kwargs)
     Bases: nanostream.node.NanoNode
     process_item()
         Default no-op for nodes.
class nanostream.node.LocalDirectoryWatchdog(directory='.', check_interval=3, **kwargs)
     Bases: nanostream.node.NanoNode
```

generator()

class nanostream.node.LocalFileReader(*args, **kwargs)

Bases: nanostream.node.NanoNode

process item()

Default no-op for nodes.

class nanostream.node.NanoNode(*args, batch=False. get_runtime_attrs=<function</pre> no op>, get runtime attrs args=None, get_runtime_attrs_kwargs=None, time_attrs_destinations=None, input_mapping=None, tain_input=True, throttle=0, keep_alive=True, max_errors=0, name=None, input_message_keypath=None, kev=None, messages_received_counter=0, $messages_sent_counter=0$, post_process_function=None, post_process_keypath=None, summary=", post_process_function_kwargs=None, put_key=None, **kwargs)

Bases: object

The foundational class of NanoStream. This class is inherited by all nodes in a computation graph.

Order of operations: 1. Child class __init__ function 2. NanoNode __init__ function 3. preflight_function (Specified in initialization params) 4. setup 5. start

These methods have the following intended uses:

- 1. __init__ Sets attribute values and calls the NanoNode __init__ method.
- 2. get_runtime_attrs Sets any attribute values that are to be determined at runtime, e.g. by checking environment variables or reading values from a database. The get_runtime_attrs should return a dictionary of attributes -> values, or else None.
- 3. setup Sets the state of the NanoNode and/or creates any attributes that require information available only at runtime.

Variables

- **send_batch_markers** If True, then a BatchStart marker will be sent when a new input is received, and a BatchEnd will be sent after the input has been processed. The intention is that a number of items will be emitted for each input received. For example, we might emit a table row-by-row for each input.
- **get_runtime_attrs** A function that returns a dictionary-like object. The keys and values will be saved to this NanoNode object's attributes. The function is executed one time, upon starting the node.
- **get_runtime_attrs_args** A tuple of arguments to be passed to the get_runtime_attrs function upon starting the node.
- **get_runtime_attrs_kwargs** A dictionary of kwargs passed to the get_runtime_attrs function.
- runtime_attrs_destinations If set, this is a dictionary mapping the keys returned from the get_runtime_attrs function to the names of the attributes to which the values will be saved.
- throttle For each input received, a delay of throttle seconds will be added.
- **keep_alive** If True, keep the node's thread alive after everything has been processed.
- name The name of the node. Defaults to a randomly generated hash. Note that this hash is not consistent from one run to the next.

4.1. Node module 15

- **input_mapping** When the node receives a dictionary-like object, this dictionary will cause the keys of the dictionary to be remapped to new keys.
- **retain_input** If True, then combine the dictionary-like input with the output. If keys clash, the output value will be kept.
- **input_message_keypath** Read the value in this keypath as the content of the incoming message.

add_edge (target, **kwargs)

Create an edge connecting self to target. The edge is really just a queue

all_connected(seen=None)

Returns all the nodes connected (directly or indirectly) to self.

Args:

seen (set): A set of all the nodes that have been identified as connected to self.

Returns:

(set of NanoNode): All the nodes connected to self. This includes self.

broadcast (broadcast_message)

Puts the message into all the input queues for all connected nodes.

cleanup()

draw_pipeline()

Draw the pipeline structure using graphviz.

global_start (datadog=False, pipeline_name=None)

Starts every node connected to self.

input_queue_size

Return the total number of items in all of the queues that are inputs to this node.

is sink

Tests whether the node is a sink or not, i.e. whether there are no outputs from the node.

is source

Tests whether the node is a source or not, i.e. whether there are no inputs to the node.

kill_pipeline()

```
log_info(message=")
```

process_item(*args, **kwargs)

Default no-op for nodes.

processor()

This calls the user's process_item with just the message content, and then returns the full message.

setup()

To be overridden by child classes when we need to do something after setting attributes and the pre-flight function.

start()

Starts the node. This is called by NanoNode.global_start().

stream()

Called in each NanoNode thread.

terminate_pipeline()

This method can be called on any node in a pipeline, and it will cause all of the nodes to terminate.

```
This function loops over all of the threads in the pipeline, checking that they are either finished or
         running. If any have had an abnormal exit, terminate the entire pipeline.
     time running
         Return the number of wall-clock seconds elapsed since the node was started.
class nanostream.node.NothingToSeeHere
     Bases: object
     Vacuous class used as a no-op message type.
class nanostream.node.Parameters(**kwargs)
     Bases: object
class nanostream.node.PrinterOfThings(*args, **kwargs)
     Bases: nanostream.node.NanoNode
     process_item()
         Default no-op for nodes.
class nanostream.node.RandomSample(sample=0.1)
     Bases: nanostream.node.NanoNode
     Lets through only a random sample of incoming messages. Might be useful for testing, or when only approxi-
     mate results are necessary.
     process item()
         Default no-op for nodes.
class nanostream.node.Remapper (mapping=None, **kwargs)
     Bases: nanostream.node.NanoNode
     process_item()
         Default no-op for nodes.
class nanostream.node.SequenceEmitter(sequence, *args, max_sequences=1, **kwargs)
     Bases: nanostream.node.NanoNode
     Emits sequence max_sequences times, or forever if max_sequences is None.
     generator()
         Emit the sequence max_sequences times.
     process_item()
         Emit the sequence max_sequences times.
class nanostream.node.Serializer(values=False, *args, **kwargs)
     Bases: nanostream.node.NanoNode
     Takes an iterable thing as input, and successively yields its items.
     process_item()
         Default no-op for nodes.
class nanostream.node.SimpleTransforms (missing_keypath_action='ignore',
                                                ing_path=None,
                                                                     transform_mapping=None,
                                                target_value=None, keypath=None, **kwargs)
     Bases: nanostream.node.NanoNode
     process_item()
         Default no-op for nodes.
```

thread monitor()

4.1. Node module

```
class nanostream.node.StreamMySQLTable(*args, host='localhost', user=None, table=None,
                                                  password=None,
                                                                   database=None,
                                                                                     port = 3306,
                                                                       send batch markers=True,
                                                  to row obj=False,
                                                  **kwargs)
     Bases: nanostream.node.NanoNode
     generator()
     get_schema()
     setup()
          To be overridden by child classes when we need to do something after setting attributes and the pre-flight
class nanostream.node.StreamingJoin(window=30, streams=None, *args, **kwargs)
     Bases: nanostream.node.NanoNode
     Joins two streams on a key, using exact match only. MVP.
     process_item()
class nanostream.node.SubstituteRegex (match_regex=None, substitute_string=None, *args,
                                                 **kwargs)
     Bases: nanostream.node.NanoNode
     process_item()
          Default no-op for nodes.
class nanostream.node.TimeWindowAccumulator(*args, **kwargs)
     Bases: nanostream.node.NanoNode
     Every N seconds, put the latest M seconds data on the queue.
class nanostream.node.bcolors
     Bases: object
     This class holds the values for the various colors that are used in the tables that monitor the status of the nodes.
     BOLD = ' \times 1b[1m']
     ENDC = ' \times 1b[0m']
     FAIL = ' x1b[91m']
     HEADER = ' \times 1b [95m']
     OKBLUE = ' \times 1b [94m']
     OKGREEN = ' \times 1b[92m']
     UNDERLINE = ' \times 1b[4m']
     WARNING = ' \times 1b[93m']
nanostream.node.class_factory(raw_config)
nanostream.node.get environment variables(*args)
     Retrieves the environment variables listed in *args.
     Args: args (list of str): List of environment variables.
     Returns:
          dict: Dictionary of environment variables to values. If the environment variable is not defined, the
              value is None.
nanostream.node.get_node_dict(node_config)
```

```
nanostream.node.kwarg_remapper(f, **kwarg_mapping)
nanostream.node.no_op(*args, **kwargs)
    No-op function to serve as default get runtime attrs.
nanostream.node.template_class(class_name,
                                                    parent_class,
                                                                      kwargs remapping,
                                   frozen arguments mapping)
4.2 Data structures module
Data types (e.g. Rows, Records) for ETL.
class nanostream.utils.data_structures.BOOL(value, original_type=None, name=None)
                 nanostream.utils.data_structures.DataType,
                                                                   nanostream.utils.
    data_structures.IntermediateTypeSystem
    python_cast_function
         alias of builtins.bool
class nanostream.utils.data structures.DATETIME (value,
                                                                     original type=None,
                 nanostream.utils.data_structures.DataType,
                                                                      nanostream.utils.
    data structures. Intermediate Type System
    python_cast_function()
class nanostream.utils.data_structures.DataSourceTypeSystem
    Bases: object
    Information about mapping one type system onto another contained in the children of this class.
    static convert (obj)
         Override this method if something more complicated is necessary.
    static type_mapping(*args, **kwargs)
class nanostream.utils.data_structures.DataType(value,
                                                                     original_type=None,
                                                       name=None)
    Bases: object
    Each DataType gets a python cast function, which is a function.
    intermediate type = None
    python_cast_function = None
    to_intermediate_type()
         Convert the DataType to an IntermediateDataType using its class's intermediate_type
         attribute.
    to_python()
    type_system
        Just for convenience to make the type system an attribute.
class nanostream.utils.data_structures.FLOAT(value, original_type=None, name=None)
                 nanostream.utils.data structures.DataType, nanostream.utils.
    data_structures.IntermediateTypeSystem
```

python_cast_function

alias of builtins.float

```
class nanostream.utils.data structures.INTEGER (value,
                                                                 original type=None,
                                                   name=None
                nanostream.utils.data_structures.DataType,
                                                                  nanostream.utils.
    data structures. Intermediate Type System
    python_cast_function
        alias of builtins.int
exception nanostream.utils.data_structures.IncompatibleTypesException
    Bases: Exception
class nanostream.utils.data_structures.IntermediateTypeSystem
    Bases: nanostream.utils.data structures.DataSourceTypeSystem
    Never instantiate this by hand.
class nanostream.utils.data_structures.MYSQL_BOOL(value,
                                                                 original_type=None,
                                                      name=None
                nanostream.utils.data_structures.DataType,
                                                                  nanostream.utils.
    data_structures.MySQLTypeSystem
    intermediate type
        alias of BOOL
    python_cast_function
        alias of builtins.bool
class nanostream.utils.data_structures.MYSQL_DATE(value,
                                                                 original_type=None,
                                                      name=None)
                nanostream.utils.data_structures.DataType,
                                                                  nanostream.utils.
    data_structures.MySQLTypeSystem
    intermediate type
        alias of DATETIME
    python cast function()
class nanostream.utils.data_structures.MYSQL_ENUM(value,
                                                                 original_type=None,
                                                      name=None)
    Bases:
                nanostream.utils.data_structures.DataType,
                                                                  nanostream.utils.
    data_structures.MySQLTypeSystem
    intermediate_type
        alias of STRING
    python cast function
        alias of builtins.str
class nanostream.utils.data_structures.MYSQL_INTEGER
    Bases: type
class nanostream.utils.data_structures.MYSQL_INTEGER0(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max length = 0
class nanostream.utils.data_structures.MYSQL_INTEGER1(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max_length = 1
```

```
class nanostream.utils.data structures.MYSOL INTEGER10 (value, original type=None,
                                                           name=None
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max length = 10
class nanostream.utils.data_structures.MYSQL_INTEGER1024 (value,
                                                                            origi-
                                                             nal_type=None,
                                                             name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max_length = 1024
class nanostream.utils.data_structures.MYSQL_INTEGER11 (value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max_length = 11
class nanostream.utils.data structures.MYSQL INTEGER12 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max length = 12
class nanostream.utils.data structures.MYSQL INTEGER128 (value,
                                                                            origi-
                                                            nal type=None,
                                                            name=None)
    Bases: nanostream.utils.data structures.MYSOL INTEGER BASE
    max length = 128
class nanostream.utils.data_structures.MYSQL_INTEGER13(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max_length = 13
class nanostream.utils.data_structures.MYSQL_INTEGER14(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max_length = 14
class nanostream.utils.data_structures.MYSQL_INTEGER15(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max length = 15
class nanostream.utils.data_structures.MYSQL_INTEGER16(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max_length = 16
class nanostream.utils.data_structures.MYSQL_INTEGER16384 (value,
                                                                            origi-
                                                               nal type=None,
                                                               name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max_length = 16384
class nanostream.utils.data structures.MYSQL INTEGER17 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
```

```
max length = 17
class nanostream.utils.data structures.MYSQL INTEGER18 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSOL INTEGER BASE
    max_length = 18
class nanostream.utils.data_structures.MYSQL_INTEGER19(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max_length = 19
class nanostream.utils.data_structures.MYSQL_INTEGER2(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max length = 2
class nanostream.utils.data structures.MYSQL INTEGER20 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max length = 20
class nanostream.utils.data_structures.MYSQL INTEGER2048(value.
                                                                            origi-
                                                             nal\_type=None,
                                                             name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max length = 2048
class nanostream.utils.data structures.MYSQL INTEGER21 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max_length = 21
class nanostream.utils.data_structures.MYSQL_INTEGER22(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max length = 22
class nanostream.utils.data_structures.MYSQL_INTEGER23(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max_length = 23
class nanostream.utils.data_structures.MYSQL_INTEGER24(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max_length = 24
class nanostream.utils.data_structures.MYSQL_INTEGER25(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max length = 25
class nanostream.utils.data_structures.MYSQL_INTEGER256(value,
                                                                            origi-
                                                            nal_type=None,
                                                            name=None)
    Bases: nanostream.utils.data structures.MYSOL INTEGER BASE
```

```
max length = 256
class nanostream.utils.data structures.MYSQL INTEGER26 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSOL INTEGER BASE
    max_length = 26
class nanostream.utils.data_structures.MYSQL_INTEGER27 (value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max_length = 27
class nanostream.utils.data_structures.MYSQL_INTEGER28(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max length = 28
class nanostream.utils.data structures.MYSQL INTEGER29 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max length = 29
class nanostream.utils.data structures.MYSQL INTEGER3 (value, original type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSOL INTEGER BASE
    max_length = 3
class nanostream.utils.data_structures.MYSQL_INTEGER30(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max_length = 30
class nanostream.utils.data_structures.MYSQL_INTEGER31(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max_length = 31
class nanostream.utils.data_structures.MYSQL_INTEGER32(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max length = 32
class nanostream.utils.data structures.MYSQL INTEGER32768 (value,
                                                                            origi-
                                                              nal_type=None,
                                                              name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max length = 32768
class nanostream.utils.data_structures.MYSQL_INTEGER4(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max length = 4
class nanostream.utils.data_structures.MYSQL_INTEGER4096(value,
                                                                            origi-
                                                             nal\ type=None,
                                                             name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
```

```
max length = 4096
class nanostream.utils.data structures.MYSQL INTEGER5 (value, original type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSOL INTEGER BASE
    max_length = 5
class nanostream.utils.data_structures.MYSQL_INTEGER512 (value,
                                                                            origi-
                                                            nal\_type=None,
                                                            name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max_length = 512
class nanostream.utils.data structures.MYSOL INTEGER6 (value, original type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max length = 6
class nanostream.utils.data_structures.MYSQL_INTEGER64(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max length = 64
class nanostream.utils.data_structures.MYSQL_INTEGER7(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max length = 7
class nanostream.utils.data_structures.MYSQL_INTEGER8(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max_length = 8
class nanostream.utils.data_structures.MYSQL_INTEGER8192 (value,
                                                                            origi-
                                                             nal\_type=None,
                                                             name=None)
    Bases: nanostream.utils.data_structures.MYSQL_INTEGER_BASE
    max_length = 8192
class nanostream.utils.data_structures.MYSQL_INTEGER9(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSQL INTEGER BASE
    max_length = 9
class nanostream.utils.data_structures.MYSQL_INTEGER_BASE(value,
                                                                            origi-
                                                              nal_type=None,
                                                              name=None)
                nanostream.utils.data structures.DataType,
                                                                  nanostream.utils.
    data_structures.MySQLTypeSystem
    intermediate_type
        alias of INTEGER
    python_cast_function
        alias of builtins.int
class nanostream.utils.data_structures.MYSQL_VARCHAR
    Bases: type
```

```
class nanostream.utils.data structures.MYSOL VARCHARO (value, original type=None,
                                                          name=None
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    \max length = 0
class nanostream.utils.data_structures.MYSQL_VARCHAR1(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSOL VARCHAR BASE
    max length = 1
class nanostream.utils.data_structures.MYSQL_VARCHAR10(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max length = 10
class nanostream.utils.data structures.MYSQL VARCHAR1024 (value,
                                                                            origi-
                                                             nal type=None,
                                                             name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max_length = 1024
class nanostream.utils.data structures.MYSQL VARCHAR11 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSOL VARCHAR BASE
    max_length = 11
class nanostream.utils.data structures.MYSQL VARCHAR12 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSOL VARCHAR BASE
    max_length = 12
class nanostream.utils.data_structures.MYSQL_VARCHAR128 (value,
                                                                            origi-
                                                            nal_type=None,
                                                            name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max_length = 128
class nanostream.utils.data_structures.MYSQL_VARCHAR13(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max length = 13
class nanostream.utils.data_structures.MYSQL_VARCHAR14(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max_length = 14
class nanostream.utils.data_structures.MYSQL_VARCHAR15(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSOL VARCHAR BASE
    max_length = 15
class nanostream.utils.data structures.MYSQL VARCHAR16 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max_length = 16
```

```
class nanostream.utils.data structures.MYSOL VARCHAR16384 (value,
                                                                            origi-
                                                              nal type=None,
                                                              name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max length = 16384
class nanostream.utils.data_structures.MYSQL_VARCHAR17 (value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max_length = 17
class nanostream.utils.data_structures.MYSQL_VARCHAR18(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max_length = 18
class nanostream.utils.data structures.MYSQL VARCHAR19 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max length = 19
class nanostream.utils.data structures.MYSQL VARCHAR2 (value, original type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSOL VARCHAR BASE
    max_length = 2
class nanostream.utils.data structures.MYSQL VARCHAR20 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSOL VARCHAR BASE
    max_length = 20
class nanostream.utils.data_structures.MYSQL_VARCHAR2048(value,
                                                                            origi-
                                                             nal_type=None,
                                                             name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max_length = 2048
class nanostream.utils.data_structures.MYSQL_VARCHAR21(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max length = 21
class nanostream.utils.data_structures.MYSQL_VARCHAR22(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max_length = 22
class nanostream.utils.data_structures.MYSQL_VARCHAR23(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSOL VARCHAR BASE
    max_length = 23
class nanostream.utils.data structures.MYSQL VARCHAR24 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max_length = 24
```

```
class nanostream.utils.data structures.MYSOL VARCHAR25 (value, original type=None,
                                                           name=None
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max length = 25
class nanostream.utils.data_structures.MYSQL_VARCHAR256 (value,
                                                                            origi-
                                                            nal\_type=None,
                                                            name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max_length = 256
class nanostream.utils.data_structures.MYSQL_VARCHAR26(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max length = 26
class nanostream.utils.data structures.MYSQL VARCHAR27 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max length = 27
class nanostream.utils.data structures.MYSQL VARCHAR28 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSOL VARCHAR BASE
    max_length = 28
class nanostream.utils.data structures.MYSQL VARCHAR29 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSOL VARCHAR BASE
    max_length = 29
class nanostream.utils.data_structures.MYSQL_VARCHAR3(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max_length = 3
class nanostream.utils.data structures.MYSOL VARCHAR30 (value, original type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max length = 30
class nanostream.utils.data_structures.MYSQL_VARCHAR31(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max length = 31
class nanostream.utils.data_structures.MYSQL_VARCHAR32(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max_length = 32
class nanostream.utils.data structures.MYSQL VARCHAR32768 (value,
                                                                            origi-
                                                              nal type=None,
                                                              name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max_length = 32768
```

```
class nanostream.utils.data_structures.MYSQL_VARCHAR4(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max_length = 4
class nanostream.utils.data_structures.MYSQL_VARCHAR4096 (value,
                                                                            origi-
                                                             nal_type=None,
                                                             name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max_length = 4096
class nanostream.utils.data_structures.MYSQL_VARCHAR5(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max_length = 5
class nanostream.utils.data structures.MYSQL VARCHAR512 (value,
                                                                            origi-
                                                            nal type=None,
                                                            name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max length = 512
class nanostream.utils.data_structures.MYSQL_VARCHAR6(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSOL VARCHAR BASE
    max length = 6
class nanostream.utils.data_structures.MYSQL_VARCHAR64(value, original_type=None,
                                                           name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max_length = 64
class nanostream.utils.data_structures.MYSQL_VARCHAR7(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max_length = 7
class nanostream.utils.data_structures.MYSQL_VARCHAR8(value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max length = 8
class nanostream.utils.data_structures.MYSQL_VARCHAR8192 (value,
                                                                            origi-
                                                             nal_type=None,
                                                             name=None)
    Bases: nanostream.utils.data structures.MYSQL VARCHAR BASE
    max length = 8192
class nanostream.utils.data_structures.MYSQL_VARCHAR9 (value, original_type=None,
                                                          name=None)
    Bases: nanostream.utils.data_structures.MYSQL_VARCHAR_BASE
    max_length = 9
class nanostream.utils.data_structures.MYSQL_VARCHAR_BASE(value,
                                                                            origi-
                                                              nal\_type=None,
                                                              name=None)
```

```
Bases:
                 nanostream.utils.data structures.DataType, nanostream.utils.
    data_structures.MySQLTypeSystem
    intermediate_type
         alias of STRING
    python cast function
         alias of builtins.str
class nanostream.utils.data structures.MySQLTypeSystem
    Bases: nanostream.utils.data_structures.DataSourceTypeSystem
    Each TypeSystem gets a type_mapping static method that takes a string and returns the class in the type
    system named by that string. For example, int (8) in a MySQL schema should return the MYSQL_INTEGER8
    class.
    static type_mapping(string)
         Parses the schema strings from MySQL and returns the appropriate class.
class nanostream.utils.data_structures.PrimitiveTypeSystem
    Bases: nanostream.utils.data structures.DataSourceTypeSystem
class nanostream.utils.data_structures.PythonTypeSystem
    Bases: nanostream.utils.data_structures.DataSourceTypeSystem
class nanostream.utils.data_structures.Row(*records, type_system=None)
    Bases: object
    A collection of DataType objects (typed values). They are dictionaries mapping the names of the values to
    the DataType objects.
    concat (other, fail_on_duplicate=True)
    static from_dict(row_dictionary, **kwargs)
         Creates a Row object form a dictionary mapping names to values.
    is_empty()
    keys()
         For implementing the mapping protocol.
class nanostream.utils.data structures.STRING(value, original type=None, name=None)
                 nanostream.utils.data structures.DataType,
                                                                       nanostream.utils.
    data structures. Intermediate Type System
    python_cast_function
         alias of builtins.str
nanostream.utils.data structures.all bases (obj)
    Return all the class to which obj belongs.
\verb|nanostream.utils.data_structures.convert_to_type_system| (obj, cls)
nanostream.utils.data_structures.get_type_system(obj)
nanostream.utils.data_structures.make_types()
nanostream.utils.data_structures.mysql_type(string)
    Parses the schema strings from MySQL and returns the appropriate class.
nanostream.utils.data_structures.primitive_to_intermediate_type(thing,
                                                                          name=None)
```

4.3 Network nodes module

Classes that deal with sending and receiving data across the interwebs.

Bases: nanostream.node.NanoNode

Node class for making simple GET requests.

```
process_item()
```

The input to this function will be a dictionary-like object with parameters to be substituted into the endpoint string and a dictionary with keys and values to be passed in the GET request.

Three use-cases: 1. Endpoint and parameters set initially and never changed. 2. Endpoint and parameters set once at runtime 3. Endpoint and parameters set by upstream messages

Bases: nanostream.node.NanoNode

Node class for HTTP API requests that require paging through sets of results.

```
process_item()
```

```
Default no-op for nodes.

class nanostream.node_classes.network_nodes.PaginatedHttpGetRequest (endpoint_template=None, additional_data_key=None, pagination_key=None, pagination_key=None, pagination_get_request_key=None, default_offset_value=", additional_data_test=<class 'bool'>)
```

Bases: object

For handling requests in a semi-general way that require paging through lists of results and repeatedly making GET requests.

```
responses()
```

Generator. Yields each response until empty.

4.4 NanoStreamMessage module

The NanoStreamMesaage encapsulates the content of each piece of data, along with some useful metadata.

A class that contains the message payloads that are queued for each NanoStreamProcessor. It holds the messages and lots of metadata used for logging, monitoring, etc.

4.5 PoisonPill module

A simple class that is sent in a message to signal that the node should be terminated.

```
class nanostream.message.poison_pill.PoisonPill
    Bases: object
```

4.6 Trigger module

A simple class containing no data, which is intended merely as a trigger, signaling that the downstream node should do something.

4.7 Batch module

We'll use markers to delimit batches of things, such as serialized files and that kind of thing.

```
class nanostream.message.batch.BatchEnd(*args, **kwargs)
    Bases: object

class nanostream.message.batch.BatchStart(*args, **kwargs)
    Bases: object

class nanostream.message.canary.Canary
    Bases: object
```

4.8 NanoStreamQueue module

These are queues that form the directed edges between nodes.

empty

get()

put (message, *args, previous_message=None, **kwargs)

Places a message on the output queues. If the message is None, then the queue is skipped.

Messages are NanoStreamMessage objects; the payload of the message is message.message_content.

CHAPTER

FIVE

LICENSE

Copyright (C) 2016 Zachary Ernst zac.ernst@gmail.com

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see http://www.gnu.org/licenses/>.

34 Chapter 5. License

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

```
nanostream.message.batch, 31
nanostream.message.canary, 31
nanostream.message.message, 31
nanostream.message.poison_pill, 31
nanostream.message.trigger, 31
nanostream.node, 13
nanostream.node_classes.network_nodes, 29
nanostream.node_queue.queue, 31
nanostream.utils.data_structures, 19
```

38 Python Module Index

INDEX

A	E
add_edge() (nanostream.node.NanoNode method), 16 AggregateValues (class in nanostream.node), 13	empty (nanostream.node_queue.queue.NanoStreamQueue attribute), 31
all_bases() (in module nanostream.utils.data_structures),	ENDC (nanostream.node.bcolors attribute), 18
all_connected() (nanostream.node.NanoNode method),	F
16	FAIL (nanostream.node.bcolors attribute), 18
В	Filter (class in nanostream.node), 14
BatchEnd (class in nanostream.message.batch), 31	FLOAT (class in nanostream.utils.data_structures), 19 fooinit() (nanostream.node.CounterOfThings
BatchMessages (class in nanostream.node), 13	method), 14
BatchStart (class in nanostream.message.batch), 31	from_dict() (nanostream.utils.data_structures.Row static
bcolors (class in nanostream.node), 18	method), 29
BOLD (nanostream.node.bcolors attribute), 18	
BOOL (class in nanostream.utils.data_structures), 19	G
broadcast() (nanostream.node.NanoNode method), 16	generator() (nanostream.node.ConstantEmitter method), 13
	generator() (nanostream.node.CounterOfThings method),
Canary (class in nanostream.message.canary), 31	14 C. F. in the William
class_factory() (in module nanostream.node), 18 cleanup() (nanostream.node.BatchMessages method), 13	generator() (nanostream.node.GetEnvironmentVariables method), 14
cleanup() (nanostream.node.NanoNode method), 16	generator() (nanostream.node.LocalDirectoryWatchdog
concat() (nanostream.utils.data_structures.Row method),	method), 14
29	generator() (nanostream.node.SequenceEmitter method),
ConstantEmitter (class in nanostream.node), 13	17
convert() (nanostream.utils.data_structures.DataSourceTypstatic method), 19	(nanostream.node.StreamMySQLTable method), 18
convert_to_type_system() (in module nanos-	$get() \hspace{0.2cm} (nanostream.node_queue.queue.NanoStreamQueue$
tream.utils.data_structures), 29	method), 32
CounterOfThings (class in nanostream.node), 13 CSVReader (class in nanostream.node), 13	get_environment_variables() (in module nanostream.node), 18
CSVToDictionaryList (class in nanostream.node), 13	get_node_dict() (in module nanostream.node), 18
D	get_schema() (nanostream.node.StreamMySQLTable method), 18
DataSourceTypeSystem (class in nanos-	get_sink() (nanostream.node.DynamicClassMediator
tream.utils.data_structures), 19	method), 14
DataType (class in nanostream.utils.data_structures), 19	get_source() (nanostream.node.DynamicClassMediator
DATETIME (class in nanostream.utils.data_structures),	method), 14
19	get_type_system() (in module nanos-
draw_pipeline() (nanostream.node.NanoNode method),	tream.utils.data_structures), 29
16 Dimensio Class Mediaton (class in nonestroom mede) 14	GetEnvironmentVariables (class in nanostream.node), 14 global_start() (nanostream.node.NanoNode method), 16
DynamicClassMediator (class in nanostream.node), 14	giodai_start() (nanostream.noue.ivanoivoue memou), 10

H	M			
HEADER (nanostream.node.bcolors attribute), 18	make_types()	(in	module	nanos-
hello_world() (in module nanostream.message.trigger),	tream.ı	utils.data_struc		
31	-		ata_structures.N	MYSQL_INTEGER0
hi() (nanostream.node.DynamicClassMediator method),	attribut	* *		WOOL DEECED1
14	-		ata_structures.	MYSQL_INTEGER1
HttpGetRequest (class in nanostream.node_classes.network_nodes), 30	attribut	* *	ata etructurae M	MYSQL_INTEGER10
HttpGetRequestPaginator (class in nanos-	attribut		ata_structures.r	WIDQL_IIVILOLKIO
tream.node_classes.network_nodes), 30		, ,	ata structures.N	MYSQL_INTEGER1024
/	attribut		_	_
I			ata_structures.N	MYSQL_INTEGER11
IncompatibleTypesException, 20	attribut		_	
input_queue_size (nanostream.node.NanoNode at-			ata_structures.N	MYSQL_INTEGER12
tribute), 16	attribut		oto structuros M	MYSQL_INTEGER128
InsertData (class in nanostream.node), 14 INTEGER (class in nanostream.utils.data_structures), 19	attribut		ata_structures.r	WI15QL_INTEGER126
intermediate_type (nanos-			ata structures.N	MYSQL_INTEGER13
tream.utils.data_structures.DataType attribute),	attribut			
19	max_length (nan	ostream.utils.d	ata_structures.N	MYSQL_INTEGER14
intermediate_type (nanos-	attribut	* *		
tream.utils.data_structures.MYSQL_BOOL	_		ata_structures.N	MYSQL_INTEGER15
attribute), 20	attribut	, ,	oto structuros M	MVCOL INTECED16
intermediate_type (nanos-	attribut		iata_structures.r	MYSQL_INTEGER16
tream.utils.data_structures.MYSQL_DATE attribute), 20			ata structures.N	MYSQL_INTEGER16384
intermediate_type (nanos-	attribut			
tream.utils.data_structures.MYSQL_ENUM	max_length (nan-	ostream.utils.d	ata_structures.N	MYSQL_INTEGER17
attribute), 20	attribut			
intermediate_type (nanos-			ata_structures.N	MYSQL_INTEGER18
tream.utils.data_structures.MYSQL_INTEGER_	BASE attribut		oto structuros M	MYSQL_INTEGER19
attribute), 24 intermediate type (nanos-	attribut		iata_structures.r	WI3QL_INTEGERIA
intermediate_type (nanos- tream.utils.data_structures.MYSQL_VARCHAR			ata structures.N	MYSOL INTEGER2
attribute), 29	attribut	te), 22		
IntermediateTypeSystem (class in nanos-	max_length (nan	ostream.utils.d	ata_structures.N	MYSQL_INTEGER20
tream.utils.data_structures), 20	attribut		_	
is_empty() (nanostream.utils.data_structures.Row	-		ata_structures.	MYSQL_INTEGER2048
method), 29	attribut	* *	oto structuros M	MYSQL_INTEGER21
is_sink (nanostream.node.NanoNode attribute), 16 is_source (nanostream.node.NanoNode attribute), 16	attribut		ata_structures.r	WITSQL_INTEGER21
is_source (nanostream.node.nanonode attribute), 10		* *	ata structures.N	MYSQL_INTEGER22
K	attribut		_	~ -
keys() (nanostream.utils.data_structures.Row method), 29	max_length (nan	ostream.utils.d	ata_structures.N	MYSQL_INTEGER23
kill_pipeline() (nanostream.node.NanoNode method), 16	attribut		_	
kwarg_remapper() (in module nanostream.node), 18	_		ata_structures.	MYSQL_INTEGER24
ı	attribut	* *	oto structuros M	MYSQL_INTEGER25
L	attribut		iata_structures.r	WITSQL_INTEGER23
LocalDirectoryWatchdog (class in nanostream.node), 14		* *	ata structures.N	MYSQL_INTEGER256
LocalFileReader (class in nanostream.node), 15	attribut			
log_info() (nanostream.node.NanoNode method), 16	_		ata_structures.N	MYSQL_INTEGER26
	attribut			
	max_length (nan	ostream.utils.d	ata_structures.	MYSQL_INTEGER27

- attribute), 23 attribute), 25
- max_length (nanostream.utils.data_structures.MYSQL_INTEGE_R28gth (nanostream.utils.data_structures.MYSQL_VARCHAR16 attribute), 23 attribute), 25
- max_length (nanostream.utils.data_structures.MYSQL_INTEGERAPgth (nanostream.utils.data_structures.MYSQL_VARCHAR16384 attribute), 23 attribute), 26
- max_length (nanostream.utils.data_structures.MYSQL_INTEGERength (nanostream.utils.data_structures.MYSQL_VARCHAR17 attribute), 23 attribute), 26
- max_length (nanostream.utils.data_structures.MYSQL_INTEGERENGth (nanostream.utils.data_structures.MYSQL_VARCHAR18 attribute), 23
- max_length (nanostream.utils.data_structures.MYSQL_INTEGE_Renigth (nanostream.utils.data_structures.MYSQL_VARCHAR19 attribute), 23 attribute), 26
- max_length (nanostream.utils.data_structures.MYSQL_INTEGE_Reagth (nanostream.utils.data_structures.MYSQL_VARCHAR2 attribute), 23 attribute), 26
- max_length (nanostream.utils.data_structures.MYSQL_INTEGE_REAGA6(nanostream.utils.data_structures.MYSQL_VARCHAR20 attribute), 23 attribute), 26
- max_length (nanostream.utils.data_structures.MYSQL_INTEGE_Rength (nanostream.utils.data_structures.MYSQL_VARCHAR2048 attribute), 23 attribute), 26
- max_length (nanostream.utils.data_structures.MYSQL_INT**EGE_ReftQt** (nanostream.utils.data_structures.MYSQL_VARCHAR21 attribute), 23 attribute), 26
- max_length (nanostream.utils.data_structures.MYSQL_INTEGE_Refngth (nanostream.utils.data_structures.MYSQL_VARCHAR22 attribute), 24 attribute), 26
- max_length (nanostream.utils.data_structures.MYSQL_INThotElectron (nanostream.utils.data_structures.MYSQL_VARCHAR23 attribute), 24 attribute), 26
- max_length (nanostream.utils.data_structures.MYSQL_INTEGE_Rength (nanostream.utils.data_structures.MYSQL_VARCHAR24 attribute), 24
- max_length (nanostream.utils.data_structures.MYSQL_INTEGE_Refigth (nanostream.utils.data_structures.MYSQL_VARCHAR25 attribute), 24 attribute), 27
- max_length (nanostream.utils.data_structures.MYSQL_INT**E** attribute), 24 attribute), 27
- $max_length \, (nanostream.utils.data_structures. MYSQL_INT \textit{EGNER} appth \, (nanostream.utils.data_structures. MYSQL_VARCHAR26 \, attribute), \, 24 \, attribute), \, 27 \, attribute), \, 28 \, attribute), \, 28 \, attribute), \, 28 \, attribute), \, 28 \, attribute), \, 29 \, attribute), \, 29 \, attribute), \, 20 \, attribute), \, 20$
- max_length (nanostream.utils.data_structures.MYSQL_INThotels.data_structures.MYSQL_VARCHAR27 attribute), 24 attribute), 27
- max_length (nanostream.utils.data_structures.MYSQL_INTEGERength (nanostream.utils.data_structures.MYSQL_VARCHAR28 attribute), 24 attribute), 27
- max_length (nanostream.utils.data_structures.MYSQL_VARGH_ARQth (nanostream.utils.data_structures.MYSQL_VARCHAR29 attribute), 25 attribute), 27
- max_length (nanostream.utils.data_structures.MYSQL_VARGHAR3 attribute), 25 attribute), 27
- max_length (nanostream.utils.data_structures.MYSQL_VARGH_ARM)th (nanostream.utils.data_structures.MYSQL_VARCHAR30 attribute), 25
- max_length (nanostream.utils.data_structures.MYSQL_VARQM_ARMM_th (nanostream.utils.data_structures.MYSQL_VARCHAR32 attribute), 25 attribute), 27
- max_length (nanostream.utils.data_structures.MYSQL_VARGHAR32768 attribute), 25

 attribute), 27

 max_length (nanostream.utils.data_structures.MYSQL_VARCHAR32768 attribute), 25

 attribute), 27
- max_length (nanostream.utils.data_structures.MYSQL_VARCHAR4 attribute), 25 attribute), 28
- max_length (nanostream.utils.data_structures.MYSQL_VARGHAR4096 attribute), 25 attribute), 28
- max_length (nanostream.utils.data_structures.MYSQL_VARGH_ARdth (nanostream.utils.data_structures.MYSQL_VARCHAR5 attribute), 25 attribute), 28
- $max_length \ (nanostream.utils.data_structures. MYSQL_VARGHAR512$

attribute), 28		tream.utils.data_structures), 22
max_length (nanostream.utils.data_structures.MYS	SOL VA	
attribute), 28	JQL_ 1/1	tream.utils.data_structures), 22
max_length (nanostream.utils.data_structures.MYS	SOL VA	
attribute), 28		tream.utils.data_structures), 22
max_length (nanostream.utils.data_structures.MYS	SOL VA	
attribute), 28		tream.utils.data_structures), 22
max_length (nanostream.utils.data_structures.MYS	SQL VA	
attribute), 28	-	tream.utils.data_structures), 22
max_length (nanostream.utils.data_structures.MYS	SQL_VA	
attribute), 28		tream.utils.data_structures), 22
max_length (nanostream.utils.data_structures.MYS	SQL_VA	ARCHSQ19_INTEGER25 (class in nanos
attribute), 28		tream.utils.data_structures), 22
MYSQL_BOOL (class in	nanos-	MYSQL_INTEGER256 (class in nanos
tream.utils.data_structures), 20		tream.utils.data_structures), 22
\ 	nanos-	MYSQL_INTEGER26 (class in nanos
tream.utils.data_structures), 20		tream.utils.data_structures), 23
`	nanos-	\
tream.utils.data_structures), 20		tream.utils.data_structures), 23
\	nanos-	\
tream.utils.data_structures), 20		tream.utils.data_structures), 23
\	nanos-	\
tream.utils.data_structures), 20		tream.utils.data_structures), 23
\	nanos-	\(-
tream.utils.data_structures), 20		tream.utils.data_structures), 23
\	nanos-	\
tream.utils.data_structures), 20		tream.utils.data_structures), 23
~ —	nanos-	~ -
tream.utils.data_structures), 21		tream.utils.data_structures), 23
\	nanos-	\
tream.utils.data_structures), 21 MYSQL_INTEGER12 (class in	nonos	tream.utils.data_structures), 23 MYSQL_INTEGER32768 (class in nanos
tream.utils.data structures), 21	nanos-	MYSQL_INTEGER32/68 (class in nanos tream.utils.data_structures), 23
	nanos-	
tream.utils.data_structures), 21	nanos-	tream.utils.data_structures), 23
	nanos-	
tream.utils.data_structures), 21	nanos	tream.utils.data_structures), 23
	nanos-	
tream.utils.data_structures), 21	1141100	tream.utils.data_structures), 24
	nanos-	
tream.utils.data_structures), 21		tream.utils.data_structures), 24
	nanos-	
tream.utils.data_structures), 21		tream.utils.data_structures), 24
	nanos-	
tream.utils.data_structures), 21		tream.utils.data_structures), 24
MYSQL_INTEGER17 (class in	nanos-	MYSQL_INTEGER7 (class in nanos
tream.utils.data_structures), 21		tream.utils.data_structures), 24
MYSQL_INTEGER18 (class in	nanos-	MYSQL_INTEGER8 (class in nanos
tream.utils.data_structures), 22		tream.utils.data_structures), 24
` _	nanos-	` -
tream.utils.data_structures), 22		tream.utils.data_structures), 24
` _	nanos-	`
tream.utils.data_structures), 22		tream.utils.data_structures), 24
MYSQL_INTEGER20 (class in	nanos-	MYSQL_INTEGER_BASE (class in nanos

tream.utils.data_structures), 24			tream.utils.data_structures), 27	
mysql_type() (in module	na	nos- I	\	nos-
tream.utils.data_structures), 29 MYSQL_VARCHAR (class in	n na	nos- l	tream.utils.data_structures), 27 MYSQL_VARCHAR28 (class in na	nos-
tream.utils.data_structures), 24			tream.utils.data_structures), 27	
\	n na	nos- I		nos-
tream.utils.data_structures), 24 MYSQL_VARCHAR1 (class i	n no	nos I	tream.utils.data_structures), 27 MYSQL_VARCHAR3 (class in na	noc
tream.utils.data_structures), 25	n na	nos- I	MYSQL_VARCHAR3 (class in na tream.utils.data structures), 27	nos-
	in na	nos- l	A CHARGO AND CONTRACT AND CONTR	nos-
tream.utils.data_structures), 25	_		tream.utils.data_structures), 27	
MYSQL_VARCHAR1024 (class	in na	nos- I	<u> </u>	nos-
tream.utils.data_structures), 25 MYSQL_VARCHAR11 (class	in no	nos I	tream.utils.data_structures), 27 MYSQL_VARCHAR32 (class in na	nos
tream.utils.data_structures), 25	in na	inos- I	MYSQL_VARCHAR32 (class in na tream.utils.data_structures), 27	nos-
	in na	nos- l	A CAMPAGNA AND CAMPAGNA AND AND AND AND AND AND AND AND AND A	nos-
tream.utils.data_structures), 25		.1105	tream.utils.data_structures), 27	1100
	in na	nos- I	A CHARGO A TANDON A DATA A DAT	nos-
tream.utils.data_structures), 25			tream.utils.data_structures), 27	
MYSQL_VARCHAR13 (class	in na	nos- I	MYSQL_VARCHAR4096 (class in na	nos-
tream.utils.data_structures), 25			tream.utils.data_structures), 28	
	in na	nos- I		nos-
tream.utils.data_structures), 25			tream.utils.data_structures), 28	
	in na	nos- I	\	nos-
tream.utils.data_structures), 25 MYSQL_VARCHAR16 (class	in no	nos I	tream.utils.data_structures), 28 MYSQL_VARCHAR6 (class in na	nos
tream.utils.data_structures), 25	in na	inos- I	tream.utils.data_structures), 28	nos-
MYSQL_VARCHAR16384 (class	in na	nos- l	A THE COLUMN TO	nos-
tream.utils.data_structures), 25			tream.utils.data_structures), 28	
	in na	nos- I	AGUS OF THE CITEDS	nos-
tream.utils.data_structures), 26			tream.utils.data_structures), 28	
MYSQL_VARCHAR18 (class	in na	nos- I	MYSQL_VARCHAR8 (class in na	nos-
tream.utils.data_structures), 26			tream.utils.data_structures), 28	
MYSQL_VARCHAR19 (class	in na			
		nos- l	MYSQL_VARCHAR8192 (class in na	nos-
tream.utils.data_structures), 26			MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28	
MYSQL_VARCHAR2 (class i			MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na	nos-
MYSQL_VARCHAR2 (class i tream.utils.data_structures), 26	n na	nos- l	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28	nos-
MYSQL_VARCHAR2 (class i tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class i	n na	nos- l	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na	
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class in tream.utils.data_structures), 26 displayed in the structure in the structur	n na in na	nos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28	nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class tream.utils.data_structures), 26 class tream.utils.data_structures), 27 class tream.utils.data_structures), 28 class tream.utils.dat	n na in na	nos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na	nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class tream.utils.data_structures), 26	n na in na in na	nnos- I nnos- I nnos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 29	nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class tream.utils.data_structures), 26	n na in na in na	nnos- I nnos- I nnos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na	nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class in tream.utils.data_st	n na in na in na in na	anos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 29	nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class tream.utils.data_structures), 26	n na in na in na in na in na	nnos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 29 N NanoNode (class in nanostream.node), 15 nanostream.message.batch (module), 31	nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class tre	n na in na in na in na in na	nnos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 29 N NanoNode (class in nanostream.node), 15 nanostream.message.batch (module), 31 nanostream.message.canary (module), 31	nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class tream.utils.data_structures), 26	n na in na in na in na in na in na	nnos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 29 N NanoNode (class in nanostream.node), 15 nanostream.message.batch (module), 31 nanostream.message.canary (module), 31 nanostream.message.message (module), 31	nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR24 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR24 (class in tream.utils.data_structures), 26	n na in na in na in na in na in na	nnos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 29 N NanoNode (class in nanostream.node), 15 nanostream.message.batch (module), 31 nanostream.message.canary (module), 31 nanostream.message.message (module), 31 nanostream.message.poison_pill (module), 31	nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR24 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR24 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR24 (class in tream.utils.data_structures), 26	n na in na in na in na in na in na in na	unos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 29 N NanoNode (class in nanostream.node), 15 nanostream.message.batch (module), 31 nanostream.message.canary (module), 31 nanostream.message.message (module), 31 nanostream.message.poison_pill (module), 31 nanostream.message.trigger (module), 31 nanostream.message.trigger (module), 31	nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class tream.utils.data_structures), 26 MYSQL_VARCHAR24 (class tream.utils.data_structures), 26 MYSQL_VARCHAR24 (class tream.utils.data_structures), 26 MYSQL_VARCHAR25 (class tre	n na in na in na in na in na in na in na	unos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 29 N NanoNode (class in nanostream.node), 15 nanostream.message.batch (module), 31 nanostream.message.canary (module), 31 nanostream.message.message (module), 31 nanostream.message.poison_pill (module), 31 nanostream.message.trigger (module), 31 nanostream.node (module), 13	nos- nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class tream.utils.data_structures), 26 MYSQL_VARCHAR24 (class tream.utils.data_structures), 26 MYSQL_VARCHAR25 (class tre	n na in na in na in na in na in na in na	unos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 29 N NanoNode (class in nanostream.node), 15 nanostream.message.batch (module), 31 nanostream.message.canary (module), 31 nanostream.message.message (module), 31 nanostream.message.poison_pill (module), 31 nanostream.message.trigger (module), 31 nanostream.node (module), 13 nanostream.node (module), 13 nanostream.node_classes.network_nodes (module), 2	nos- nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class tream.utils.data_structures), 26 MYSQL_VARCHAR24 (class tream.utils.data_structures), 26 MYSQL_VARCHAR25 (class tre	n na in na in na in na in na in na in na	nnos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 29 N NanoNode (class in nanostream.node), 15 nanostream.message.batch (module), 31 nanostream.message.canary (module), 31 nanostream.message.message (module), 31 nanostream.message.poison_pill (module), 31 nanostream.message.trigger (module), 31 nanostream.node (module), 13 nanostream.node (module), 13 nanostream.node_classes.network_nodes (module), 2 nanostream.node_queue.queue (module), 31	nos- nos-
MYSQL_VARCHAR2 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR20 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR2048 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR21 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR22 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR23 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR24 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR25 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR25 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR256 (class in tream.utils.data_structures), 26 MYSQL_VARCHAR256 (class in tream.utils.data_structures), 27	n na in na	nnos- I	MYSQL_VARCHAR8192 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR9 (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MYSQL_VARCHAR_BASE (class in na tream.utils.data_structures), 28 MySQLTypeSystem (class in na tream.utils.data_structures), 29 N NanoNode (class in nanostream.node), 15 nanostream.message.batch (module), 31 nanostream.message.canary (module), 31 nanostream.message.message (module), 31 nanostream.message.poison_pill (module), 31 nanostream.message.trigger (module), 31 nanostream.node (module), 13 nanostream.node (module), 13 nanostream.node_classes.network_nodes (module), 2	nos- nos-

NanoStreamMessage (class in nanostream.message.message), 31 NanoStreamQueue (class in nanos-	processor() (nanostream.node.NanoNode method), 16 put() (nanostream.node_queue.queue.NanoStreamQueue method), 32
tream.node_queue.queue), 31 no_op() (in module nanostream.node), 19 NothingToSeeHere (class in nanostream.node), 17	python_cast_function (nanostream.utils.data_structures.BOOL attribute), 19
OKBLUE (nanostream.node.bcolors attribute), 18	python_cast_function (nanos- tream.utils.data_structures.DataType attribute), 19
OKGREEN (nanostream.node.bcolors attribute), 18	python_cast_function (nanos- tream.utils.data_structures.FLOAT attribute), 19
PaginatedHttpGetRequest (class in nanostream.node_classes.network_nodes), 30 Parameters (class in nanostream.node), 17	python_cast_function (nanostream.utils.data_structures.INTEGER attribute), 20
PoisonPill (class in nanostream.message.poison_pill), 31 primitive_to_intermediate_type() (in module nanostream.utils.data_structures), 29	python_cast_function (nanostream.utils.data_structures.MYSQL_BOOL attribute), 20
PrimitiveTypeSystem (class in nanostream.utils.data_structures), 29 PrinterOfThings (class in nanostream.node), 17	python_cast_function (nanos- tream.utils.data_structures.MYSQL_ENUM attribute), 20
process_item() (nanostream.node.AggregateValues method), 13 process_item() (nanostream.node.BatchMessages	python_cast_function (nanos- tream.utils.data_structures.MYSQL_INTEGER_BASE attribute), 24
method), 13 process_item() (nanostream.node.CSVReader method), 13	python_cast_function (nanos- tream.utils.data_structures.MYSQL_VARCHAR_BASE attribute), 29
$\begin{array}{c} process_item() (nanostream.node.CSVToDictionaryList\\ method), \ 13 \end{array}$	python_cast_function (nanos- tream.utils.data_structures.STRING attribute), 29
process_item() (nanostream.node.Filter method), 14 process_item() (nanostream.node.GetEnvironmentVariable method), 14	s python_cast_function() (nanostream.utils.data_structures.DATETIME method), 19
process_item() (nanostream.node.InsertData method), 14 process_item() (nanostream.node.LocalFileReader method), 15	python_cast_function() (nanos- tream.utils.data_structures.MYSQL_DATE method), 20
process_item() (nanostream.node.NanoNode method), 16 process_item() (nanostream.node.PrinterOfThings method), 17	PythonTypeSystem (class in nanostream.utils.data_structures), 29
process_item() (nanostream.node.RandomSample method), 17 process_item() (nanostream.node.Remapper method), 17	RandomSample (class in nanostream.node), 17
process_item() (nanostream.node.Keniapper inclind), 17 process_item() (nanostream.node.SequenceEmitter method), 17 process_item() (nanostream.node.Serializer method), 17	Remapper (class in nanostream.node), 17 responses() (nanostream.node_classes.network_nodes.PaginatedHttpGetRed method), 30
process_item() (nanostream.node.SimpleTransforms method), 17	Row (class in nanostream.utils.data_structures), 29
process_item() (nanostream.node.StreamingJoin method), 18	SequenceEmitter (class in nanostream.node), 17
process_item() (nanostream.node.SubstituteRegex method), 18 process_item() (nanostream.node_classes.network_nodes.F	Serializer (class in nanostream.node), 17 setup() (nanostream.node.NanoNode method), 16 Hitch Grant Reference of the company of the setup of the set
method), 30 process_item() (nanostream.node_classes.network_nodes.F	18
method), 30	-

```
sink list()
              (nanostream.node.DynamicClassMediator
         method), 14
source_list() (nanostream.node.DynamicClassMediator
         method), 14
start() (nanostream.node.NanoNode method), 16
stream() (nanostream.node.NanoNode method), 16
StreamingJoin (class in nanostream.node), 18
StreamMySQLTable (class in nanostream.node), 17
STRING (class in nanostream.utils.data structures), 29
SubstituteRegex (class in nanostream.node), 18
Т
template_class() (in module nanostream.node), 19
terminate_pipeline()
                          (nanostream.node.NanoNode
         method), 16
thread_monitor() (nanostream.node.NanoNode method),
time_running (nanostream.node.NanoNode attribute), 17
TimeWindowAccumulator (class in nanostream.node), 18
to_intermediate_type()
         tream.utils.data structures.DataType method),
to_python() (nanostream.utils.data_structures.DataType
         method), 19
Trigger (class in nanostream.message.trigger), 31
type_mapping()
                                               (nanos-
         tream.utils.data_structures.DataSourceTypeSystem
         static method), 19
type_mapping()
                                               (nanos-
         tream.utils.data_structures.MySQLTypeSystem
         static method), 29
type_system (nanostream.utils.data_structures.DataType
         attribute), 19
U
UNDERLINE (nanostream.node.bcolors attribute), 18
W
WARNING (nanostream.node.bcolors attribute), 18
```