JARVIS

Generated by Doxygen 1.12.0

Chapter 1

Namespace Index

1.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 JARVIS Namespace Reference

JARVIS is a logging package designed to help developers manage complex logging scenarios.

5.1.1 Detailed Description

JARVIS is a logging package designed to help developers manage complex logging scenarios.

JARVIS is a logging package designed to aid in debugging, formatting output, and managing complex logging contexts.

JARVIS is a logging package designed to aid in debugging and formatting output.

It provides various utilities for logging messages at different levels, managing contexts, and testing functionality. This module defines the main JARVIS class, which integrates all the utility functions and logging capabilities into a single, easy-to-use interface.

This segment of the package introduces the JarvisPrintingMixin class, which provides utility functions for formatting and printing output, as well as managing indentation levels during the logging process.

This module introduces the JarvisUtilitiesMixin class, which provides various utility functions to manage and manipulate logging contexts, handle naming conventions, and trigger appropriate log messages based on test results.

5.2 jarvis Namespace Reference

Classes

- class JARVIS

  Main class for the JARVIS logging system.

5.3 jarvis_printing Namespace Reference

Classes

- class JarvisPrintingMixin

5.4 jarvis_utilities Namespace Reference

Classes

- class JarvisUtilitiesMixin

  Mixin class providing utility functions for the Logger class.

5.5 launch_tests Namespace Reference

Functions

- _importHelp (folder)
- importTests ()

Variables

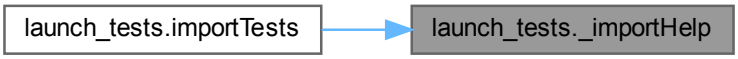
- testLib = importTests()

5.5.1 Function Documentation

5.5.1.1 _importHelp()

```
launch_tests._importHelp (
              folder)  [protected]
```

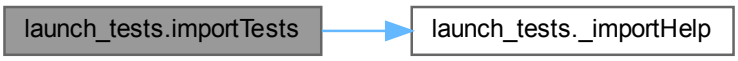
Here is the caller graph for this function:



5.5.1.2 importTests()

```
launch_tests.importTests ()
```

Here is the call graph for this function:



5.5.2 Variable Documentation

5.5.2.1 testLib

```
launch_tests.testLib = importTests()
```

5.6 TEST_GPT_example Namespace Reference

Variables

- event1 = Event(230, False)
- handled
- ID_SUP_100
- ID_SUP_150
- test_handled

5.6.1 Variable Documentation

5.6.1.1 event1

```
TEST_GPT_example.event1 = Event(230, False)
```

5.6.1.2 handled

```
TEST_GPT_example.handled
```

5.6.1.3 ID_SUP_100

```
TEST_GPT_example.ID_SUP_100
```

Initial value:

```
00001 =  jarvis.AddTest(
00002     test_fn=is_more_100["test_fn"],
00003     name="Event id >100"
00004 )
```

5.6.1.4 ID_SUP_150

```
TEST_GPT_example.ID_SUP_150
```

Initial value:

```
00001 =  jarvis.AddTest(
00002     test_fn=is_more_150["test_fn"],
00003     name="Event id >150"
00004 )
```

5.6.1.5 test_handled

```
TEST_GPT_example.test_handled
```

Initial value:

```
00001 =  jarvis.AddTest(
00002     test_fn=lambda x: x,
00003     name="Event handled"
00004 )
```

5.7 test_helpers Namespace Reference

Classes

- class Event

  Represents a MIDI event.

Variables

- is_more_100
- is_more_150
- jarvis = JARVIS("INFO")

5.7.1 Variable Documentation

5.7.1.1 is_more_100

```
test_helpers.is_more_100
```

Initial value:

```
00001 =  jarvis.AddTest(
00002     test_fn=lambda x: x > 100,
00003     name=">100"
00004 )
```

5.7.1.2 is_more_150

```
test_helpers.is_more_150
```

Initial value:

```
00001 =  jarvis.AddTest(
00002     test_fn=lambda x: x > 150,
00003     name=">150"
00004 )
```

5.7.1.3 jarvis

```
test_helpers.jarvis = JARVIS("INFO")
```

5.8 TEST_minimal_example Namespace Reference

Variables

- event1 = Event(230, False)
- handled
- ID_SUP_100
- ID_SUP_150
- test_handled

5.8.1 Variable Documentation

5.8.1.1 event1

```
TEST_minimal_example.event1 = Event(230, False)
```

5.8.1.2 handled

```
TEST_minimal_example.handled
```

5.8.1.3 ID_SUP_100

```
TEST_minimal_example.ID_SUP_100
```

Initial value:

```
00001 =  jarvis.AddTest(
00002     test_fn=is_more_100["test_fn"],
00003     name="Event id >100"
00004 )
```

5.8.1.4 ID_SUP_150

```
TEST_minimal_example.ID_SUP_150
```

Initial value:

```
00001 =  jarvis.AddTest(
00002     test_fn=is_more_150["test_fn"],
00003     name="Event id >150"
00004 )
```

5.8.1.5 test_handled

```
TEST_minimal_example.test_handled
```

Initial value:

```
00001 =  jarvis.AddTest(
00002     test_fn=lambda x: x,
00003     name="Event handled"
00004 )
```

Chapter 6

Class Documentation

6.1 test_helpers.Event Class Reference

Represents a MIDI event.

Public Member Functions

- __init__ (self, event_id, handled=False)

  Initializes a new instance of the Event class.
- mark_handled (self)

  Marks the event as handled.
- set_event_details (self, event_type, channel, data)

  Set the MIDI event details.

Public Attributes

- channel = None
- list data = [ ]
- bool handled = handled
- id = event_id
- type = None

6.1.1 Detailed Description

Represents a MIDI event.

The Event class encapsulates data related to MIDI events. This includes the event ID, whether the event has been handled, and potentially other MIDI-specific information such as the type of event (e.g., Note On, Note Off), the channel number, and additional data bytes (e.g., note number, velocity).

6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

```
test_helpers.Event.__init__ (
               self,
               event_id,
               handled = False)
```

Initializes a new instance of the Event class.

The constructor initializes the event with an ID and a handled status. The handled attribute is set to False by default, indicating that the event has not yet been processed.

Parameters

event_id	The unique identifier for the event.
handled	A boolean indicating whether the event has been handled (default is False).

6.1.3 Member Function Documentation

6.1.3.1 mark_handled()

```
test_helpers.Event.mark_handled (
              self)
```

Marks the event as handled.

This method updates the handled attribute to True, indicating that the event has been processed.

6.1.3.2 set_event_details()

```
test_helpers.Event.set_event_details (
              self,
              event_type,
              channel,
              data)
```

Set the MIDI event details.

This method allows setting the type, channel, and data bytes of the MIDI event.

Parameters

event_type	The type of MIDI event (e.g., "Note On", "Control Change").
channel	The MIDI channel number (0-15).
data	A list of data bytes associated with the event.

6.1.4 Member Data Documentation

6.1.4.1 channel

```
test_helpers.Event.channel = None
```

6.1.4.2 data

```
list test_helpers.Event.data = []
```

6.1.4.3 handled

```
bool test_helpers.Event.handled = handled
```

6.1.4.4 id

```
test_helpers.Event.id = event_id
```

6.1.4.5 type

```
test_helpers.Event.type = None
```

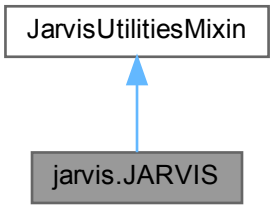
The documentation for this class was generated from the following file:

- tests/test_helpers.py

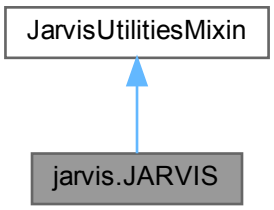
6.2 jarvis.JARVIS Class Reference

Main class for the JARVIS logging system.

Inheritance diagram for jarvis.JARVIS:



Collaboration diagram for jarvis.JARVIS:



Public Member Functions

- __init__ (self, level="WARNING")

  Initializes the JARVIS logger with a default logging level.
- AddTest (self, test_fn=lambda:False, result_key=True, callback_true=None, callback_false=None, name="test")

  Add a new test to the current context.
- Cb_False (self, message, level="FAIL")

  Generate a failure callback message.
- Cb_True (self, message, level="SUCCESS")

  Generate a success callback message.
- Debug (self, message)

  Log a debug message.
- Error (self, message)

  Log an error message.
- Log (self, message, level)

  Log a message at a specified level.
- Navigate (self, destination="children")

  Navigate to a different context within the logger.
- TriggerTest (self, test, val)

  Trigger a specific test.
- Warning (self, message)

  Log a warning message.
- WriteLog (self)

  Print all the log entries stored in `msg_log`.

Public Attributes

- dict contexts = {}
- list current_path = [ ]
- level = level
- dict levels
- list msg_log = [ ]
- int print_lvl = -1
- list tests = [ ]
- int unnamed_tests = 0

6.2.1 Detailed Description

Main class for the JARVIS logging system.

The JARVIS class extends JarvisUtilitiesMixin, incorporating a variety of logging and utility functions. This class provides methods for logging messages at different levels, navigating between contexts, adding and triggering tests, and generating callback messages. The class is designed to be the central point of interaction for the JARVIS logging system.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__()

```
jarvis.JARVIS.__init__ (
              self,
              level = "WARNING")
```

Initializes the JARVIS logger with a default logging level.

This constructor sets up the initial state of the JARVIS logger, including the logging level, context hierarchy, and test management. It also initializes internal variables that track indentation levels, logging contexts, and the log itself.

Parameters

level	The default logging level (defaults to "WARNING").

```
Initialize the Logger with a default logging level.

:param level: Default logging level, defaults to "WARNING".
```

6.2.3 Member Function Documentation

6.2.3.1 AddTest()

```
jarvis.JARVIS.AddTest (
              self,
              test_fn = lambda: False,
              result_key = True,
              callback_true = None,
              callback_false = None,
              name = "test")
```

Add a new test to the current context.

This method adds a new test to the current context within the logging hierarchy. The test is defined by a test function, expected result, and optional callback functions for handling success and failure. The test is automatically named to avoid conflicts, and the method returns the test object.

Parameters

test_fn	The test function to evaluate (defaults to a lambda returning False).
result_key	Expected result key for the test (defaults to True).
callback_true	Callback function if the test passes (defaults to a success message).
callback_false	Callback function if the test fails (defaults to a failure message).
name	Name of the test (defaults to "test").

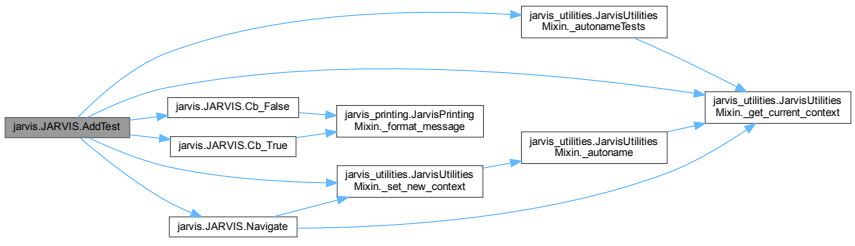
Returns

The newly added test object.

```
Add a new test to the current context.

:param test_fn: The test function to evaluate.
:param result_key: Expected result key for the test.
:param callback_true: Callback function if the test passes.
:param callback_false: Callback function if the test fails.
:param name: Name of the test.
:return: The newly added test object.
```

Here is the call graph for this function:



6.2.3.2 Cb_False()

```
jarvis.JARVIS.Cb_False (
              self,
              message,
              level = "FAIL")
```

Generate a failure callback message.

This method generates a callback message indicating that a test has failed. The message is formatted with a "FAIL" level by default.

Parameters

message	The message to format.
level	The logging level (defaults to "FAIL").

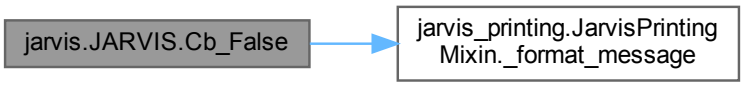
Returns

The formatted message.

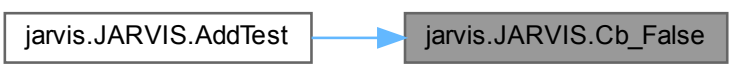
```
Generate a failure callback message.

:param message: The message to format.
:param level: The logging level, defaults to "FAIL".
:return: The formatted message.
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.3.3 Cb_True()

```
jarvis.JARVIS.Cb_True (
              self,
              message,
              level = "SUCCESS")
```

Generate a success callback message.

This method generates a callback message indicating that a test has passed. The message is formatted with a "SUCCESS" level by default.

Parameters

message	The message to format.
level	The logging level (defaults to "SUCCESS").

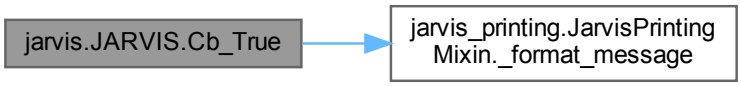
Returns

The formatted message.

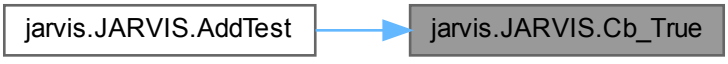
```
Generate a success callback message.

:param message: The message to format.
:param level: The logging level, defaults to "SUCCESS".
:return: The formatted message.
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.3.4 Debug()

```
jarvis.JARVIS.Debug (
            self,
            message)
```

Log a debug message.

This method logs a message at the "DEBUG" level, which is typically used for detailed debugging information that may not be required during normal operation.

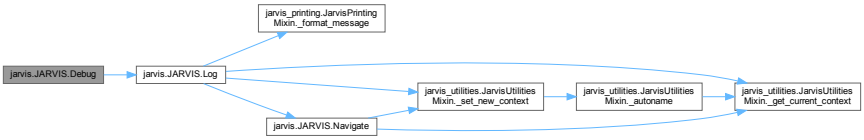
Parameters

message	The debug message to log.

```
Log a debug message.

:param message: The debug message to log.
```

Here is the call graph for this function:



6.2.3.5 Error()

```
jarvis.JARVIS.Error (
            self,
            message)
```

Log an error message.

This method logs a message at the "ERROR" level, which is typically used for logging serious issues that might cause the application to behave unexpectedly.

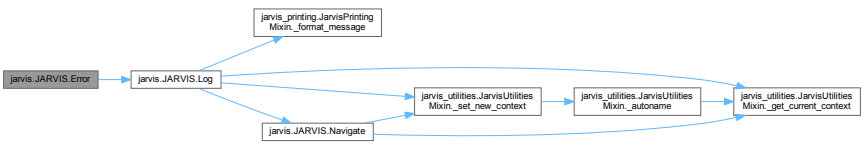
Parameters

message	The error message to log.

```
Log an error message.

:param message: The error message to log.
```

Here is the call graph for this function:



6.2.3.6 Log()

```
jarvis.JARVIS.Log (
            self,
            message,
            level)
```

Log a message at a specified level.

This method logs a message at the given level by navigating to the appropriate context, formatting the message, and storing it in the current context. It also returns the formatted message.

Parameters

message	The message to log.
level	The level at which to log the message.

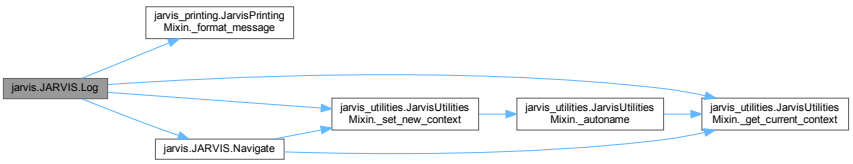
Returns

The formatted message.

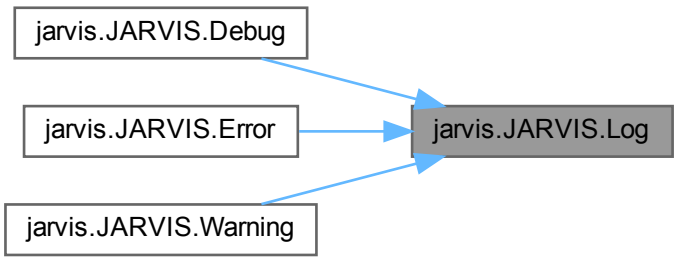
```
Log a message at the given level.

:param message: The message to log.
:param level: The level at which to log the message.
:return: The formatted message.
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.3.7 Navigate()

```
jarvis.JARVIS.Navigate (
            self,
            destination = "children")
```

Navigate to a different context within the logger.

This method allows the user to move between different contexts within the logging hierarchy. It can navigate to a child context or move up to the parent context. If the destination context does not exist, it creates a new one.

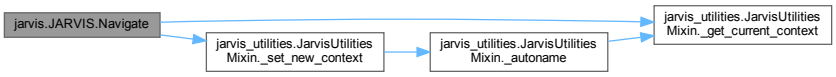
Parameters

destination	The context to navigate to (defaults to "children").

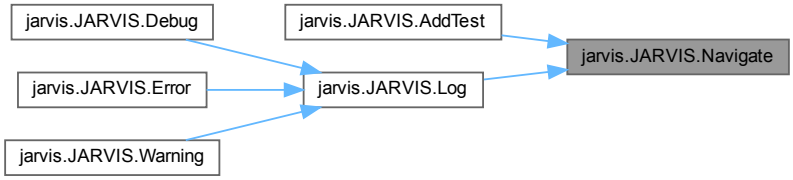
```
Navigate to a different context.

:param destination: The context to navigate to. Defaults to "children".
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.3.8 TriggerTest()

```
jarvis.JARVIS.TriggerTest (
            self,
            test,
            val)
```

Trigger a specific test.

This method triggers a specific test by executing the test function with the provided value. It then logs appropriate success or failure messages based on the test result and updates the test's status.

Parameters

test	The test object to trigger.
val	The value to pass to the test function.

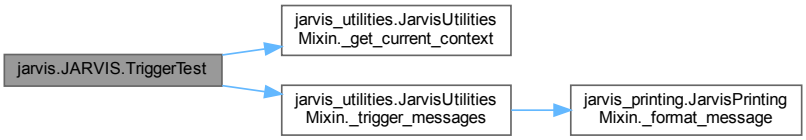
Returns

The test object after execution.

```
Trigger a specific test.

:param test: The test object to trigger.
:param val: The value to pass to the test function.
:return: The test object after execution.
```

Here is the call graph for this function:



6.2.3.9 Warning()

```
jarvis.JARVIS.Warning (
              self,
              message)
```

Log a warning message.

This method logs a message at the "WARNING" level, which is typically used for logging potentially harmful situations or important notices that require attention.

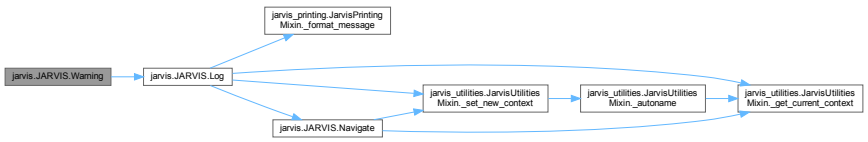
Parameters

message	The warning message to log.

```
Log a warning message.

:param message: The warning message to log.
```

Here is the call graph for this function:



6.2.3.10 WriteLog()

```
jarvis.JARVIS.WriteLog (
              self)
```

Print all the log entries stored in `msg_log`.

This method iterates over all log entries stored in the `msg_log` list and prints them. This provides a simple way to output the entire log to the console.

```
Print all the log entries stored in `msg_log`.
```

6.2.4 Member Data Documentation

6.2.4.1 contexts

```
dict jarvis.JARVIS.contexts = {}
```

6.2.4.2 current_path

```
jarvis.JARVIS.current_path = []
```

6.2.4.3 level

```
jarvis.JARVIS.level = level
```

6.2.4.4 levels

```
dict jarvis.JARVIS.levels
```

Initial value:

```
=  {
            "INFO": 0,
            "DEBUG": 10,
            "FAIL": 15,
            "SUCCESS": 15,
            "WARNING": 20,
            "ERROR": 30
        }
```

6.2.4.5 msg_log

```
list jarvis.JARVIS.msg_log = []
```

6.2.4.6 print_lvl

```
int jarvis.JARVIS.print_lvl = -1
```

6.2.4.7 tests

```
list jarvis.JARVIS.tests = []
```

6.2.4.8 unnamed_tests

```
int jarvis.JARVIS.unnamed_tests = 0
```

The documentation for this class was generated from the following file:

- JARVIS/jarvis.py

6.3 jarvis_printing.JarvisPrintingMixin Class Reference

Public Attributes

- contexts
- current_path

Protected Member Functions

- _format_message (self, level, message, name="", ignore=False)

  Formats a log message based on its level and other parameters.
- _print (self, name, val="")

  Prints a debug message with the current indentation level.
- _print_contexts (self)

  Prints the current state of contexts for debugging purposes.
- _println (self, fn_name)

  Increases indentation level and prints an entering message.
- _printOut (self, fn_name)

  Decreases indentation level and prints an exiting message.

6.3.1 Member Function Documentation

6.3.1.1 _format_message()

```
jarvis_printing.JarvisPrintingMixin._format_message (
              self,
              level,
              message,
              name = "",
              ignore = False)  [protected]
```

Formats a log message based on its level and other parameters.

This method creates a formatted log message that includes information such as the logging level, an optional name, and the message content. It can also decide whether to append the message to the log based on the ignore parameter.

Parameters

level	The logging level (e.g., SUCCESS, DEBUG, ERROR).
message	The message to format.
name	Optional name to include in the message.
ignore	Whether to ignore adding the message to the log.

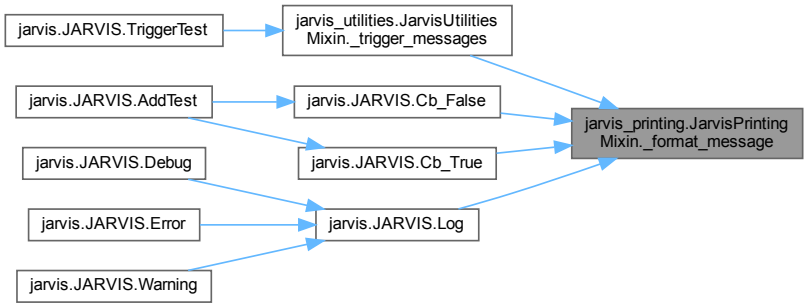
Returns

The formatted message.

```
Format the log message based on its level.

:param level: The logging level.
:param message: The message to format.
:param name: Optional name to include in the message.
:param ignore: Whether to ignore adding the message to the log.
:return: The formatted message.
```

Here is the caller graph for this function:



6.3.1.2 _print()

```
jarvis_printing.JarvisPrintingMixin._print (
            self,
            name,
            val = "")  [protected]
```

Prints a debug message with the current indentation level.

This method outputs a debug message that is formatted according to the current level of indentation. It is useful for providing contextual information during the execution of the program.

Parameters

name	The name of the debug message.
val	The value or content of the debug message.

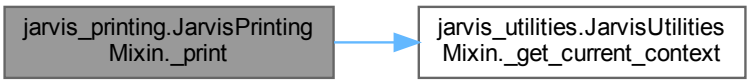
Returns

None

```
Print a debug message with the current indentation.

:param name: The name of the debug message.
:param val: The value or content of the debug message.
```

Here is the call graph for this function:



Here is the caller graph for this function:

6.3.1.3 _print_contexts()

```
jarvis_printing.JarvisPrintingMixin._print_contexts (
              self)  [protected]
```

Prints the current state of contexts for debugging purposes.

This method outputs the current state of various contexts, which can be helpful for understanding the internal state of the program at a particular point in time.

Returns

None

```
Print the current state of contexts for debugging.
```

Here is the call graph for this function:

6.3.1.4 _printIn()

```
jarvis_printing.JarvisPrintingMixin._printIn (
              self,
              fn_name)  [protected]
```

Increases indentation level and prints an entering message.

This method increments the indentation level and prints a message indicating the entry into a specific function. This is useful for tracking the flow of execution and understanding the nested structure of function calls.

Parameters

fn_name	The name of the function being entered.

Returns

None

```
Increase indentation level and print entering message.

:param fn_name: The name of the function being entered.
```

6.3.1.5 _printOut()

```
jarvis_printing.JarvisPrintingMixin._printOut (
              self,
              fn_name)  [protected]
```

Decreases indentation level and prints an exiting message.

This method decreases the indentation level and prints a message indicating the exit from a specific function. This helps in tracking when functions are completed and the flow of control is returning to a higher level.

Parameters

fn_name	The name of the function being exited.

Returns

None

```
Decrease indentation level and print exiting message.

:param fn_name: The name of the function being exited.
```

6.3.2 Member Data Documentation

6.3.2.1 contexts

```
jarvis_printing.JarvisPrintingMixin.contexts
```

6.3.2.2 current_path

`jarvis_printing.JarvisPrintingMixin.current_path`

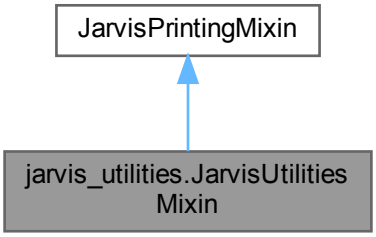
The documentation for this class was generated from the following file:

- JARVIS/jarvis_printing.py

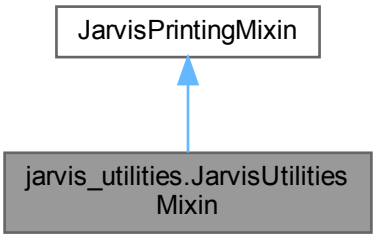
6.4 jarvis_utilities.JarvisUtilitiesMixin Class Reference

Mixin class providing utility functions for the Logger class.

Inheritance diagram for jarvis_utilities.JarvisUtilitiesMixin:



Collaboration diagram for jarvis_utilities.JarvisUtilitiesMixin:



Public Attributes

- contexts
- # If no current path, initialize the root context current_path = context.get(part, context)
- dict current_path = {}

Protected Member Functions

- _autoname (self, name="children", parent="")

  Automatically generates a name for a new context to avoid conflicts.
- _autonameTests (self, name="test", parent="TESTS")

  Automatically generates a unique name for a new test.
- _get_current_context (self)

  Retrieves the current context based on the current path.
- _set_new_context (self, name="children", parent=None)

  Sets a new context in the logger's hierarchy.
- _trigger_messages (self, result, test)

  Triggers success or failure messages based on the result of a test.

6.4.1 Detailed Description

Mixin class providing utility functions for the Logger class.

The JarvisUtilitiesMixin class extends the JarvisPrintingMixin class and includes several utility functions that manage logging contexts, handle automatic naming of contexts and tests, and trigger messages based on the outcomes of tests. These utilities are designed to work seamlessly with the logging system in JARVIS.

```
Mixin class for various utility functions used by the Logger class.
```

6.4.2 Member Function Documentation

6.4.2.1 _autoname()

```
jarvis_utilities.JarvisUtilitiesMixin._autoname (
              self,
              name = "children",
              parent = "")  [protected]
```

Automatically generates a name for a new context to avoid conflicts.

This method generates a name for a new context, ensuring that it does not overwrite existing contexts with the same name. It appends a number to the name if a conflict is detected, making it unique within the parent context.

Parameters

name	The desired context name (defaults to "children").
parent	The parent context name (defaults to an empty string).

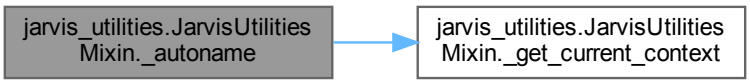
Returns

The auto-generated name.

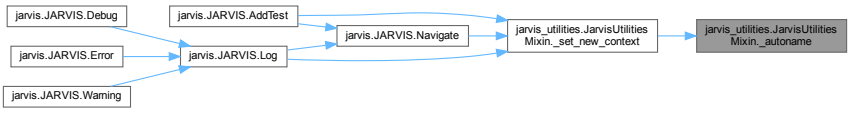
```
Automatically name the context to avoid overwriting existing contexts.

:param name: The desired context name.
:param parent: The parent context name, defaults to an empty string.
:return: The auto-generated name.
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.2.2 _autonameTests()

```
jarvis_utilities.JarvisUtilitiesMixin._autonameTests (
              self,
              name = "test",
              parent = "TESTS")  [protected]
```

Automatically generates a unique name for a new test.

This method automatically names a new test within a specific parent context (by default, the "TESTS" context). It ensures that the test name is unique by appending a number if necessary, which prevents overwriting existing tests.

Parameters

name	The desired test name (defaults to "test").
parent	The parent context name (defaults to "TESTS").

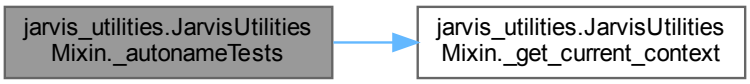
Returns

The auto-generated test name.

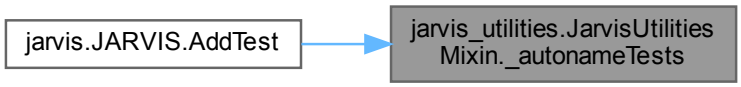
```
Automatically name the test to avoid overwriting existing tests.

:param name: The desired test name.
:param parent: The parent context name, defaults to "TESTS".
:return: The auto-generated test name.
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.2.3 _get_current_context()

```
jarvis_utilities.JarvisUtilitiesMixin._get_current_context (
              self)  [protected]
```

Retrieves the current context based on the current path.

This method navigates the hierarchy of contexts according to the current path and returns the dictionary representing the current context. It allows for easy access to the current working context within the logger.

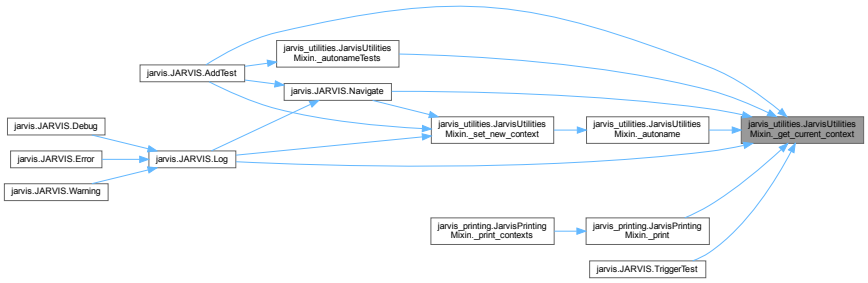
Returns

The current context dictionary.

```
Get the current context based on the current path.

:return: The current context dictionary.
```

Here is the caller graph for this function:



6.4.2.4 _set_new_context()

```
jarvis_utilities.JarvisUtilitiesMixin._set_new_context (
              self,
              name = "children",
              parent = None)  [protected]
```

Sets a new context in the logger's hierarchy.

This method creates and sets a new context within the logging hierarchy. It can either initialize a root context if none exists, or it navigates the existing hierarchy to add a new sub-context. The new context is named either as specified or automatically to avoid conflicts.

Parameters

name	The name of the new context (defaults to "children").
parent	The parent context in which to create the new context (defaults to None).

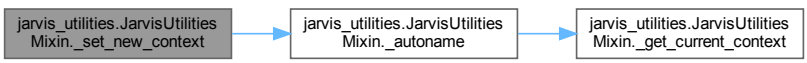
Returns

None

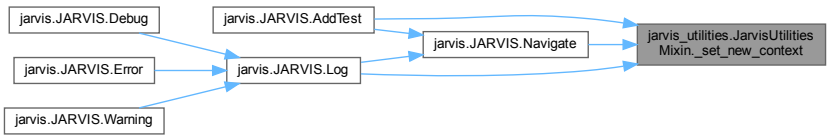
```
Set a new context in the logger's hierarchy.

:param name: The name of the new context.
:param parent: The parent context, defaults to None.
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.2.5 _trigger_messages()

```
jarvis_utilities.JarvisUtilitiesMixin._trigger_messages (
              self,
              result,
              test)  [protected]
```

Triggers success or failure messages based on the result of a test.

This method appends appropriate success or failure messages to the log, depending on the outcome of a test. It also handles the callback functions associated with the test, which are triggered when a test passes or fails. The method ensures that the correct sequence of messages is logged, providing clear feedback on the test's status.

Parameters

result	The result of the test (True for success, False for failure).
test	The test object containing information about the test and callbacks.

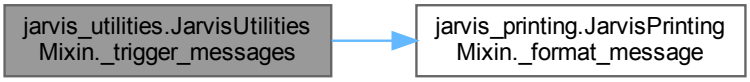
Returns

None

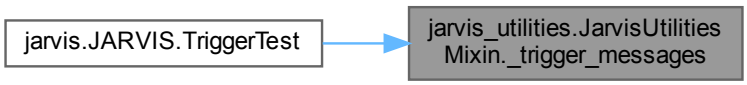
```
Trigger success or failure messages based on the test result.

:param result: The result of the test (True/False).
:param test: The test object.
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.3 Member Data Documentation

6.4.3.1 contexts

```
jarvis_utilities.JarvisUtilitiesMixin.contexts
```

6.4.3.2 current_path [1/2]

```
jarvis_utilities.JarvisUtilitiesMixin.current_path = context.get(part, context)
```

6.4.3.3 current_path [2/2]

```
dict jarvis_utilities.JarvisUtilitiesMixin.current_path = {}
```

The documentation for this class was generated from the following file:

- JARVIS/jarvis_utilities.py

Chapter 7

File Documentation

7.1 JARVIS/jarvis.py File Reference

Classes

- class jarvis.JARVIS

  Main class for the JARVIS logging system.

Namespaces

- namespace JARVIS

  JARVIS is a logging package designed to help developers manage complex logging scenarios.
- namespace jarvis

7.2 JARVIS/jarvis_printing.py File Reference

Classes

- class jarvis_printing.JarvisPrintingMixin

Namespaces

- namespace JARVIS

  JARVIS is a logging package designed to help developers manage complex logging scenarios.
- namespace jarvis_printing

7.3 JARVIS/jarvis_utilities.py File Reference

Classes

- class jarvis_utilities.JarvisUtilitiesMixin

  Mixin class providing utility functions for the Logger class.

Namespaces

- namespace JARVIS

  JARVIS is a logging package designed to help developers manage complex logging scenarios.
- namespace jarvis_utilities

7.4 launch_tests.py File Reference

Namespaces

- namespace launch_tests

Functions

- launch_tests._importHelp (folder)
- launch_tests.importTests ()

Variables

- launch_tests.testLib = importTests()

7.5 tests/TEST_GPT_example.py File Reference

Namespaces

- namespace TEST_GPT_example

Variables

- TEST_GPT_example.event1 = Event(230, False)
- TEST_GPT_example.handled
- TEST_GPT_example.ID_SUP_100
- TEST_GPT_example.ID_SUP_150
- TEST_GPT_example.test_handled

7.6 tests/test_helpers.py File Reference

Classes

- class test_helpers.Event

  Represents a MIDI event.

Namespaces

- namespace test_helpers

Variables

- test_helpers.is_more_100
- test_helpers.is_more_150
- test_helpers.jarvis = JARVIS("INFO")

7.7 tests/TEST_minimal_example.py File Reference

Namespaces

- namespace TEST_minimal_example

Variables

- TEST_minimal_example.event1 = Event(230, False)
- TEST_minimal_example.handled
- TEST_minimal_example.ID_SUP_100
- TEST_minimal_example.ID_SUP_150
- TEST_minimal_example.test_handled

Index