

# nf.io: A File System Abstraction for NFV Orchestration

Generated by Doxygen 1.8.6

Mon Jan 25 2016 11:16:56



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	<a href="#">fuse.c_stat Class Reference</a>	5
3.2	<a href="#">fuse.c_statvfs Class Reference</a>	5
3.3	<a href="#">fuse.c_timespec Class Reference</a>	5
3.4	<a href="#">fuse.c_utimbuf Class Reference</a>	6
3.5	<a href="#">hypervisor.docker_driver.Docker Class Reference</a>	6
3.5.1	Detailed Description	7
3.5.2	Constructor & Destructor Documentation	7
3.5.2.1	<code>__init__</code>	7
3.5.3	Member Function Documentation	7
3.5.3.1	<code>deploy</code>	7
3.5.3.2	<code>get_id</code>	7
3.6	<a href="#">fuse.FUSE Class Reference</a>	8
3.6.1	Detailed Description	9
3.6.2	Constructor & Destructor Documentation	9
3.6.2.1	<code>__init__</code>	9
3.6.3	Member Data Documentation	9
3.6.3.1	<code>OPTIONS</code>	9
3.7	<a href="#">fuse.fuse_context Class Reference</a>	9
3.8	<a href="#">fuse.fuse_file_info Class Reference</a>	10
3.9	<a href="#">fuse.fuse_operations Class Reference</a>	10
3.10	<a href="#">fuse.FuseOSError Class Reference</a>	10
3.11	<a href="#">hypervisor.hypervisor_base.HypervisorBase Class Reference</a>	10
3.11.1	Detailed Description	11
3.11.2	Member Function Documentation	11
3.11.2.1	<code>deploy</code>	11

3.11.2.2	destroy	11
3.11.2.3	execute_in_guest	11
3.11.2.4	get_id	12
3.11.2.5	guest_status	12
3.11.2.6	pause	12
3.12	hypervisor.hypervisor_factory.HypervisorFactory Class Reference	12
3.12.1	Detailed Description	12
3.12.2	Constructor & Destructor Documentation	13
3.12.2.1	__init__	13
3.12.3	Member Function Documentation	13
3.12.3.1	get_hypervisor_instance	13
3.13	hypervisor.libvirt_driver.Libvirt Class Reference	13
3.14	fuse.LoggingMixIn Class Reference	13
3.15	nfio.Nfio Class Reference	14
3.15.1	Constructor & Destructor Documentation	15
3.15.1.1	__init__	15
3.15.2	Member Function Documentation	15
3.15.2.1	getattr	15
3.15.2.2	mkdir	15
3.15.2.3	read	15
3.15.2.4	write	16
3.16	fuse.Operations Class Reference	16
3.16.1	Detailed Description	17
3.16.2	Member Function Documentation	17
3.16.2.1	create	17
3.16.2.2	getattr	17
3.16.2.3	init	18
3.16.2.4	open	18
3.16.2.5	readdir	18
3.16.2.6	statfs	18
3.17	vnfs_operations.VNFSOperations Class Reference	18
3.17.1	Detailed Description	19
3.17.2	Member Function Documentation	19
3.17.2.1	vnfs_create_vnf_instance	19
3.17.2.2	vnfs_deploy_nf	19
3.17.2.3	vnfs_get_file_name	19
3.17.2.4	vnfs_get_instance_configuration	20
3.17.2.5	vnfs_get_nf_type	20
3.17.2.6	vnfs_get_opcode	20
3.17.2.7	vnfs_get_pkt_drops	20

---

3.17.2.8 vnfs_get_rx_bytes . . . . .	20
3.17.2.9 vnfs_get_status . . . . .	20
3.17.2.10 vnfs_get_tx_bytes . . . . .	20
3.17.2.11 vnfs_is_nf_instance . . . . .	21
3.17.2.12 vnfs_stop_vnf . . . . .	21
 Index	 22



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

fuse.LoggingMixIn . . . . .	13
object	
fuse.FUSE . . . . .	8
fuse.Operations . . . . .	16
nfio.Nfio . . . . .	14
hypervisor.hypervisor_base.HypervisorBase . . . . .	10
hypervisor.hypervisor_factory.HypervisorFactory . . . . .	12
OSError	
fuse.FuseOSError . . . . .	10
Structure	
fuse.c_stat . . . . .	5
fuse.c_statvfs . . . . .	5
fuse.c_statvfs . . . . .	5
fuse.c_timespec . . . . .	5
fuse.c_utimbuf . . . . .	6
fuse.fuse_context . . . . .	9
fuse.fuse_file_info . . . . .	10
fuse.fuse_operations . . . . .	10
vnfs_operations.VNFSOperations . . . . .	18
HypervisorBase	
hypervisor.docker_driver.Docker . . . . .	6
hypervisor.libvirt_driver.Libvirt . . . . .	13





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">fuse.c_stat</a>	5
<a href="#">fuse.c_statvfs</a>	5
<a href="#">fuse.c_timespec</a>	5
<a href="#">fuse.c_utimbuf</a>	6
<a href="#">hypervisor.docker_driver.Docker</a>	
Hypervisor driver for <a href="#">Docker</a>	6
<a href="#">fuse.FUSE</a>	
This class is the lower level interface and should not be subclassed under normal use	8
<a href="#">fuse.fuse_context</a>	9
<a href="#">fuse.fuse_file_info</a>	10
<a href="#">fuse.fuse_operations</a>	10
<a href="#">fuse.FuseOSError</a>	10
<a href="#">hypervisor.hypervisor_base.HypervisorBase</a>	
Base class for hypervisors	10
<a href="#">hypervisor.hypervisor_factory.HypervisorFactory</a>	
A singleton class for creating hypervisor driver objects	12
<a href="#">hypervisor.libvirt_driver.Libvirt</a>	13
<a href="#">fuse.LoggingMixIn</a>	13
<a href="#">nfio.Nfio</a>	14
<a href="#">fuse.Operations</a>	
This class should be subclassed and passed as an argument to <a href="#">FUSE</a> on initialization	16
<a href="#">vnfs_operations.VNFSOperations</a>	
Provides a common set of operations for nfio	18

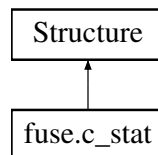


## Chapter 3

# Class Documentation

### 3.1 fuse.c\_stat Class Reference

Inheritance diagram for fuse.c\_stat:

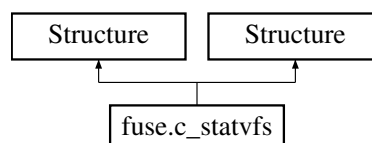


The documentation for this class was generated from the following file:

- /home/mfbari/works/WatNFV/nf.io/src/fuse.py

### 3.2 fuse.c\_statvfs Class Reference

Inheritance diagram for fuse.c\_statvfs:

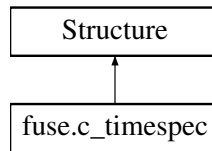


The documentation for this class was generated from the following file:

- /home/mfbari/works/WatNFV/nf.io/src/fuse.py

### 3.3 fuse.c\_timespec Class Reference

Inheritance diagram for fuse.c\_timespec:

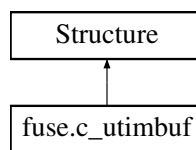


The documentation for this class was generated from the following file:

- /home/mfbari/works/WatNFV/nf.io/src/fuse.py

### 3.4 fuse.c\_utimbuf Class Reference

Inheritance diagram for fuse.c\_utimbuf:



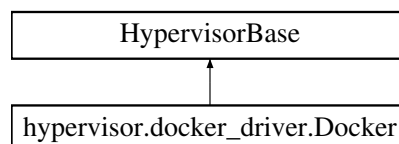
The documentation for this class was generated from the following file:

- /home/mfbari/works/WatNFV/nf.io/src/fuse.py

### 3.5 hypervisor.docker\_driver.Docker Class Reference

Hypervisor driver for [Docker](#).

Inheritance diagram for hypervisor.docker\_driver.Docker:



#### Public Member Functions

- def [\\_\\_init\\_\\_](#)  
*Instantiates a [Docker](#) object.*
- def [get\\_id](#)  
*Returns a container's ID.*
- def [deploy](#)  
*Deploys a docker container.*
- def [start](#)  
*Starts a docker container.*
- def [restart](#)  
*Restarts a docker container.*
- def [stop](#)

- Stops a docker container.*
  - def [pause](#)
*Pauses a docker container.*
  - def [unpause](#)
*Unpauses a docker container.*
  - def [destroy](#)
*Destroys a docker container.*
  - def [execute\\_in\\_guest](#)
*Executed commands inside a docker container.*
  - def [guest\\_status](#)
*Returns the status of a docker container.*

### 3.5.1 Detailed Description

Hypervisor driver for [Docker](#).

This class provides methods for managing docker containers.

### 3.5.2 Constructor & Destructor Documentation

#### 3.5.2.1 def hypervisor.docker\_driver.Docker.\_\_init\_\_( self )

Instantiates a [Docker](#) object.

Args: None.

Returns: None.

Note: This method initializes a set of values for configuring [Docker](#) remote API client. `__port` is the port number used for report API invocation. `__version` is the version number for the report API. `__dns_list` is the list of DNS server(s) used by each container.

### 3.5.3 Member Function Documentation

#### 3.5.3.1 def hypervisor.docker\_driver.Docker.deploy( self, host, user, image\_name, vnf\_name )

Deploys a docker container.

Args: host: IP address or hostname of the machine where the docker container is to be deployed user: name of the user who owns the VNF image\_name: docker image name for the VNF vnf\_name: name of the VNF instance

Returns: If the operation is successful then returns a tuple consisting of the following values: container\_id: docker container id return\_code: SUCCESS return\_message: EMPTY in this case otherwise returns the error as the following tuple: None as the first value return\_code: one of the error codes defined in hypervisor\_return\_codes return\_message: detailed message for the return code

#### 3.5.3.2 def hypervisor.docker\_driver.Docker.get\_id( self, host, user, vnf\_name )

Returns a container's ID.

Args: host: IP address or hostname of the machine where the docker container is deployed user: name of the user who owns the VNF vnf\_type: type of the deployed VNF vnf\_name: name of the VNF instance whose ID is being queried

Returns: [Docker](#) container ID.

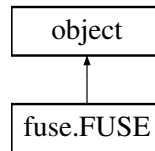
The documentation for this class was generated from the following file:

- /home/mfbari/works/WatNFV/nf.io/src/hypervisor/docker\_driver.py

### 3.6 fuse.FUSE Class Reference

This class is the lower level interface and should not be subclassed under normal use.

Inheritance diagram for fuse.FUSE:



#### Public Member Functions

- def [\\_\\_init\\_\\_](#)  
*Setting raw\_fi to True will cause [FUSE](#) to pass the [fuse\\_file\\_info](#) class as is to [Operations](#), instead of just the fh field.*
- def **getattr**
- def **readlink**
- def **mknod**
- def **mkdir**
- def **unlink**
- def **rmdir**
- def **symlink**
- def **rename**
- def **link**
- def **chmod**
- def **chown**
- def **truncate**
- def **open**
- def **read**
- def **write**
- def **statfs**
- def **flush**
- def **release**
- def **fsync**
- def **setxattr**
- def **getxattr**
- def **listxattr**
- def **removexattr**
- def **opendir**
- def **readdir**
- def **releasedir**
- def **fsyncdir**
- def **init**
- def **destroy**
- def **access**
- def **create**
- def **ftruncate**
- def **fgetattr**
- def **lock**
- def **utimens**
- def **bmap**

## Public Attributes

- **operations**
- **raw\_fi**
- **encoding**

## Static Public Attributes

- tuple **OPTIONS**

### 3.6.1 Detailed Description

This class is the lower level interface and should not be subclassed under normal use. Its methods are called by fuse.

Assumes API version 2.6 or later.

### 3.6.2 Constructor & Destructor Documentation

**3.6.2.1** `def fuse.FUSE.__init__( self, operations, mountpoint, raw_fi=False, encoding='utf-8', kwargs )`

Setting raw\_fi to True will cause FUSE to pass the `fuse_file_info` class as is to `Operations`, instead of just the fh field. This gives you access to `direct_io`, `keep_cache`, etc.

### 3.6.3 Member Data Documentation

**3.6.3.1** `tuple fuse.FUSE.OPTIONS` [static]

**Initial value:**

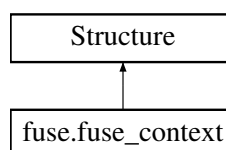
```
1 = (
2     ('foreground', '-f'),
3     ('debug', '-d'),
4     ('nothreads', '-s'),
5 )
```

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

- `/home/mfbari/works/WatNFV/nf.io/src/fuse.py`

## 3.7 fuse.fuse\_context Class Reference

Inheritance diagram for `fuse.fuse_context`:

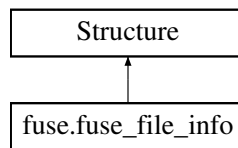


The documentation for this class was generated from the following file:

- `/home/mfbari/works/WatNFV/nf.io/src/fuse.py`

### 3.8 fuse.fuse\_file\_info Class Reference

Inheritance diagram for fuse.fuse\_file\_info:

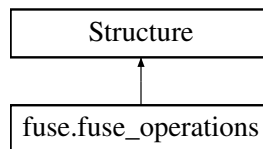


The documentation for this class was generated from the following file:

- /home/mfbari/works/WatNFV/nf.io/src/fuse.py

### 3.9 fuse.fuse\_operations Class Reference

Inheritance diagram for fuse.fuse\_operations:

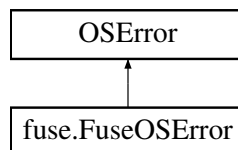


The documentation for this class was generated from the following file:

- /home/mfbari/works/WatNFV/nf.io/src/fuse.py

### 3.10 fuse.FuseOSError Class Reference

Inheritance diagram for fuse.FuseOSError:



#### Public Member Functions

- def `__init__`

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

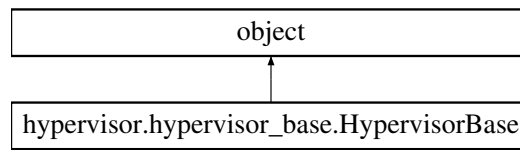
- /home/mfbari/works/WatNFV/nf.io/src/fuse.py

### 3.11 hypervisor.hypervisor\_base.HypervisorBase Class Reference

Base class for hypervisors.



Inheritance diagram for hypervisor.hypervisor\_base.HypervisorBase:



## Public Member Functions

- def [get\\_id](#)  
*Returns the hypervisor specific ID of the VM or container.*
- def [deploy](#)  
*Deploys a VM or container.*
- def [pause](#)  
*Pauses a VM or container.*
- def [destroy](#)  
*Destroys a VM or container.*
- def [execute\\_in\\_guest](#)  
*Executes a command in the VM or container.*
- def [guest\\_status](#)  
*Returns the current status of a VM or container.*

### 3.11.1 Detailed Description

Base class for hypervisors.

This class must be extended by a hypervisor driver.

### 3.11.2 Member Function Documentation

#### 3.11.2.1 def hypervisor.hypervisor\_base.HypervisorBase.deploy ( self )

Deploys a VM or container.

Args: Defined in derived class.

Returns: Hypervisor specific return code.

#### 3.11.2.2 def hypervisor.hypervisor\_base.HypervisorBase.destroy ( self )

Destroys a VM or container.

Args: Defined in derived class.

Returns: Hypervisor specific return code.

#### 3.11.2.3 def hypervisor.hypervisor\_base.HypervisorBase.execute\_in\_guest ( self )

Executes a command in the VM or container.

Args: Defined in derived class.

Returns: Hypervisor specific return code.

#### 3.11.2.4 `def hypervisor.hypervisor_base.HypervisorBase.get_id ( self )`

Returns the hypervisor specific ID of the VM or container.

Args: Defined in derived class.

Returns: Hypervisor specific ID for a VM or container.

#### 3.11.2.5 `def hypervisor.hypervisor_base.HypervisorBase.guest_status ( self )`

Returns the current status of a VM or container.

Args: Defined in derived class.

Returns: Current status of a VM or container.

#### 3.11.2.6 `def hypervisor.hypervisor_base.HypervisorBase.pause ( self )`

Pauses a VM or container.

Args: Defined in derived class.

Returns: Hypervisor specific return code.

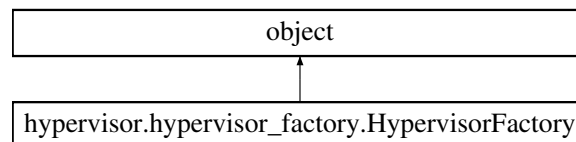
The documentation for this class was generated from the following file:

- `/home/mfbari/works/WatNFV/nf.io/src/hypervisor/hypervisor_base.py`

## 3.12 `hypervisor.hypervisor_factory.HypervisorFactory` Class Reference

A singleton class for creating hypervisor driver objects.

Inheritance diagram for `hypervisor.hypervisor_factory.HypervisorFactory`:



### Public Member Functions

- `def \_\_init\_\_`  
*Instantiates a [HypervisorFactory](#) object.*

### Static Public Member Functions

- `def get\_hypervisor\_instance`  
*Returns the hypervisor driver instance.*

#### 3.12.1 Detailed Description

A singleton class for creating hypervisor driver objects.

For an instantiation of `nf.io` there can be exactly one object of only one type of hypervisor. `HypervisorFactory` takes care of the creation logic.

### 3.12.2 Constructor & Destructor Documentation

3.12.2.1 `def hypervisor.hypervisor_factory.HypervisorFactory.__init__( self, hypervisor_type = "Docker" )`

Instantiates a [HypervisorFactory](#) object.

Args: `hypervisor_type`: The type of hypervisor object to instantiate. Valid hypervisor types are 'Docker' and 'Libvirt' for the time being.

Returns: Nothing. Initializaes the factory object.

Note: If this factory class is instantiated multiple times with different types of `hypervisor_type` argument then it raises a `ValueError`.

If this factory class is instantiated with a hypervisor type other than Docker or Libvirt it raises a `TypeError`.

### 3.12.3 Member Function Documentation

3.12.3.1 `def hypervisor.hypervisor_factory.HypervisorFactory.get_hypervisor_instance( ) [static]`

Returns the hypervisor driver nstance.

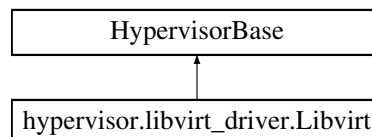
If the instance is not initialized then a `RuntimeError` is raised.

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

- `/home/mfbari/works/WatNFV/nf.io/src/hypervisor/hypervisor_factory.py`

## 3.13 hypervisor.libvirt\_driver.Libvirt Class Reference

Inheritance diagram for `hypervisor.libvirt_driver.Libvirt`:



### Public Member Functions

- `def deploy`
- `def pause`
- `def destroy`

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

- `/home/mfbari/works/WatNFV/nf.io/src/hypervisor/libvirt_driver.py`

## 3.14 fuse.LoggingMixIn Class Reference

### Public Member Functions

- `def __call__`

## Static Public Attributes

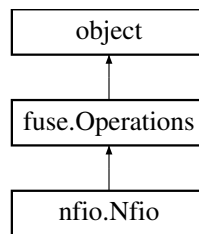
- tuple **log** = logging.getLogger('fuse.log-mixin')

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

- /home/mfbari/works/WatNFV/nf.io/src/fuse.py

## 3.15 nfio.Nfio Class Reference

Inheritance diagram for nfio.Nfio:



## Public Member Functions

- def [\\_\\_init\\_\\_](#)  
*Instantiates a [Nfio](#) object.*
- def **access**
- def **chmod**
- def **chown**
- def [getattr](#)  
*Returns the file attributes of the file specified by path Args: path: Path of the file fh: Open file handle to the file  
Returns: A dictionary containing file attributes.*
- def **readdir**
- def **readlink**
- def **mknod**
- def **rmdir**
- def [mkdir](#)  
*The semantics have been redefined to create a new VNF instance when a directory is created under a specific type of VNF directory.*
- def **statfs**
- def **unlink**
- def **symlink**
- def **rename**
- def **link**
- def **utimens**
- def **open**
- def **create**
- def [read](#)  
*Reads an open file.*
- def [write](#)  
*Write to an open file.*
- def **truncate**
- def **flush**
- def **release**
- def **fsync**

## Public Attributes

- **root**
- **mountpoint**
- **hypervisor**
- **vnfs\_ops**
- **module\_root**

## Additional Inherited Members

### 3.15.1 Constructor & Destructor Documentation

3.15.1.1 `def nfio.Nfio.__init__( self, root, mountpoint, hypervisor = ' Docker' , module_root = ' middleboxes' )`

Instantiates a [Nfio](#) object.

Args: root: The root directory of nfio file system. The root directory stores persistent state about the system. mountpoint: The mountpoint of nfio file system. The mountpoint is required to intercept the file system calls via fuse. All the file system calls for fuse mounted files/directories are intercepted by libfuse and our provided implementation is executed. hypervisor: The type of hypervisor to use for deploying VNFs. The default is to use Docker containers. However, we also plan to add support for Libvirt. module\_root: Root directory of the middlebox modules. Each middlebox provides it's own implementation of certain system calls in a separate module. module\_root points to the root of that module. If nothing is provided a default of 'middleboxes' will be assumed. Returns: Nothing. Mounts nf.io file system at the specified mountpoint and creates a loop to act upon different file system calls.

### 3.15.2 Member Function Documentation

3.15.2.1 `def nfio.Nfio.getattr( self, path, fh = None )`

Returns the file attributes of the file specified by path Args: path: Path of the file fh: Open file handle to the file Returns: A dictionary containing file attributes.

The dictionary contains the following keys: st\_atime: Last access time st\_ctime: File creation time st\_gid: Group id of the owner group st\_mode: File access mode st\_mtime: Last modification time st\_nlink: Number of symbolic links to the file st\_size: Size of the file in bytes st\_uid: User id of the file owner Note: For special placeholder files for VNFs, st\_size is set to a constant 1000. This is to make sure read utilities such as cat work for these special placeholder files.

3.15.2.2 `def nfio.Nfio.mkdir( self, path, mode )`

The semantics have been redefined to create a new VNF instance when a directory is created under a specific type of VNF directory.

Args: path: path of the directory to create. The path also represents the name of the new VNF instance to be created. mode: File access mode for the new directory. Returns: If path does not correspond to a directory under a specific VNF type directory then errno.EPERM is returned. Otherwise the return code is same as os.mkdir()'s return code.

3.15.2.3 `def nfio.Nfio.read( self, path, length, offset, fh )`

Reads an open file.

This nfio specific implementation parses path to see if the read is from any VNF or not. In case the read is from a VNF, the corresponding VNF module is loaded and the module's `_read` function is invoked to complete the read system call.

Args: path: path represents the path of the file to read from length: number of bytes to read from the file offset: byte offset indicating the starting byte to read from fh: file descriptor of the open file represented by path

Returns: length bytes from offset byte of the file represented by fh and path

Notes: VNFs can have special files which are placeholders for statistics such as number of received/sent bytes etc. VNFs provide their own implementation of read and handle reading of these special placeholder files.

#### 3.15.2.4 `def nfio.Nfio.write( self, path, buf, offset, fh )`

Write to an open file.

In this nfio specific implementation the path is parsed to see if the write is for any specific VNF or not. If the write is for any file under a VNF directory then the corresponding VNF module is loaded and the module's `_write` function is invoked.

Args: path: path to the file to write buf: the data to write offset: the byte offset at which the write should begin fh: file descriptor of the open file represented by path

Returns: Returns the number of bytes written to the file starting at offset

Note: VNFs can have special files where writing specific strings trigger a specific function. For example, writing 'activate' to the 'action' file of a VNF will start the VNF. VNF specific modules handle such special cases of writing.

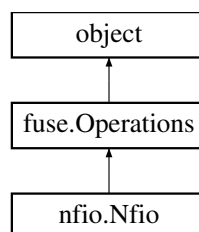
The documentation for this class was generated from the following file:

- `/home/mfbari/works/WatNFV/nf.io/src/nfio.py`

## 3.16 fuse.Operations Class Reference

This class should be subclassed and passed as an argument to [FUSE](#) on initialization.

Inheritance diagram for fuse.Operations:



### Public Member Functions

- `def __call__`
- `def access`
- `def chmod`
- `def chown`
- `def create`

*When raw\_fi is False (default case), fi is None and create should return a numerical file handle.*

- `def destroy`
- `def flush`
- `def fsync`
- `def fsyncdir`
- `def getattr`

*Returns a dictionary with keys identical to the stat C structure of stat(2).*

- `def getxattr`

- def **init**  
*Called on filesystem initialization.*
- def **link**
- def **listxattr**
- def **mkdir**
- def **mknod**
- def **open**  
*When raw\_fi is False (default case), open should return a numerical file handle.*
- def **opendir**
- def **read**
- def **readdir**  
*Can return either a list of names, or a list of (name, attrs, offset) tuples.*
- def **readlink**
- def **release**
- def **releasedir**
- def **removexattr**
- def **rename**
- def **rmdir**
- def **setxattr**
- def **statfs**  
*Returns a dictionary with keys identical to the statvfs C structure of statvfs(3).*
- def **symlink**
- def **truncate**
- def **unlink**
- def **utimens**
- def **write**

### Static Public Attributes

- **bmap** = None
- **lock** = None

### 3.16.1 Detailed Description

This class should be subclassed and passed as an argument to [FUSE](#) on initialization.

All operations should raise a [FuseOSError](#) exception on error.

When in doubt of what an operation should do, check the [FUSE](#) header file or the corresponding system call man page.

### 3.16.2 Member Function Documentation

#### 3.16.2.1 def fuse.Operations.create ( self, path, mode, fi=None )

When raw\_fi is False (default case), fi is None and create should return a numerical file handle.

When raw\_fi is True the file handle should be set directly by create and return 0.

#### 3.16.2.2 def fuse.Operations.getattr ( self, path, fh=None )

Returns a dictionary with keys identical to the stat C structure of stat(2).

st\_atime, st\_mtime and st\_ctime should be floats.

NOTE: There is an incompatibility between Linux and Mac OS X concerning st\_nlink of directories. Mac OS X counts all files inside the directory, while Linux counts only the subdirectories.

### 3.16.2.3 `def fuse.Operations.init ( self, path )`

Called on filesystem initialization.

(Path is always /)

Use it instead of `__init__` if you start threads on initialization.

### 3.16.2.4 `def fuse.Operations.open ( self, path, flags )`

When `raw_fi` is False (default case), `open` should return a numerical file handle.

When `raw_fi` is True the signature of `open` becomes: `open(self, path, fi)`

and the file handle should be set directly.

### 3.16.2.5 `def fuse.Operations.readdir ( self, path, fh )`

Can return either a list of names, or a list of (name, attrs, offset) tuples.

attrs is a dict as in `getattr`.

### 3.16.2.6 `def fuse.Operations.statfs ( self, path )`

Returns a dictionary with keys identical to the `statvfs` C structure of `statvfs(3)`.

On Mac OS X `f_bsize` and `f_frsize` must be a power of 2 (minimum 512).

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

- `/home/mfbari/works/WatNFV/nf.io/src/fuse.py`

## 3.17 `vnfs_operations.VNFSOperations` Class Reference

Provides a common set of operations for `nfio`.

### Public Member Functions

- `def __init__`
- `def vnfs_create_vnf_instance`  
*Create the file system structure for a VNF.*
- `def vnfs_get_opcode`  
*Determinse the type of operation based on the path.*
- `def vnfs_get_nf_type`  
*Parse the type of VNF from path.*
- `def vnfs_get_file_name`  
*Return the name of the file represented by a path.*
- `def vnfs_is_nf_instance`  
*Determines if a path represents an nf instance directory.*
- `def vnfs_get_instance_configuration`  
*Return the configuration parameters related to a VNF instance.*
- `def vnfs_deploy_nf`  
*Deploys a VNF instance.*
- `def vnfs_stop_vnf`



*Stops a VNF instance.*

- def [vnfs\\_get\\_rx\\_bytes](#)

*Reads the number of bytes received by a VNF instance.*

- def [vnfs\\_get\\_tx\\_bytes](#)

*Reads the number of bytes sent by a VNF instance.*

- def [vnfs\\_get\\_pkt\\_drops](#)

*Reads the number of packets dropped by a VNF instance.*

- def [vnfs\\_get\\_status](#)

*Get the status of a VNF instance, e.g., the VNF is running/suspended/stopped etc.*

## Public Attributes

- **vnfs\_root**

## Static Public Attributes

- int **OP\_UNDEFINED** = 0xFF
- int **OP\_NF** = 0x01

### 3.17.1 Detailed Description

Provides a common set of operations for nfio.

These operations act as a helper.

### 3.17.2 Member Function Documentation

#### 3.17.2.1 def vnfs\_operations.VNFSOperations.vnfs\_create\_vnf\_instance ( self, path, mode )

Create the file system structure for a VNF.

Args: path: path of the new VNF instance. mode: file creation mode for the new VNF instance directory.

Returns: returns the return code of os.mkdir

#### 3.17.2.2 def vnfs\_operations.VNFSOperations.vnfs\_deploy\_nf ( self, nf\_path )

Deploys a VNF instance.

Args: nf\_path: path of the VNF instance.

Returns: return codes are described in hypervisor.hypervisor\_return\_codes module.

#### 3.17.2.3 def vnfs\_operations.VNFSOperations.vnfs\_get\_file\_name ( self, path )

Return the name of the file represented by a path.

Args: path: the path of the file in concern

Returns: returns the name of the file, i.e., last token after / in the path.

#### 3.17.2.4 `def vnfs_operations.VNFSOperations.vnfs_get_instance_configuration ( self, nf_path )`

Return the configuration parameters related to a VNF instance.

Args: `nf_path`: path of the VNF instance. e.g., `/mnt/vnfsmnt/firewall/fw-alpha`

Returns: A tuple representing the configuration of the VNF instance. The tuple is organized in the following order:  
`nf_instance_name`: name of the VNF instance. `nf_type`: type of the VNF. `ip_address`: IP address of the machine where this VNF will be deployed. `image_name`: name of the VM/container image for that VNF.

#### 3.17.2.5 `def vnfs_operations.VNFSOperations.vnfs_get_nf_type ( self, path )`

Parse the type of VNF from path.

Args: `path`: the path of the file/directory on which some operation is being performed.

Returns: Returns the type of VNF parsed from the path, e.g., if the path is `/mnt/vnfsroot/nf-types/firewall/fw-alpha/action` then returns `firewall`.

#### 3.17.2.6 `def vnfs_operations.VNFSOperations.vnfs_get_opcode ( self, path )`

Determine the type of operation based on the path.

Args: `path`: path to the file/directory on which the operation is being performed

Returns: If the file is under `nf-types` subdirectory in the `nfio` mount, then returns `OP_NF`. Otherwise, returns `OP_UNDEFINED`.

#### 3.17.2.7 `def vnfs_operations.VNFSOperations.vnfs_get_pkt_drops ( self, nf_path )`

Reads the number of packets dropped by a VNF instance.

Args: `nf_path`: path of the VNF instance.

Returns: returns the number of packets dropped by a VNF instance.

#### 3.17.2.8 `def vnfs_operations.VNFSOperations.vnfs_get_rx_bytes ( self, nf_path )`

Reads the number of bytes received by a VNF instance.

Args: `nf_path`: path of the VNF instance.

Returns: returns the number of bytes received by a VNF instance.

#### 3.17.2.9 `def vnfs_operations.VNFSOperations.vnfs_get_status ( self, nf_path )`

Get the status of a VNF instance, e.g., the VNF is running/suspended/stopped etc.

Args: `nf_path`: path of the VNF instance.

Returns: Hypervisor specific status of the VNF. For example, if Docker is being used for VNF deployment then Docker specific container status message is returned.

#### 3.17.2.10 `def vnfs_operations.VNFSOperations.vnfs_get_tx_bytes ( self, nf_path )`

Reads the number of bytes sent by a VNF instance.

Args: `nf_path`: path of the VNF instance.

Returns: returns the number of bytes sent by a VNF instance.

**3.17.2.11** `def vnfs_operations.VNFSOperations.vnfs_is_nf_instance ( self, path )`

Determines if a path represents an nf instance directory.

Args: path: path of the file/directory in concern.

Returns: True: if path represents an nf instance directory. For example, if path is /mnt/vnfsmnt/nf-types/firewall/fw-alpha then returns True.

False: if the path does not represent an nf instance directory. For example, if path is /mnt/vnfsmnt/nf-types/firewall/fw-alpha/action then returns False.

**3.17.2.12** `def vnfs_operations.VNFSOperations.vnfs_stop_vnf ( self, nf_path )`

Stops a VNF instance.

Args: nf\_path: path of the VNF instance.

Returns: return codes are described in `hypervisor.hypervisor_return_codes` module.

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

- /home/mfbari/works/WatNFV/nf.io/src/vnfs\_operations.py

# Index

- `__init__`
  - `fuse::FUSE`, 9
  - `hypervisor::docker_driver::Docker`, 7
  - `hypervisor::hypervisor_factory::HypervisorFactory`, 13
  - `nfio::Nfio`, 15
- `create`
  - `fuse::Operations`, 17
- `deploy`
  - `hypervisor::docker_driver::Docker`, 7
  - `hypervisor::hypervisor_base::HypervisorBase`, 11
- `destroy`
  - `hypervisor::hypervisor_base::HypervisorBase`, 11
- `execute_in_guest`
  - `hypervisor::hypervisor_base::HypervisorBase`, 11
- `fuse.c_stat`, 5
- `fuse.c_statvfs`, 5
- `fuse.c_timespec`, 5
- `fuse.c_utimbuf`, 6
- `fuse.FUSE`, 8
- `fuse.fuse_context`, 9
- `fuse.fuse_file_info`, 10
- `fuse.fuse_operations`, 10
- `fuse.FuseOSError`, 10
- `fuse.LoggingMixIn`, 13
- `fuse.Operations`, 16
- `fuse::FUSE`
  - `__init__`, 9
  - `OPTIONS`, 9
- `fuse::Operations`
  - `create`, 17
  - `getattr`, 17
  - `init`, 17
  - `open`, 18
  - `readdir`, 18
  - `statfs`, 18
- `get_hypervisor_instance`
  - `hypervisor::hypervisor_factory::HypervisorFactory`, 13
- `get_id`
  - `hypervisor::docker_driver::Docker`, 7
  - `hypervisor::hypervisor_base::HypervisorBase`, 11
- `getattr`
  - `fuse::Operations`, 17
  - `nfio::Nfio`, 15
- `guest_status`
  - `hypervisor::hypervisor_base::HypervisorBase`, 12
- `hypervisor.docker_driver.Docker`, 6
- `hypervisor.hypervisor_base.HypervisorBase`, 10
- `hypervisor.hypervisor_factory.HypervisorFactory`, 12
- `hypervisor.libvirt_driver.Libvirt`, 13
- `hypervisor::docker_driver::Docker`
  - `__init__`, 7
  - `deploy`, 7
  - `get_id`, 7
- `hypervisor::hypervisor_base::HypervisorBase`
  - `deploy`, 11
  - `destroy`, 11
  - `execute_in_guest`, 11
  - `get_id`, 11
  - `guest_status`, 12
  - `pause`, 12
- `hypervisor::hypervisor_factory::HypervisorFactory`
  - `__init__`, 13
  - `get_hypervisor_instance`, 13
- `init`
  - `fuse::Operations`, 17
- `mkdir`
  - `nfio::Nfio`, 15
- `nfio.Nfio`, 14
- `nfio::Nfio`
  - `__init__`, 15
  - `getattr`, 15
  - `mkdir`, 15
  - `read`, 15
  - `write`, 16
- `OPTIONS`
  - `fuse::FUSE`, 9
- `open`
  - `fuse::Operations`, 18
- `pause`
  - `hypervisor::hypervisor_base::HypervisorBase`, 12
- `read`
  - `nfio::Nfio`, 15
- `readdir`
  - `fuse::Operations`, 18
- `statfs`
  - `fuse::Operations`, 18
- `vnfs_create_vnf_instance`

- vnfs\_operations::VNFSOperations, [19](#)
- vnfs\_deploy\_nf
  - vnfs\_operations::VNFSOperations, [19](#)
- vnfs\_get\_file\_name
  - vnfs\_operations::VNFSOperations, [19](#)
- vnfs\_get\_instance\_configuration
  - vnfs\_operations::VNFSOperations, [19](#)
- vnfs\_get\_nf\_type
  - vnfs\_operations::VNFSOperations, [20](#)
- vnfs\_get\_opcode
  - vnfs\_operations::VNFSOperations, [20](#)
- vnfs\_get\_pkt\_drops
  - vnfs\_operations::VNFSOperations, [20](#)
- vnfs\_get\_rx\_bytes
  - vnfs\_operations::VNFSOperations, [20](#)
- vnfs\_get\_status
  - vnfs\_operations::VNFSOperations, [20](#)
- vnfs\_get\_tx\_bytes
  - vnfs\_operations::VNFSOperations, [20](#)
- vnfs\_is\_nf\_instance
  - vnfs\_operations::VNFSOperations, [20](#)
- vnfs\_operations.VNFSOperations, [18](#)
- vnfs\_operations::VNFSOperations
  - vnfs\_create\_vnf\_instance, [19](#)
  - vnfs\_deploy\_nf, [19](#)
  - vnfs\_get\_file\_name, [19](#)
  - vnfs\_get\_instance\_configuration, [19](#)
  - vnfs\_get\_nf\_type, [20](#)
  - vnfs\_get\_opcode, [20](#)
  - vnfs\_get\_pkt\_drops, [20](#)
  - vnfs\_get\_rx\_bytes, [20](#)
  - vnfs\_get\_status, [20](#)
  - vnfs\_get\_tx\_bytes, [20](#)
  - vnfs\_is\_nf\_instance, [20](#)
  - vnfs\_stop\_vnf, [21](#)
- vnfs\_stop\_vnf
  - vnfs\_operations::VNFSOperations, [21](#)
- write
  - nfio::Nfio, [16](#)