# FutureSynchronyzer /doc

Generated by Doxygen 1.8.9.1

Thu Feb 26 2015 23:34:59

# **Contents**

1	Nam	nespace	Index															1
	1.1	Names	space List								 	 	 	 		 		1
2	Hier	archica	l Index															3
	2.1	Class	Hierarchy								 	 	 	 		 		3
3	Clas	ss Index																5
	3.1	Class	List							٠.	 	 	 	 		 		5
4	File	Index																7
	4.1	File Lis	st								 	 	 	 		 		7
5	Nam	nespace	Docume	ntatio	on													9
	5.1	factory	Namespa	ace R	eferen	ice .					 	 	 	 		 		9
	5.2	policy	Namespac	ce Re	ferenc	е					 	 	 	 		 		9
		5.2.1	Typedef	Docu	menta	ıtion .					 	 	 	 		 		9
			5.2.1.1	Nol	Resultl	Policy	/				 	 	 	 		 		9
	5.3	thread	Namespa	ice Re	eferen	ce .					 	 	 	 		 		9
6	Clas	ss Docu	mentation	n														11
	6.1	policy:	:AddNumb	erRe	sultPo	olicy C	Class	Refe	rence		 	 	 	 		 		11
		6.1.1	Detailed	Desc	ription	١					 	 	 	 		 		11
		6.1.2	Construc	ctor &	Destr	uctor	Docu	umen	tatior	١	 	 	 	 		 		11
			6.1.2.1	Add	dNumb	erRe	sultP	olicy			 	 	 	 		 		11
			6.1.2.2	$\sim$ A	ddNun	nberF	Resul	tPolic	у		 	 	 	 		 		11
		6.1.3	Member	Fund	tion D	ocum	nentat	tion			 	 	 	 		 		12
			6.1.3.1	app	olyResi	ult .					 	 	 	 		 		12
	6.2	factory	::AsyncFu	ıtureF	- actory	/ Clas	s Re	feren	ce .		 	 	 	 		 		12
		6.2.1	Detailed	Desc	ription	١					 	 	 	 		 		12
		6.2.2	Construc	ctor &	Destr	uctor	Docu	umen	tatior	١	 	 	 	 		 		12
			6.2.2.1	Asy	/ncFuti	ureFa	actory	<i>'</i>			 	 	 	 		 		12
			6.2.2.2	$\sim$ A	syncF	uture	Facto	ory .			 	 	 	 		 		12
		6.2.3	Member	Fund	tion D	ocum	nentat	tion			 	 	 	 				12

iv CONTENTS

			6.2.3.1	createFuture		12
	6.3	Future	Factory Cla	lass Reference		13
	6.4	thread:	:FutureSy	${\it vnchronyzer}{<}{\sf ResultPolicy},{\sf FutureFactory},{\sf OutputResult}{>}{\sf Class}{\sf Template}{\sf Reference}$	rence	14
		6.4.1	Detailed	Description		16
		6.4.2	Member	Typedef Documentation		16
			6.4.2.1	Function		16
			6.4.2.2	FunctionPtr		16
		6.4.3	Construc	ctor & Destructor Documentation		17
			6.4.3.1	FutureSynchronyzer		17
			6.4.3.2	$\sim$ FutureSynchronyzer		18
		6.4.4	Member	Function Documentation		18
			6.4.4.1	addFunction		18
			6.4.4.2	applyResult		18
			6.4.4.3	applyResult		18
			6.4.4.4	getNextFunction		19
			6.4.4.5	launch		19
			6.4.4.6	launchAndWait		20
			6.4.4.7	numThreads		20
			6.4.4.8	outputResult		20
			6.4.4.9	wait		20
		6.4.5	Member	Data Documentation		21
			6.4.5.1	m_functions		21
			6.4.5.2	m_futures		21
			6.4.5.3	m_numThreads		21
			6.4.5.4	m_outputResult		21
	6.5	policy::	NoPolicy (	Class Reference		21
		6.5.1	Detailed	Description		22
	6.6	Result	Policy Clas	ss Reference		23
7	File I	Docume	entation			25
•	7.1			x File Reference		25
	7.2			yzer.hxx File Reference		26
	7.3			Reference		27
		7.3.1		refinition Documentation		27
		7.0.1	7.3.1.1	UNUSED		27
	7.4	Policy h	_	eference		27
		•		File Reference		28

# Namespace Index

4 4						
1.1	NI4	ame	Cha	$\sim$	10	•1
	11/1/	111111€	2014			٩I
	140	4111	Opu	~~		~ •

Here is a list of all namespaces with brief descriptions:

factory						 																		
policy						 													 					9
thread						 													 					9

2 Namespace Index

# **Hierarchical Index**

## 2.1 Class Hierarchy

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

policy::AddNumberResultPolicy	11
factory::AsyncFutureFactory	12
FutureFactory	13
thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >	14
policy::NoPolicy	21
ResultPolicy	23
$thread:: Future Synchronyzer < Result Policy, \ Future Factory, \ Output Result > \dots $	14
OutputResult *	
size_t	
vector < FunctionPtr >	
vector< std::future< OutputResult >>	

**Hierarchical Index** 

# **Class Index**

## 3.1 Class List

Here are the classes	, structs,	unions	and inte	erfaces	with	brief	descri	ptions
----------------------	------------	--------	----------	---------	------	-------	--------	--------

policy::AddNumberResultPolicy	1
factory::AsyncFutureFactory	'
FutureFactory	'
thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >	
policy::NoPolicy	
ResultPolicy	2

6 Class Index

# File Index

## 4.1 File List

Here is a list of all files with brief descriptions:

FutureFactory.hxx	25
FutureSynchronyzer.hxx	26
Macro.hpp	27
Policy.hxx	27
ResultPolicy.hxx	28

8 File Index

# **Namespace Documentation**

## 5.1 factory Namespace Reference

#### Classes

class AsyncFutureFactory

## 5.2 policy Namespace Reference

#### Classes

- class AddNumberResultPolicy
- class NoPolicy

### **Typedefs**

• typedef NoPolicy NoResultPolicy

#### 5.2.1 Typedef Documentation

5.2.1.1 typedef NoPolicy policy::NoResultPolicy

The class does nothing with the result passed to it's arguments

## 5.3 thread Namespace Reference

#### Classes

• class FutureSynchronyzer

Names	pace	Docur	ment	ation

## **Class Documentation**

## 6.1 policy::AddNumberResultPolicy Class Reference

```
#include <ResultPolicy.hxx>
```

Collaboration diagram for policy::AddNumberResultPolicy:

### policy::AddNumberResultPolicy

- + AddNumberResultPolicy()
- + ~AddNumberResultPolicy()
- + applyResult()

#### **Public Member Functions**

- AddNumberResultPolicy ()
- virtual ~AddNumberResultPolicy ()
- template<typename OutputResult >
   void applyResult (OutputResult \*out, OutputResult \*otherValue)

#### 6.1.1 Detailed Description

This class provides a policy to add a value to another one.

#### 6.1.2 Constructor & Destructor Documentation

- **6.1.2.1** policy::AddNumberResultPolicy::AddNumberResultPolicy( ) [inline]
- **6.1.2.2** virtual policy::AddNumberResultPolicy::~AddNumberResultPolicy( ) [inline], [virtual]

12 Class Documentation

#### 6.1.3 Member Function Documentation

6.1.3.1 template<typename OutputResult > void policy::AddNumberResultPolicy::applyResult ( OutputResult \* out, OutputResult \* otherValue ) [inline]

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

· ResultPolicy.hxx

### 6.2 factory::AsyncFutureFactory Class Reference

```
#include <FutureFactory.hxx>
```

Collaboration diagram for factory::AsyncFutureFactory:

### factory::AsyncFutureFactory

- + AsyncFutureFactory()
- + ~AsyncFutureFactory()
- + createFuture()

#### **Public Member Functions**

- AsyncFutureFactory ()
- virtual ~AsyncFutureFactory ()
- template < typename OutputResult >
   std::future < OutputResult > createFuture (const std::function < OutputResult() > &f)

#### 6.2.1 Detailed Description

This class provides a factory to build future with std::async methods.

#### 6.2.2 Constructor & Destructor Documentation

- **6.2.2.1** factory::AsyncFutureFactory::AsyncFutureFactory() [inline]
- **6.2.2.2 virtual factory::**AsyncFutureFactory::~AsyncFutureFactory( ) [inline],[virtual]

#### 6.2.3 Member Function Documentation

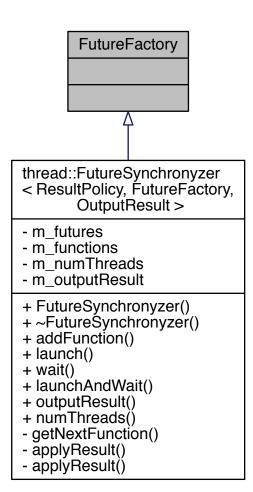
6.2.3.1 template<typename OutputResult > std::future<OutputResult> factory::AsyncFutureFactory::createFuture ( const std::function<OutputResult()> & f ) [inline]

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

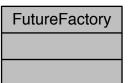
• FutureFactory.hxx

### 6.3 FutureFactory Class Reference

Inheritance diagram for FutureFactory:



Collaboration diagram for FutureFactory:



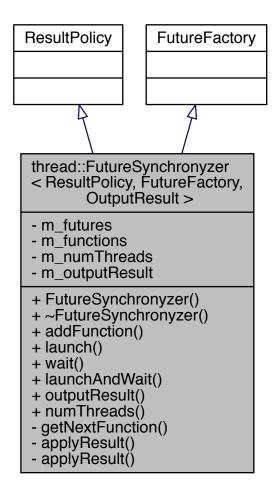
The documentation for this class was generated from the following file:

• FutureSynchronyzer.hxx

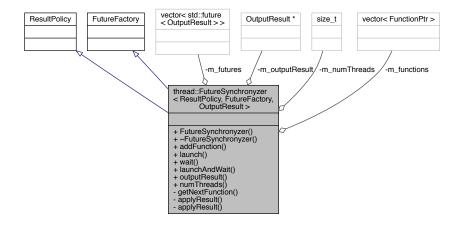
6.4 thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult > Class Template Reference

#include <FutureSynchronyzer.hxx>

Inheritance diagram for thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >:



 $Collaboration\ diagram\ for\ thread:: Future Synchronyzer < Result Policy,\ Future Factory,\ Output Result >: Policy,\ Ou$ 



16 Class Documentation

#### **Public Member Functions**

- FutureSynchronyzer (OutputResult \*outputResult=NULL, size\_t numThreads=std::thread::hardware\_←
   concurrency() > 0?std::thread::hardware\_concurrency():1)
- virtual ∼FutureSynchronyzer ()
- template<typename Fn , typename... Args>
   void addFunction (Fn &&function, Args &&...args)
- void launch ()
- void wait ()
- void launchAndWait ()
- OutputResult \* outputResult () const
- size t numThreads () const

#### **Private Types**

- typedef std::function< OutputResult()> Function
- typedef std::shared\_ptr< Function > FunctionPtr

#### **Private Member Functions**

- FunctionPtr getNextFunction ()
- template<typename Output = OutputResult, typename Policy = ResultPolicy>
   std::enable\_if<!(std::is\_same< Output, void >::value)&!(std::is\_same< Policy, policy::NoResultPolicy >
   ::value), void >::type applyResult (std::future< Output > &f)
- template<typename Output = OutputResult, typename Policy = ResultPolicy> std::enable\_if<(std::is\_same< Output, void >::value)||(std::is\_same< Policy, policy::NoResultPolicy > ::value), void >::type applyResult (std::future< Output > &f)

#### **Private Attributes**

- std::vector< std::future< OutputResult >> m futures
- std::vector< FunctionPtr > m functions
- size t m numThreads
- OutputResult \* m\_outputResult

#### 6.4.1 Detailed Description

 $template < class \ Result Policy = policy:: No Result Policy, \ class \ Future Factory = factory:: Async Future Factory, \ typename \ Output \leftarrow Result = void > class \ thread:: Future Synchronyzer < Result Policy, Future Factory, Output Result >$ 

This class permits to add functions and it's arguments and launch them with a limit number of threads defined at the onbject's construction. All the function added must returned the same type of argument.

#### 6.4.2 Member Typedef Documentation

- 6.4.2.1 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
  OutputResult = void> typedef std::function < OutputResult()> thread::FutureSynchronyzer < ResultPolicy,
  FutureFactory, OutputResult >::Function [private]
- 6.4.2.2 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
  OutputResult = void> typedef std::shared\_ptr < Function> thread::FutureSynchronyzer < ResultPolicy,
  FutureFactory, OutputResult >::FunctionPtr [private]

- 6.4.3 Constructor & Destructor Documentation
- 6.4.3.1 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename OutputResult = void> thread::FutureSynchronyzer < ResultPolicy, FutureFactory, OutputResult >::FutureSynchronyzer ( OutputResult \* outputResult = NULL, size\_t numThreads = std::thread::hardware\_concurrency(), 0?std::thread::hardware\_concurrency():1 )
  [inline]

Construct the Future Synchronizer.

18 Class Documentation

#### **Parameters**

outputResult	the return result (optional);
numThreads	the max number of threads. By default, the number of concurrent threads supported if de-
	tected or 1.

- 6.4.3.2 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
  OutputResult = void> virtual thread::FutureSynchronyzer < ResultPolicy, FutureFactory, OutputResult
  >::~FutureSynchronyzer() [inline], [virtual]
- 6.4.4 Member Function Documentation
- 6.4.4.1 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
  OutputResult = void> template < typename Fn , typename... Args> void thread::FutureSynchronyzer <
  ResultPolicy, FutureFactory, OutputResult >::addFunction ( Fn && function, Args &&... args ) [inline]

Add the function and its arguments to the list of function

#### **Parameters**

function	
args	all arguments

6.4.4.2 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename OutputResult = void> template < typename Output = OutputResult, typename Policy = ResultPolicy> std::enable\_if <!(std::is\_same < Output, void>::value) && !(std::is\_same < Policy, policy::NoResultPolicy>::value), void>::type thread::FutureSynchronyzer < ResultPolicy, FutureFactory, OutputResult >::applyResult ( std::future < Output > & f ) [inline], [private]

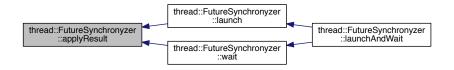
Launch the result operation. This method is used if OutputResult is not void and ResultPolicy is not ResultPolicy.

#### **Parameters**

f	the finished future with the result

Referenced by thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::launch(), and thread ::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::wait().

Here is the caller graph for this function:



Launch the result operation. This method is used if OutputResult is void or ResultPolicy is ResultPolicy.

**Parameters** 

f the finished future with the result

References UNUSED.

6.4.4.4 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
OutputResult = void > FunctionPtr thread::FutureSynchronyzer < ResultPolicy, FutureFactory,
OutputResult >::getNextFunction() [inline], [private]

#### Returns

the next function to be treated.

Referenced by thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::launch(). Here is the caller graph for this function:



6.4.4.5 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
OutputResult = void > void thread::FutureSynchronyzer < ResultPolicy, FutureFactory, OutputResult
>::launch() [inline]

#### Launch all functions

References thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::applyResult(), thread $\leftarrow$ ::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::getNextFunction(), and thread::Future  $\leftarrow$  Synchronyzer< ResultPolicy, FutureFactory, OutputResult >::m numThreads.

Referenced by thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::launchAndWait(). Here is the call graph for this function:

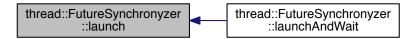
thread::FutureSynchronyzer ::applyResult

thread::FutureSynchronyzer ::applyResult

thread::FutureSynchronyzer ::getNextFunction

20 Class Documentation

Here is the caller graph for this function:



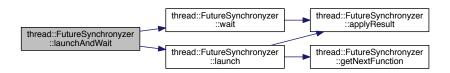
6.4.4.6 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
OutputResult = void > void thread::FutureSynchronyzer < ResultPolicy, FutureFactory, OutputResult
>::launchAndWait() [inline]

Launch all the functions and wait them to be finished.

References thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::launch(), and thread::

FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::wait().

Here is the call graph for this function:



6.4.4.7 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
OutputResult = void > size\_t thread::FutureSynchronyzer < ResultPolicy, FutureFactory, OutputResult
>::numThreads ( ) const [inline]

References thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::m\_numThreads.

6.4.4.8 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
OutputResult = void > OutputResult\* thread::FutureSynchronyzer < ResultPolicy, FutureFactory,
OutputResult >::outputResult ( ) const [inline]

References thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::m outputResult.

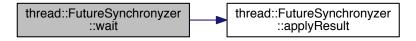
6.4.4.9 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
OutputResult = void > void thread::FutureSynchronyzer < ResultPolicy, FutureFactory, OutputResult
>::wait ( ) [inline]

Wait all functions to be finished.

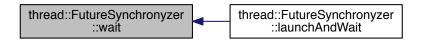
References thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::applyResult().

 $Referenced\ by\ thread::Future Synchronyzer < Result Policy,\ Future Factory,\ Output Result > :: launch And Wait().$ 

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.4.5 Member Data Documentation

- 6.4.5.1 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename OutputResult = void> std::vector<FunctionPtr> thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::m\_functions [private]
- 6.4.5.2 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
  OutputResult = void> std::vector<std::future<OutputResult>> thread::FutureSynchronyzer< ResultPolicy,
  FutureFactory, OutputResult>::m\_futures [private]
- 6.4.5.3 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
  OutputResult = void > size\_t thread::FutureSynchronyzer < ResultPolicy, FutureFactory, OutputResult
  >::m\_numThreads [private]

Referenced by thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::launch(), and thread ::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::numThreads().

6.4.5.4 template < class ResultPolicy = policy::NoResultPolicy, class FutureFactory = factory::AsyncFutureFactory, typename
OutputResult = void > OutputResult \* thread::FutureSynchronyzer < ResultPolicy, FutureFactory,
OutputResult >::m\_outputResult [private]

Referenced by thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::outputResult(). The documentation for this class was generated from the following file:

• FutureSynchronyzer.hxx

#### 6.5 policy::NoPolicy Class Reference

#include <Policy.hxx>

22 Class Documentation

Collaboration diagram for policy::NoPolicy:

policy::NoPolicy

### 6.5.1 Detailed Description

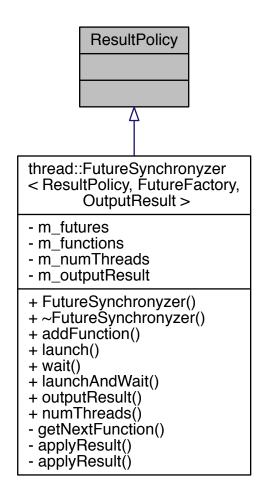
This class represents an empty policy

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

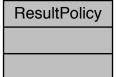
• Policy.hxx

### 6.6 ResultPolicy Class Reference

Inheritance diagram for ResultPolicy:



Collaboration diagram for ResultPolicy:



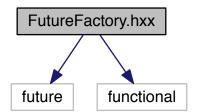
The documentation for this class was generated from the following file:

24 **Class Documentation** • FutureSynchronyzer.hxx

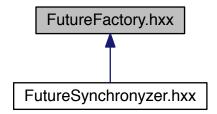
## **File Documentation**

## 7.1 FutureFactory.hxx File Reference

#include <future>
#include <functional>
Include dependency graph for FutureFactory.hxx:



This graph shows which files directly or indirectly include this file:



26 File Documentation

#### Classes

class factory::AsyncFutureFactory

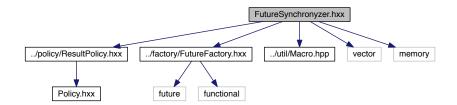
#### **Namespaces**

factory

## 7.2 FutureSynchronyzer.hxx File Reference

```
#include "../policy/ResultPolicy.hxx"
#include "../factory/FutureFactory.hxx"
#include "../util/Macro.hpp"
#include <vector>
#include <memory>
```

Include dependency graph for FutureSynchronyzer.hxx:



### Classes

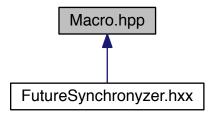
class thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >

#### **Namespaces**

thread

## 7.3 Macro.hpp File Reference

This graph shows which files directly or indirectly include this file:



#### **Macros**

#define UNUSED(x) (void)(x)

#### 7.3.1 Macro Definition Documentation

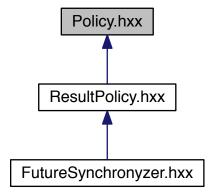
#### 7.3.1.1 #define UNUSED(x) (void)(x)

This macro avoid warning of unused parameter

Referenced by thread::FutureSynchronyzer< ResultPolicy, FutureFactory, OutputResult >::applyResult().

## 7.4 Policy.hxx File Reference

This graph shows which files directly or indirectly include this file:



28 File Documentation

### Classes

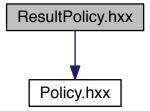
class policy::NoPolicy

### **Namespaces**

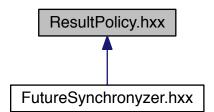
policy

## 7.5 ResultPolicy.hxx File Reference

#include "Policy.hxx"
Include dependency graph for ResultPolicy.hxx:



This graph shows which files directly or indirectly include this file:



#### **Classes**

• class policy::AddNumberResultPolicy

#### **Namespaces**

policy

7			_	-1	- 4	e .
-11	W	n	Δ	n	ΔI	rc
	v	м	c	ч		ı

• typedef NoPolicy policy::NoResultPolicy

30 File Documentation