MyVensin

Generated by Doxygen 1.9.3

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

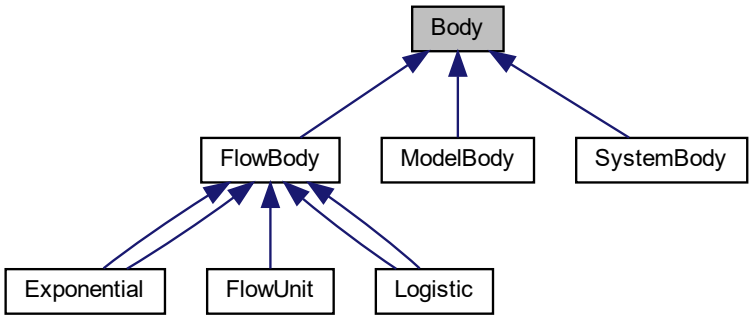
Class Documentation

4.1 Body Class Reference

The class Implementation was implemented based on the class teCounted writed by Ricardo Cartaxo and Gilberto Câmara and founded in the geographic library TerraLib.

`#include <handleBodySemDebug.h>`

Inheritance diagram for Body:



Public Member Functions

- Body ()

  Constructor: zero references when the object is being built.
- void attach ()

  Increases the number of references to this object.
- void detach ()
- int refCount ()

  Returns the number of references to this object.
- virtual ~Body ()

  Destructor.

4.1.1 Detailed Description

The class Implementation was implemented based on the class teCounted writed by Ricardo Cartaxo and Gilberto Câmara and founded in the geographic library TerraLib.

Definition at line 75 of file handleBodySemDebug.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Body()

```
Body::Body ( )  [inline]
```

Constructor: zero references when the object is being built.

Definition at line 78 of file handleBodySemDebug.h.

4.1.2.2 ~Body()

```
virtual Body::~Body ( )  [inline], [virtual]
```

Destructor.

Definition at line 100 of file handleBodySemDebug.h.

4.1.3 Member Function Documentation

4.1.3.1 attach()

```
void Body::attach ( )  [inline]
```

Increases the number of references to this object.

Definition at line 86 of file handleBodySemDebug.h.

4.1.3.2 detach()

```
void Body::detach ( )  [inline]
```

Decreases the number of references to this object. Destroy it if there are no more references to it

Definition at line 90 of file handleBodySemDebug.h.

4.1.3.3 refCount()

```
int Body::refCount ( )  [inline]
```

Returns the number of references to this object.

Definition at line 97 of file handleBodySemDebug.h.

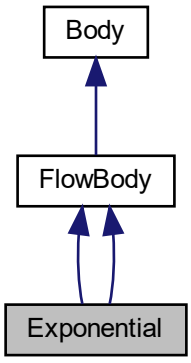
The documentation for this class was generated from the following file:

- src/lib/handleBodySemDebug.h

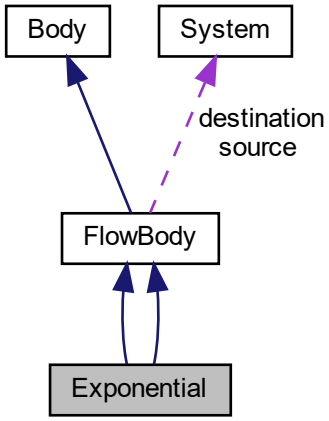
4.2 Exponential Class Reference

```
#include <functional_tests.h>
```

Inheritance diagram for Exponential:



Collaboration diagram for Exponential:



Public Member Functions

- Exponential ()
- ~Exponential ()
- double run ()
- Exponential ()
- ~Exponential ()
- double run ()

Additional Inherited Members

4.2.1 Detailed Description

Definition at line 42 of file functional_tests.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Exponential() [1/2]

```
Exponential::Exponential ( )  [inline]
```

Definition at line 44 of file functional_tests.h.

4.2.2.2 ~Exponential() [1/2]

```
Exponential::~Exponential ( )  [inline]
```

Definition at line 45 of file functional_tests.h.

4.2.2.3 Exponential() [2/2]

```
Exponential::Exponential ( )  [inline]
```

Definition at line 30 of file unit_model.h.

4.2.2.4 ~Exponential() [2/2]

```
Exponential::~Exponential ( )  [inline]
```

Definition at line 31 of file unit_model.h.

4.2.3 Member Function Documentation

4.2.3.1 run() [1/2]

```
double Exponential::run ( )  [inline], [virtual]
```

Implements FlowBody.

Definition at line 46 of file functional_tests.h.

4.2.3.2 run() [2/2]

```
double Exponential::run ( )  [inline], [virtual]
```

Implements FlowBody.

Definition at line 32 of file unit_model.h.

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

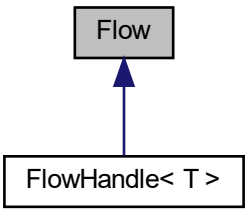
- test/functional/functional_tests.h
- test/unit/unit_model.h

4.3 Flow Class Reference

File responsible for project flows.

`#include <flow.h>`

Inheritance diagram for Flow:



Public Member Functions

- virtual ~Flow ()
  Destructor to destroy the flow.
- virtual void setSources (System *)=0
  Add an input system to the stream.
- virtual void setDestination (System *)=0
  Add an exit system to the stream.
- virtual System * getSource ()=0
  Function to return an input system.
- virtual System * getDestination ()=0
  Function to return an output system.
- virtual double run ()=0
  Virtual function to run the stream.

4.3.1 Detailed Description

File responsible for project flows.

Author

Gabriel Niquini 19.1.4113

Definition at line 12 of file flow.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 ~Flow()

```
virtual Flow::~Flow ( )  [inline], [virtual]
```

Destructor to destroy the flow.

Definition at line 18 of file flow.h.

4.3.3 Member Function Documentation

4.3.3.1 getDestination()

```
virtual System * Flow::getDestination ( )  [pure virtual]
```

Function to return an output system.

Returns

Returns a System object.

Implemented in FlowHandle< T >.

4.3.3.2 getSource()

```
virtual System * Flow::getSource ( )  [pure virtual]
```

Function to return an input system.

Returns

Returns a System object.

Implemented in FlowHandle< T >.

4.3.3.3 run()

```
virtual double Flow::run ( )  [pure virtual]
```

Virtual function to run the stream.

Returns

Returns value of 0.

Implemented in FlowHandle< T >.

4.3.3.4 setDestination()

```
virtual void Flow::setDestination (
            System *  )  [pure virtual]
```

Add an exit system to the stream.

Parameters

system	System pointer.

Implemented in FlowHandle< T >.

4.3.3.5 setSources()

```
virtual void Flow::setSources (
            System *  )  [pure virtual]
```

Add an input system to the stream.

Parameters

system	System pointer.

Implemented in FlowHandle< T >.

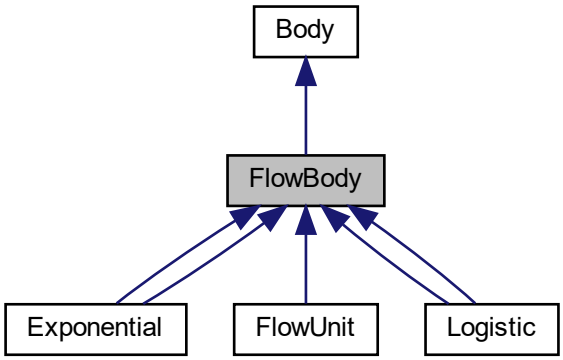
The documentation for this class was generated from the following file:

- src/lib/flow.h

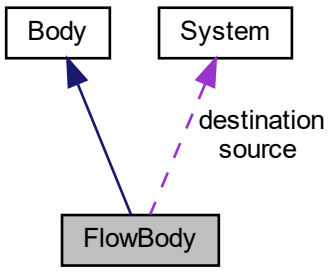
4.4 FlowBody Class Reference

```
#include <flow_Imp.h>
```

Inheritance diagram for FlowBody:



Collaboration diagram for FlowBody:



Public Member Functions

- FlowBody ()
- FlowBody (System ∗, System ∗)
- virtual ∼FlowBody ()
- void setSources (System ∗)
- void setDestination (System ∗)
- System ∗ getSource ()
- System ∗ getDestination ()
- virtual double run ()=0
- FlowBody ∗ operator= (FlowBody ∗)

Protected Attributes

- System ∗ source
- System ∗ destination

4.4.1 Detailed Description

Definition at line 11 of file flow_Imp.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 FlowBody() [1/2]

```
FlowBody::FlowBody ( )
```

Definition at line 3 of file flow_Imp.cpp.

4.4.2.2 FlowBody() [2/2]

```
FlowBody::FlowBody (
            System * source,
            System * destination )
```

Definition at line 5 of file flow_Imp.cpp.

4.4.2.3 ~FlowBody()

```
FlowBody::~FlowBody ( )  [virtual]
```

Definition at line 10 of file flow_Imp.cpp.

4.4.3 Member Function Documentation

4.4.3.1 getDestination()

```
System * FlowBody::getDestination ( )
```

Definition at line 24 of file flow_Imp.cpp.

4.4.3.2 getSource()

```
System * FlowBody::getSource ( )
```

Definition at line 20 of file flow_Imp.cpp.

4.4.3.3 operator=()

```
FlowBody * FlowBody::operator= (
            FlowBody * flow )
```

Definition at line 28 of file flow_Imp.cpp.

4.4.3.4 run()

```
virtual double FlowBody::run ( )  [pure virtual]
```

Implemented in Exponential, Logistic, FlowUnit, Exponential, and Logistic.

4.4.3.5 setDestination()

```
void FlowBody::setDestination (
            System * destination )
```

Definition at line 16 of file flow_Imp.cpp.

4.4.3.6 setSources()

```
void FlowBody::setSources (
            System * source )
```

Definition at line 12 of file flow_Imp.cpp.

4.4.4 Member Data Documentation

4.4.4.1 destination

```
System* FlowBody::destination  [protected]
```

Definition at line 14 of file flow_Imp.h.

4.4.4.2 source

```
System* FlowBody::source  [protected]
```

Definition at line 13 of file flow_Imp.h.

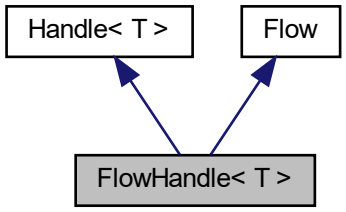
The documentation for this class was generated from the following files:

- src/lib/flow_Imp.h
- src/lib/flow_Imp.cpp

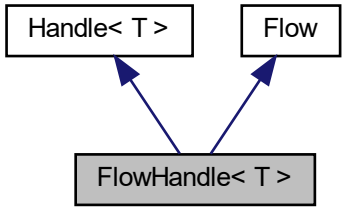
4.5 FlowHandle< T > Class Template Reference

`#include <flow_Imp.h>`

Inheritance diagram for FlowHandle< T >:



Collaboration diagram for FlowHandle< T >:



Public Member Functions

- FlowHandle ()
- virtual ~FlowHandle ()
- void setSources (System *source)

  Add an input system to the stream.
- void setDestination (System *destination)

  Add an exit system to the stream.
- System * getSource ()

  Function to return an input system.
- System * getDestination ()

  Function to return an output system.
- virtual double run ()

  Virtual function to run the stream.

Additional Inherited Members

4.5.1 Detailed Description

template<typename T>
class FlowHandle< T >

Definition at line 28 of file flow_Imp.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 FlowHandle()

```
template<typename T >
FlowHandle< T >::FlowHandle ( )  [inline]
```

Definition at line 24 of file flow_Imp.h.

4.5.2.2 ~FlowHandle()

```
template<typename T >
virtual FlowHandle< T >::~FlowHandle ( )  [inline], [virtual]
```

Definition at line 31 of file flow_Imp.h.

4.5.3 Member Function Documentation

4.5.3.1 getDestination()

```
template<typename T >
System * FlowHandle< T >::getDestination ( )  [inline], [virtual]
```

Function to return an output system.

Returns

Returns a System object.

Implements Flow.

Definition at line 41 of file flow_Imp.h.

4.5.3.2 getSource()

```
template<typename T >
System * FlowHandle< T >::getSource ( )  [inline], [virtual]
```

Function to return an input system.

Returns

Returns a System object.

Implements Flow.

Definition at line 38 of file flow_Imp.h.

4.5.3.3 run()

```
template<typename T >
virtual double FlowHandle< T >::run ( )  [inline], [virtual]
```

Virtual function to run the stream.

Returns

Returns value of 0.

Implements Flow.

Definition at line 44 of file flow_Imp.h.

4.5.3.4 setDestination()

```
template<typename T >
void FlowHandle< T >::setDestination (
            System *  )  [inline], [virtual]
```

Add an exit system to the stream.

Parameters

system	System pointer.

Implements Flow.

Definition at line 35 of file flow_Imp.h.

4.5.3.5 setSources()

```
template<typename T >
void FlowHandle< T >::setSources (
             System *  )  [inline], [virtual]
```

Add an input system to the stream.

Parameters

system	System pointer.

Implements Flow.

Definition at line 32 of file flow_Imp.h.

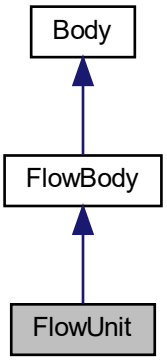
The documentation for this class was generated from the following file:

- src/lib/flow_Imp.h

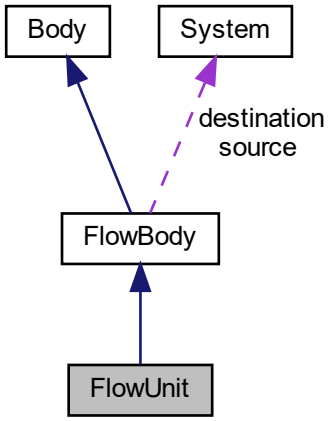
4.6 FlowUnit Class Reference

```
#include <unit_flow.h>
```

Inheritance diagram for FlowUnit:



Collaboration diagram for FlowUnit:



Public Member Functions

- FlowUnit ()
- ~FlowUnit ()
- double run ()

Additional Inherited Members

4.6.1 Detailed Description

Definition at line 25 of file unit_flow.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 FlowUnit()

```
FlowUnit::FlowUnit ( )  [inline]
```

Definition at line 27 of file unit_flow.h.

4.6.2.2 ~FlowUnit()

```
FlowUnit::~FlowUnit ( )  [inline]
```

Definition at line 28 of file unit_flow.h.

4.6.3 Member Function Documentation

4.6.3.1 run()

```
double FlowUnit::run ( )  [inline], [virtual]
```

Implements FlowBody.

Definition at line 29 of file unit_flow.h.

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

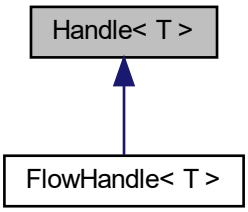
- test/unit/unit_flow.h

4.7 Handle< T > Class Template Reference

The classes Handle and Body implements the "bridge" design pattern (also known as "handle/body idiom").

```
#include <handleBodySemDebug.h>
```

Inheritance diagram for Handle< T >:



Public Member Functions

- Handle ()
  constructor
- virtual ~Handle ()
  Destructor.
- Handle (const Handle &hd)
  copy constructor
- Handle< T > & operator= (const Handle &hd)
  assignment operator

Protected Attributes

- T * pImpl_
  referência para a implementação

4.7.1 Detailed Description

template<class T>
class Handle< T >

The classes Handle and Body implements the "bridge" design pattern (also known as "handle/body idiom").

Definition at line 28 of file handleBodySemDebug.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Handle() [1/2]

```
template<class T >
Handle< T >::Handle ( )  [inline]
```

constructor

Definition at line 17 of file handleBodySemDebug.h.

4.7.2.2 ~Handle()

```
template<class T >
virtual Handle< T >::~Handle ( )  [inline], [virtual]
```

Destructor.

Definition at line 17 of file handleBodySemDebug.h.

4.7.2.3 Handle() [2/2]

```
template<class T >
Handle< T >::Handle (
            const Handle< T > & hd )  [inline]
```

copy constructor

Definition at line 17 of file handleBodySemDebug.h.

4.7.3 Member Function Documentation

4.7.3.1 operator=()

```
template<class T >
Handle< T > & Handle< T >::operator= (
            const Handle< T > & hd )  [inline]
```

assignment operator

Definition at line 53 of file handleBodySemDebug.h.

4.7.4 Member Data Documentation

4.7.4.1 pImpl_

```
template<class T >
T* Handle< T >::pImpl_  [protected]
```

referência para a implementação

Definition at line 66 of file handleBodySemDebug.h.

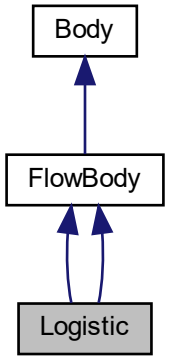
The documentation for this class was generated from the following file:

- src/lib/handleBodySemDebug.h

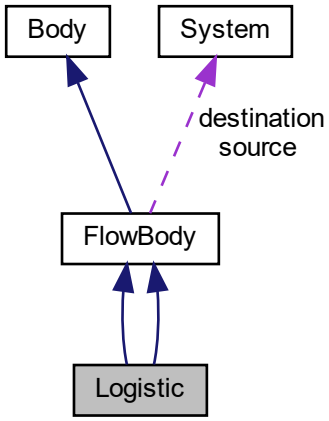
4.8 Logistic Class Reference

`#include <functional_tests.h>`

Inheritance diagram for Logistic:



Collaboration diagram for Logistic:



Public Member Functions

- Logistic ()
- ~Logistic ()
- double run ()
- Logistic ()
- ~Logistic ()
- double run ()

Additional Inherited Members

4.8.1 Detailed Description

Definition at line 51 of file functional_tests.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Logistic() [1/2]

```
Logistic::Logistic ( )  [inline]
```

Definition at line 53 of file functional_tests.h.

4.8.2.2 ~Logistic() [1/2]

```
Logistic::~Logistic ( )  [inline]
```

Definition at line 54 of file functional_tests.h.

4.8.2.3 Logistic() [2/2]

```
Logistic::Logistic ( )  [inline]
```

Definition at line 39 of file unit_model.h.

4.8.2.4 ~Logistic() [2/2]

```
Logistic::~Logistic ( )  [inline]
```

Definition at line 40 of file unit_model.h.

4.8.3 Member Function Documentation

4.8.3.1 run() [1/2]

```
double Logistic::run ( )  [inline], [virtual]
```

Implements FlowBody.

Definition at line 55 of file functional_tests.h.

4.8.3.2 run() [2/2]

```
double Logistic::run ( )  [inline], [virtual]
```

Implements FlowBody.

Definition at line 41 of file unit_model.h.

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

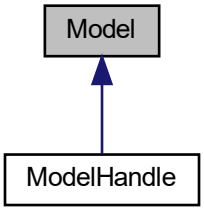
- test/functional/functional_tests.h
- test/unit/unit_model.h

4.9 Model Class Reference

File responsible for project templates.

```
#include <model.h>
```

Inheritance diagram for Model:



Public Member Functions

- virtual ~Model ()

  Destructor to destroy the model.
- virtual double run (int, int)=0

  Function to run the model.
- virtual System * createSystem (string, double)=0
- virtual System * getSystem (string name)=0

  Function to overload operator =.
- virtual vector< Flow * > getFlows ()=0
- template<typename T_FLOW >
  Flow * createFlow (System *source=nullptr, System *destination=nullptr)

Static Public Member Functions

- static Model * createModel (string)

  Function to create the model.

4.9.1 Detailed Description

File responsible for project templates.

Author

Gabriel Niquini 19.1.4113

Definition at line 12 of file model.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 ~Model()

```
virtual Model::~Model ( )  [inline], [virtual]
```

Destructor to destroy the model.

Definition at line 18 of file model.h.

4.9.3 Member Function Documentation

4.9.3.1 createFlow()

```
template<typename T_FLOW >
Flow * Model::createFlow (
            System * source = nullptr,
            System * destination = nullptr )  [inline]
```

Definition at line 48 of file model.h.

4.9.3.2 createModel()

```
Model * Model::createModel (
            string id )  [static]
```

Function to create the model.

Parameters

string	Initial value.

Returns

Returns final model.

Definition at line 21 of file model_Imp.cpp.

4.9.3.3 createSystem()

```
virtual System * Model::createSystem (
            string ,
            double  )  [pure virtual]
```

Implemented in ModelHandle.

4.9.3.4 getFlows()

```
virtual vector< Flow * > Model::getFlows ( )  [pure virtual]
```

Implemented in ModelHandle.

4.9.3.5 getSystem()

```
virtual System * Model::getSystem (
            string name )  [pure virtual]
```

Function to overload operator =.

Parameters

model	Model pointer.

Returns

Returns model.

Implemented in ModelHandle.

4.9.3.6 run()

```
virtual double Model::run (
            int ,
            int  )  [pure virtual]
```

Function to run the model.

Parameters

start	Initial value.
finish	Final value.

Returns

Returns final value.

Implemented in ModelHandle.

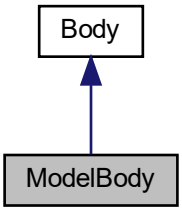
The documentation for this class was generated from the following files:

- src/lib/model.h
- src/lib/model_Imp.cpp

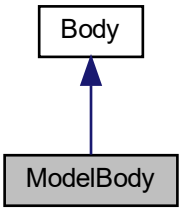
4.10 ModelBody Class Reference

```
#include <model_Imp.h>
```

Inheritance diagram for ModelBody:



Collaboration diagram for ModelBody:



Public Member Functions

- ModelBody ()
- ModelBody (string)
- virtual ~ModelBody ()
- double run (int, int)
- System * createSystem (string, double)
- System * getSystem (string name)
- vector< Flow * > getFlows ()
- string getId ()
- void setId (string)
- void add (System *)
- void add (Flow *)
- void add (Model *model)

Static Public Member Functions

- static Model * createModel (string)

Protected Attributes

- vector< Flow * > flows
- vector< System * > systems
- string id

Static Protected Attributes

- static vector< Model * > models

4.10.1 Detailed Description

Definition at line 16 of file model_Imp.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 ModelBody() [1/2]

```
ModelBody::ModelBody ( )
```

Definition at line 8 of file model_Imp.cpp.

4.10.2.2 ModelBody() [2/2]

```
ModelBody::ModelBody (
            string id )
```

Definition at line 10 of file model_Imp.cpp.

4.10.2.3 ~ModelBody()

```
ModelBody::~ModelBody ( )  [virtual]
```

Definition at line 14 of file model_Imp.cpp.

4.10.3 Member Function Documentation

4.10.3.1 add() [1/3]

```
void ModelBody::add (
            Flow * flow )
```

Definition at line 57 of file model_Imp.cpp.

4.10.3.2 add() [2/3]

```
void ModelBody::add (
            Model * model )
```

Definition at line 61 of file model_Imp.cpp.

4.10.3.3 add() [3/3]

```
void ModelBody::add (
            System * system )
```

Definition at line 53 of file model_Imp.cpp.

4.10.3.4 createModel()

```
static Model * ModelBody::createModel (
            string  )  [static]
```

4.10.3.5 createSystem()

```
System * ModelBody::createSystem (
            string name,
            double value )
```

Definition at line 47 of file model_Imp.cpp.

4.10.3.6 getFlows()

```
vector< Flow * > ModelBody::getFlows ( )
```

Definition at line 73 of file model_Imp.cpp.

4.10.3.7 getId()

```
string ModelBody::getId ( )
```

Definition at line 77 of file model_Imp.cpp.

4.10.3.8 getSystem()

```
System * ModelBody::getSystem (
            string name )
```

Definition at line 65 of file model_Imp.cpp.

4.10.3.9 run()

```
double ModelBody::run (
            int start,
            int finish )
```

Definition at line 25 of file model_Imp.cpp.

4.10.3.10 setId()

```
void ModelBody::setId (
            string id )
```

Definition at line 81 of file model_Imp.cpp.

4.10.4 Member Data Documentation

4.10.4.1 flows

```
vector<Flow*> ModelBody::flows  [protected]
```

Definition at line 18 of file model_Imp.h.

4.10.4.2 id

```
string ModelBody::id  [protected]
```

Definition at line 21 of file model_Imp.h.

4.10.4.3 models

```
vector< Model * > ModelBody::models  [static], [protected]
```

Definition at line 20 of file model_Imp.h.

4.10.4.4 systems

```
vector<System*> ModelBody::systems  [protected]
```

Definition at line 19 of file model_Imp.h.

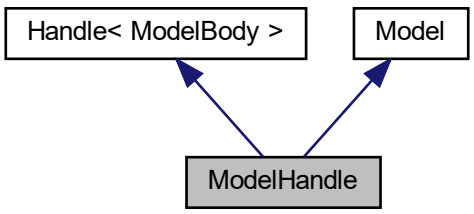
The documentation for this class was generated from the following files:

- src/lib/model_Imp.h
- src/lib/model_Imp.cpp

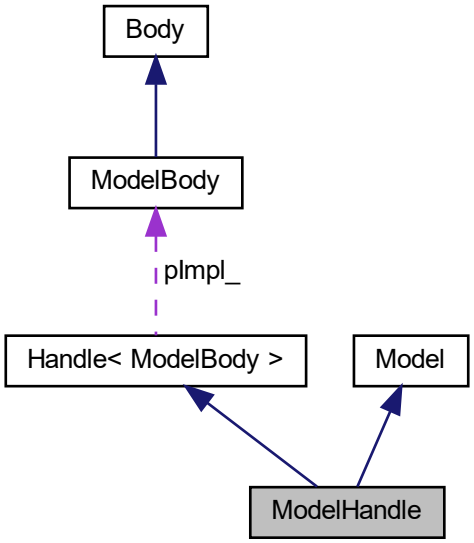
4.11 ModelHandle Class Reference

```
#include <model_Imp.h>
```

Inheritance diagram for ModelHandle:



Collaboration diagram for ModelHandle:



Public Member Functions

- ModelHandle ()
- virtual ~ModelHandle ()
- double run (int start, int finish)

  Function to run the model.
- System * createSystem (string name, double value)
- System * getSystem (string name)

  Function to overload operator =.
- vector< Flow * > getFlows ()
- string getId ()
- void setId (string id)
- void add (System *system)

  Function to add a new system.
- void add (Flow *flow)

  Function to add new flow.
- void add (Model *model)

Static Public Member Functions

- static Model * createModel (string id)

Additional Inherited Members

4.11.1 Detailed Description

Definition at line 38 of file model_Imp.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 ModelHandle()

```
ModelHandle::ModelHandle ( )  [inline]
```

Definition at line 40 of file model_Imp.h.

4.11.2.2 ~ModelHandle()

```
virtual ModelHandle::~ModelHandle ( )  [inline], [virtual]
```

Definition at line 41 of file model_Imp.h.

4.11.3 Member Function Documentation

4.11.3.1 add() [1/3]

```
void ModelHandle::add (
            Flow *  )  [inline], [virtual]
```

Function to add new flow.

Implements Model.

Definition at line 69 of file model_Imp.h.

4.11.3.2 add() [2/3]

```
void ModelHandle::add (
            Model * model )  [inline]
```

Definition at line 72 of file model_Imp.h.

4.11.3.3 add() [3/3]

```
void ModelHandle::add (
            System *  )  [inline], [virtual]
```

Function to add a new system.

Implements Model.

Definition at line 66 of file model_Imp.h.

4.11.3.4 createModel()

```
static Model * ModelHandle::createModel (
            string id )  [inline], [static]
```

Definition at line 42 of file model_Imp.h.

4.11.3.5 createSystem()

```
System * ModelHandle::createSystem (
            string name,
            double value )  [inline], [virtual]
```

Implements Model.

Definition at line 51 of file model_Imp.h.

4.11.3.6 getFlows()

```
vector< Flow * > ModelHandle::getFlows ( )  [inline], [virtual]
```

Implements Model.

Definition at line 57 of file model_Imp.h.

4.11.3.7 getId()

```
string ModelHandle::getId ( )  [inline]
```

Definition at line 60 of file model_Imp.h.

4.11.3.8 getSystem()

```
System * ModelHandle::getSystem (
            string name )  [inline], [virtual]
```

Function to overload operator =.

Parameters

model	Model pointer.

Returns

Returns model.

Implements Model.

Definition at line 54 of file model_Imp.h.

4.11.3.9 run()

```
double ModelHandle::run (
            int ,
            int  )  [inline], [virtual]
```

Function to run the model.

Parameters

start	Initial value.
finish	Final value.

Returns

Returns final value.

Implements Model.

Definition at line 48 of file model_Imp.h.

4.11.3.10 setId()

```
void ModelHandle::setId (
            string id )  [inline]
```

Definition at line 63 of file model_Imp.h.

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

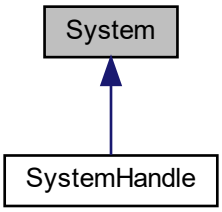
- src/lib/model_Imp.h

4.12 System Class Reference

File responsible for project systems.

`#include <system.h>`

Inheritance diagram for System:



Public Member Functions

- virtual ~System ()

  Destructor to destroy the system.
- virtual void setName (string)=0

  Add a name for the system.
- virtual void setValue (double)=0

  Add a value to the system.
- virtual double getValue ()=0

  Function to return system value.
- virtual string getName ()=0

  Function to return system name.

4.12.1 Detailed Description

File responsible for project systems.

Author

Gabriel Niquini 19.1.4113

Definition at line 15 of file system.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 ~System()

```
virtual System::~System ( )  [inline], [virtual]
```

Destructor to destroy the system.

Definition at line 21 of file system.h.

4.12.3 Member Function Documentation

4.12.3.1 getName()

```
virtual string System::getName ( )  [pure virtual]
```

Function to return system name.

Returns

Returns a string.

Implemented in SystemHandle.

4.12.3.2 getValue()

```
virtual double System::getValue ( )  [pure virtual]
```

Function to return system value.

Returns

Returns a double.

Implemented in SystemHandle.

4.12.3.3 setName()

```
virtual void System::setName (
            string  )  [pure virtual]
```

Add a name for the system.

Parameters

name	System name.

Implemented in SystemHandle.

4.12.3.4 setValue()

```
virtual void System::setValue (
            double  )  [pure virtual]
```

Add a value to the system.

Parameters

value	System value.

Implemented in SystemHandle.

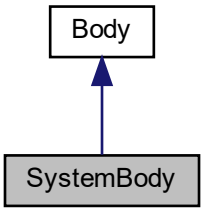
The documentation for this class was generated from the following file:

- src/lib/system.h

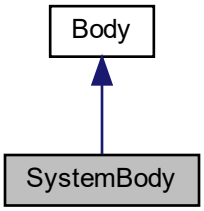
4.13 SystemBody Class Reference

```
#include <system_Imp.h>
```

Inheritance diagram for SystemBody:



Collaboration diagram for SystemBody:



Public Member Functions

- SystemBody ()
- SystemBody (string n, double v)
- virtual ~SystemBody ()
- void setName (string)
- void setValue (double)
- double getValue ()
- string getName ()
- SystemBody * operator= (SystemBody *system)

Protected Attributes

- string name
- double value

4.13.1 Detailed Description

Definition at line 12 of file system_Imp.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 SystemBody() [1/2]

```
SystemBody::SystemBody ( )
```

Definition at line 3 of file system_Imp.cpp.

4.13.2.2 SystemBody() [2/2]

```
SystemBody::SystemBody (
            string n,
            double v )
```

Definition at line 7 of file system_lmp.cpp.

4.13.2.3 ~SystemBody()

```
SystemBody::~SystemBody ( )  [virtual]
```

Definition at line 5 of file system_lmp.cpp.

4.13.3 Member Function Documentation

4.13.3.1 getName()

```
string SystemBody::getName ( )
```

Definition at line 20 of file system_lmp.cpp.

4.13.3.2 getValue()

```
double SystemBody::getValue ( )
```

Definition at line 24 of file system_lmp.cpp.

4.13.3.3 operator=()

```
SystemBody * SystemBody::operator= (
            SystemBody * system )
```

Definition at line 28 of file system_lmp.cpp.

4.13.3.4 setName()

```
void SystemBody::setName (
            string name )
```

Definition at line 12 of file system_lmp.cpp.

4.13.3.5 setValue()

```
void SystemBody::setValue (
            double value )
```

Definition at line 16 of file system_lmp.cpp.

4.13.4 Member Data Documentation

4.13.4.1 name

```
string SystemBody::name  [protected]
```

Definition at line 14 of file system_lmp.h.

4.13.4.2 value

```
double SystemBody::value  [protected]
```

Definition at line 15 of file system_lmp.h.

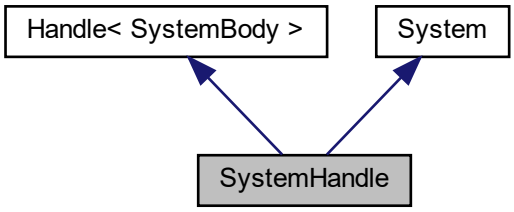
The documentation for this class was generated from the following files:

- src/lib/system_lmp.h
- src/lib/system_lmp.cpp

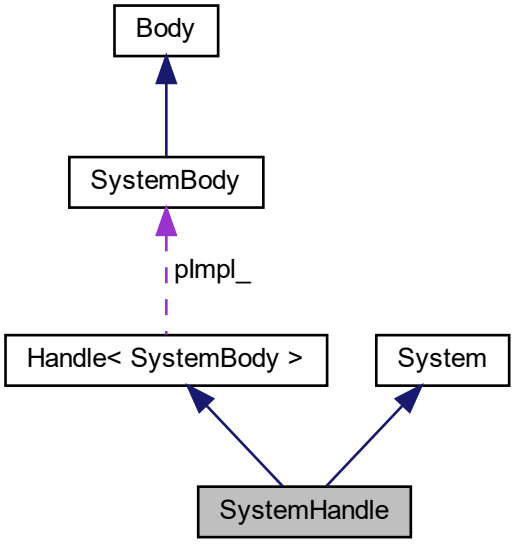
4.14 SystemHandle Class Reference

`#include <system_Imp.h>`

Inheritance diagram for SystemHandle:



Collaboration diagram for SystemHandle:



Public Member Functions

- SystemHandle ()
- SystemHandle (string n, double v)
- virtual ~SystemHandle ()
- void setName (string name)

Add a name for the system.

- void setValue (double value)

  Add a value to the system.

- double getValue ()

  Function to return system value.

- string getName ()

  Function to return system name.

Additional Inherited Members

4.14.1 Detailed Description

Definition at line 27 of file system_Imp.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 SystemHandle() [1/2]

```
SystemHandle::SystemHandle ( )  [inline]
```

Definition at line 29 of file system_Imp.h.

4.14.2.2 SystemHandle() [2/2]

```
SystemHandle::SystemHandle (
            string n,
            double v )  [inline]
```

Definition at line 30 of file system_Imp.h.

4.14.2.3 ~SystemHandle()

```
virtual SystemHandle::~SystemHandle ( )  [inline], [virtual]
```

Definition at line 34 of file system_Imp.h.

4.14.3 Member Function Documentation

4.14.3.1 getName()

```
string SystemHandle::getName ( )  [inline], [virtual]
```

Function to return system name.

Returns

Returns a string.

Implements System.

Definition at line 44 of file system_Imp.h.

4.14.3.2 getValue()

```
double SystemHandle::getValue ( )  [inline], [virtual]
```

Function to return system value.

Returns

Returns a double.

Implements System.

Definition at line 41 of file system_Imp.h.

4.14.3.3 setName()

```
void SystemHandle::setName (
            string  )  [inline], [virtual]
```

Add a name for the system.

Parameters

name	System name.

Implements System.

Definition at line 35 of file system_Imp.h.

4.14.3.4 setValue()

```
void SystemHandle::setValue (
            double  )  [inline], [virtual]
```

Add a value to the system.

Parameters

value	System value.

Implements System.

Definition at line 38 of file system_lmp.h.

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

- src/lib/system_lmp.h

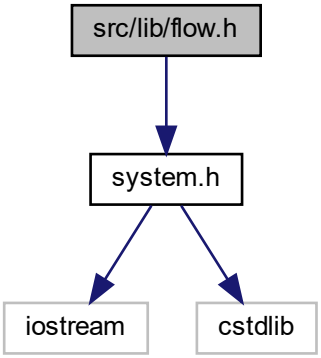
Chapter 5

File Documentation

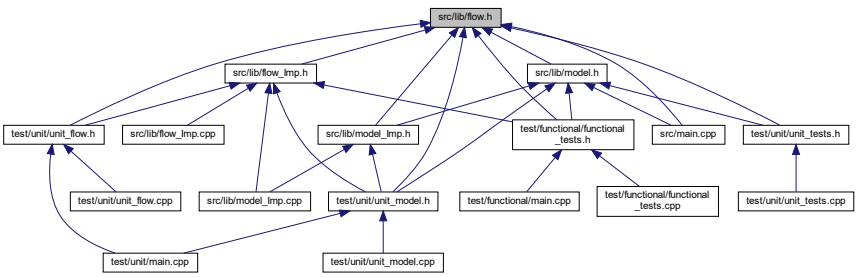
5.1 src/lib/flow.h File Reference

```
#include "system.h"
```

Include dependency graph for flow.h:



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



Classes

- class Flow

  File responsible for project flows.

5.2 flow.h

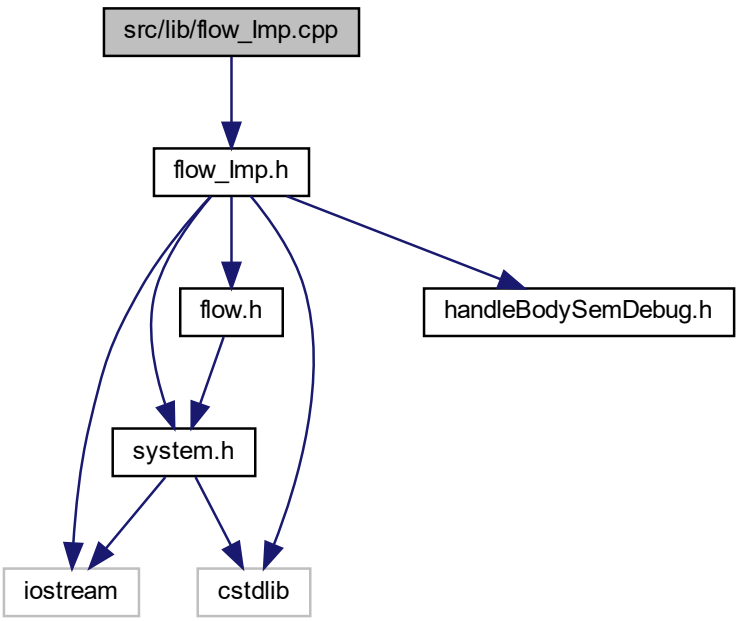
Go to the documentation of this file.

```
00001 #ifndef FLOW_H
00002 #define FLOW_H
00003
00004 #include "system.h"
00005
00012 class Flow{
00013     public:
00018         virtual ~Flow(){};
00019
00025         virtual void setSources(System*) = 0;
00026
00027
00032         virtual void setDestination(System*) = 0;
00033
00039         virtual System* getSource() = 0;
00040
00046         virtual System* getDestination() = 0;
00047
00053         virtual double run() = 0;
00054
00055 };
00056
00057 #endif
```

5.3 src/lib/flow_Imp.cpp File Reference

```
#include "flow_Imp.h"
```

Include dependency graph for flow_Imp.cpp:



5.4 flow_Imp.cpp

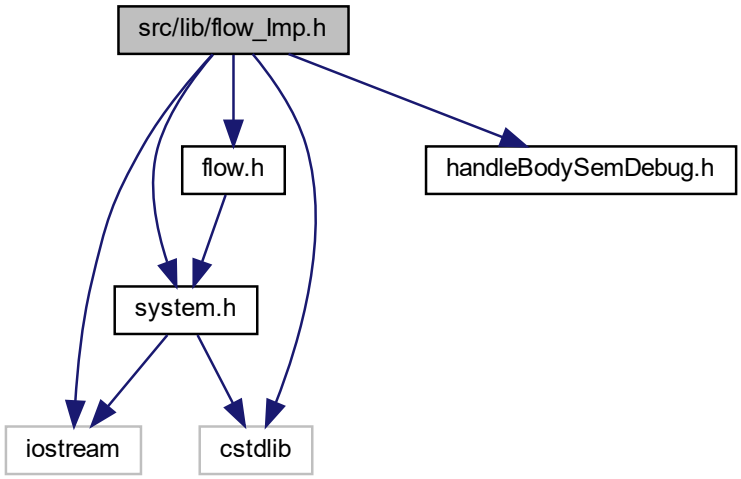
Go to the documentation of this file.

```
00001 #include "flow_Imp.h"
00002
00003 FlowBody::FlowBody(){}
00004
00005 FlowBody::FlowBody(System* source, System* destination){
00006     this-> source = source;
00007     this->destination = destination;
00008 }
00009
00010 FlowBody::~FlowBody(){}
00011
00012 void FlowBody::setSources(System* source){
00013     this->source = source;
00014 }
00015
00016 void FlowBody::setDestination(System* destination){
00017     this->destination = destination;
00018 }
00019
00020 System* FlowBody::getSource(){
00021     return this->source;
00022 }
00023
00024 System* FlowBody::getDestination(){
00025     return this->destination;
00026 }
00027
00028 FlowBody* FlowBody::operator=(FlowBody* flow){
00029     if (this == flow)
00030         return this;
00031
00032     this->source = flow->getSource();
00033     this->destination = flow->getSource();
00034     return this;
00035 }
```

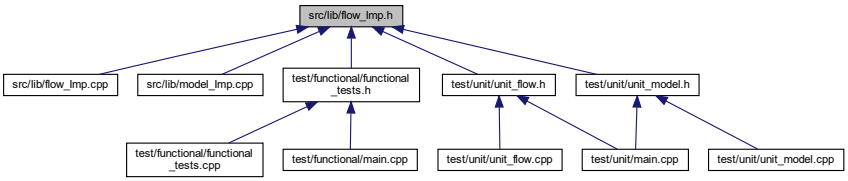
5.5 src/lib/flow_Imp.h File Reference

```
#include "system.h"
#include "flow.h"
#include "handleBodySemDebug.h"
#include <iostream>
#include <cstdlib>
```

Include dependency graph for flow_Imp.h:



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



Classes

- class FlowBody
- class FlowHandle< T >

5.6 flow_Imp.h

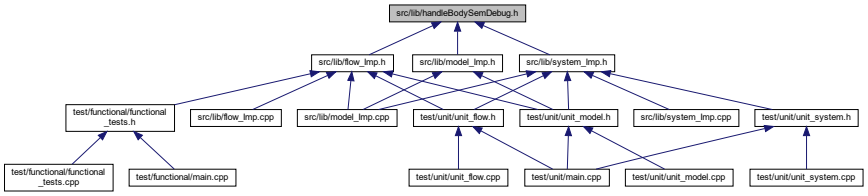
Go to the documentation of this file.

```
00001 #ifndef FLOW_IMP_H
00002 #define FLOW_IMP_H
00003
00004 #include "system.h"
00005 #include "flow.h"
00006 #include "handleBodySemDebug.h"
00007
00008 #include <iostream>
00009 #include <cstdlib>
00010
00011 class FlowBody : public Body{
00012     protected:
```

```
00013         System* source;
00014         System* destination;
00015     public:
00016         FlowBody();
00017         FlowBody(System*, System*);
00018         virtual ~FlowBody();
00019         void setSources(System*);
00020         void setDestination(System*);
00021         System* getSource();
00022         System* getDestination();
00023         virtual double run() = 0;
00024         FlowBody* operator=(FlowBody*);
00025 };
00026
00027 template<typename T>
00028 class FlowHandle : public Handle<T>, public Flow{
00029     public:
00030         FlowHandle<T>():Handle<T>(){};
00031         virtual ~FlowHandle() {};
00032         void setSources(System* source){
00033             this->pImpl_->setSources(source);
00034         }
00035         void setDestination(System* destination){
00036             this->pImpl_->setDestination(destination);
00037         }
00038         System* getSource(){
00039             return this->pImpl_->getSource();
00040         }
00041         System* getDestination(){
00042             return this->pImpl_->getDestination();
00043         }
00044         virtual double run(){
00045             return this->pImpl_->run();
00046         }
00047 };
00048
00049 #endif
```

5.7 src/lib/handleBodySemDebug.h File Reference

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



Classes

- class Handle< T >

  The classes Handle and Body implements the "bridge" design pattern (also known as "handle/body idiom").
- class Body

  The class Implementation was implemented based on the class teCounted writed by Ricardo Cartaxo and Gilberto Câmara and founded in the geographic library TerraLib.

Macros

- #define DEBUGING

Variables

- int numHandleCreated
- int numHandleDeleted
- int numBodyCreated
- int numBodyDeleted

5.7.1 Macro Definition Documentation

5.7.1.1 DEBUGING

```
#define DEBUGING
```

Definition at line 12 of file handleBodySemDebug.h.

5.7.2 Variable Documentation

5.7.2.1 numBodyCreated

```
int numBodyCreated  [extern]
```

Definition at line 9 of file main.cpp.

5.7.2.2 numBodyDeleted

```
int numBodyDeleted  [extern]
```

Definition at line 10 of file main.cpp.

5.7.2.3 numHandleCreated

```
int numHandleCreated  [extern]
```

Definition at line 7 of file main.cpp.

5.7.2.4 numHandleDeleted

```
int numHandleDeleted  [extern]
```

Definition at line 8 of file main.cpp.

5.8 handleBodySemDebug.h

Go to the documentation of this file.

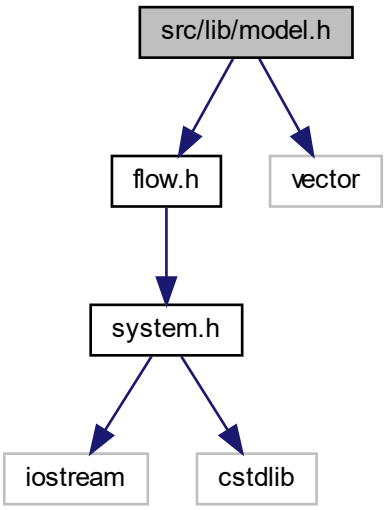
```
00001
00009 #if ! defined( HANDLE_BODY )
00010 #define HANDLE_BODY
00011
00012 #define DEBUGING
00013 #ifdef DEBUGING
00014     extern int numHandleCreated;
00015     extern int numHandleDeleted;
00016     extern int numBodyCreated;
00017     extern int  numBodyDeleted;
00018 #endif
00019
00027 template <class T>
00028 class Handle {
00029
00030     public:
00031
00033         Handle<T>( ){
00034             pImpl_ = new T;
00035             pImpl_->attach();
00036             #ifdef DEBUGING
00037                 numHandleCreated++;
00038             #endif
00039         }
00040
00042         virtual ~Handle<T>(){
00043             pImpl_->detach();
00044             #ifdef DEBUGING
00045                 numHandleDeleted++;
00046             #endif
00047         }
00048
00050         Handle<T>( const Handle& hd ):pImpl_( hd.pImpl_ ) { pImpl_->attach();  }
00051
00053         Handle<T>& operator=( const Handle& hd ) {
00054             if (  this != &hd )
00055             {
00056                 hd.pImpl_->attach();
00057                 pImpl_->detach();
00058                 pImpl_  = hd.pImpl_;
00059             }
00060             return *this;
00061         }
00062
00063     protected:
00064
00066         T *pImpl_;
00067 };
00068
00075 class Body {
00076     public:
00078         Body(): refCount_ ( 0 ){
00079             #ifdef DEBUGING
00080                 numBodyCreated++;
00081             #endif
00082         }
00083
00084
00086         void attach ()    { refCount_++; }
00087
00090         void detach (){
00091             if ( --refCount_ == 0 ) {
00092                 delete this;
00093             }
00094         }
00095
00097         int refCount(){ return refCount_; }
00098
00100         virtual ~Body(){
```

```
00101             #ifdef DEBUGING
00102                 numBodyDeleted++;
00103             #endif
00104         }
00105
00106     private:
00107
00109         Body(const Body&);
00110
00112         Body& operator=(const Body&){return *this;}
00113
00114         int refCount_;
00115
00116     };
00117
00118 #endif
```

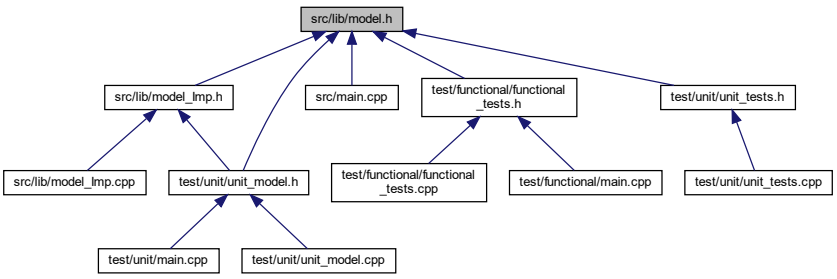
5.9 src/lib/model.h File Reference

```
#include "flow.h"
#include <vector>
```

Include dependency graph for model.h:



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



Classes

- class Model

  File responsible for project templates.

5.10 model.h

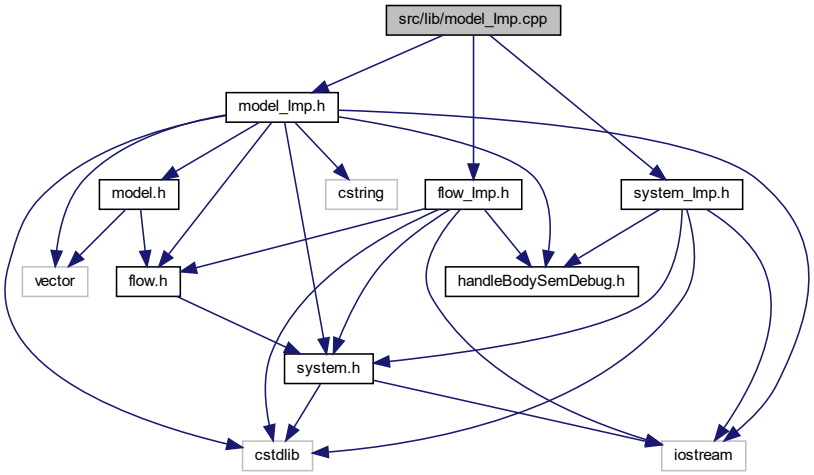
Go to the documentation of this file.

```
00001 #ifndef MODEL_H
00002 #define MODEL_H
00003 #include "flow.h"
00004 #include <vector>
00005
00012 class Model{
00013     public:
00018         virtual ~Model(){};
00019
00027         virtual double run(int,int) = 0;
00028
00035         static Model* createModel(string);
00036         virtual System* createSystem(string,double) = 0;
00037         //virtual Model* operator=(Model*) = 0;
00038
00045         virtual System* getSystem(string name) = 0;
00046         virtual vector<Flow*> getFlows() = 0;
00047         template <typename T_FLOW>
00048         Flow* createFlow(System* source = nullptr, System* destination = nullptr){
00049             Flow* flow = new T_FLOW();
00050             flow->setSources(source);
00051             flow->setDestination(destination);
00052             add(flow);
00053             return flow;
00054         }
00055
00056     private:
00061         virtual void add(System*) = 0;
00062
00067         virtual void add(Flow*) = 0;
00068 };
00069
00070 #endif
```

5.11 src/lib/model_Imp.cpp File Reference

```
#include "model_Imp.h"
#include "flow_Imp.h"
#include "system_Imp.h"
```

Include dependency graph for model_Imp.cpp:



5.12 model_Imp.cpp

Go to the documentation of this file.

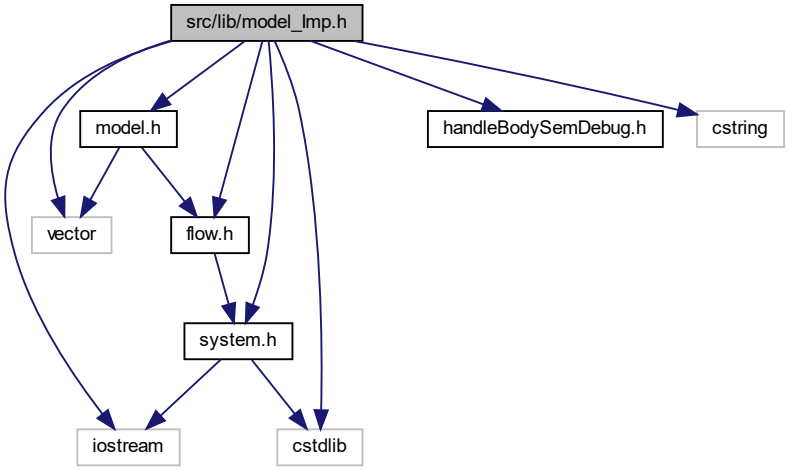
```
00001 #include "model_Imp.h"
00002 #include "flow_Imp.h"
00003 #include "system_Imp.h"
00004
00005 //Global Variable
00006 vector<Model*> ModelBody:: models;
00007
00008 ModelBody::ModelBody(){}
00009
00010 ModelBody::ModelBody(string id){
00011     this->id = id;
00012 }
00013
00014 ModelBody::~ModelBody(){
00015     for (auto it = flows.begin(); it != flows.end(); it++)
00016         delete *it;
00017     for (auto it = systems.begin(); it != systems.end(); it++)
00018         delete *it;
00019 }
00020
00021 Model* Model::createModel(string id){
00022     return ModelHandle::createModel(id);
00023 }
00024
00025 double ModelBody::run(int start,int finish){
00026     vector<double> values;
00027     System* source;
00028     System* destination;
00029     for (int i = start; i < finish; i++){
00030
00031         int size = flows.size();
00032
00033         for(int j=0; j< size; j++){
00034             values.push_back(flows[j]->run());
00035         }
00036         for(int k=0; k < size; k++){
00037             source = flows[k]->getSource();
00038             source->setValue(source->getValue() - values[k]);
00039             destination = flows[k]->getDestination();
00040             destination->setValue(destination->getValue() + values[k]);
00041         }
00042         values.clear();
00043     }
00044     return values[finish];
00045 }
```

```
00046
00047 System* ModelBody::createSystem(string name, double value){
00048     System* s = new SystemHandle(name,value);
00049     this->add(s);
00050     return s;
00051 }
00052
00053 void ModelBody::add(System* system){
00054     this->systems.push_back(system);
00055 }
00056
00057 void ModelBody::add(Flow* flow){
00058     this->flows.push_back(flow);
00059 }
00060
00061 void ModelBody::add(Model* model) {
00062     this->models.push_back(model);
00063 }
00064
00065 System* ModelBody::getSystem(string name){
00066     for(vector<System*>::iterator it= systems.begin(); it != systems.end(); it++){
00067         if(name == (*it)->getName())
00068             return *it;
00069     }
00070     return NULL;
00071 }
00072
00073 vector<Flow*> ModelBody::getFlows(){
00074     return this->flows;
00075 }
00076
00077 string ModelBody::getId(){
00078     return this->id;
00079 }
00080
00081 void ModelBody::setId(string id){
00082     this->id = id;
00083 }
00084 /*
00085 ModelBody* ModelBody::operator=(Model* model){
00086     if(this == model)
00087         return this;
00088
00089     for(vector<System*>::iterator it= systems.begin(); it != systems.end(); it++){
00090         delete *it;
00091     }
00092
00093     this->systems.clear();
00094
00095     for(vector<Flow*>::iterator it= flows.begin(); it != flows.end(); it++){
00096         delete *it;
00097     }
00098
00099     this->flows.clear();
00100
00101     return this;
00102 }*/
```

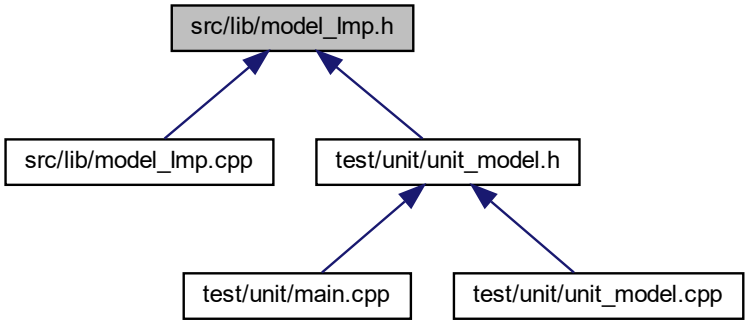
5.13 src/lib/model_Imp.h File Reference

```
#include "model.h"
#include "flow.h"
#include "system.h"
#include "handleBodySemDebug.h"
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <vector>
```

Include dependency graph for model_Imp.h:



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



Classes

- class ModelBody
- class ModelHandle

5.14 model_Imp.h

Go to the documentation of this file.

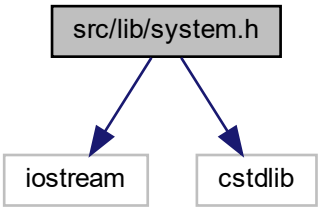
```
00001 #ifndef MODEL_IMP_H
00002 #define MODEL_IMP_H
00003
```

```
00004 #include "model.h"
00005 #include "flow.h"
00006 #include "system.h"
00007 #include "handleBodySemDebug.h"
00008
00009 #include <iostream>
00010 #include <cstring>
00011 #include <cstdlib>
00012 #include <vector>
00013
00014 using namespace std;
00015
00016 class ModelBody : public Body{
00017     protected:
00018         vector<Flow*> flows;
00019         vector<System*> systems;
00020         static vector<Model*> models;
00021         string id;
00022     public:
00023         ModelBody();
00024         ModelBody(string);
00025         virtual ~ModelBody();
00026         static Model* createModel(string);
00027         double run(int,int);
00028         System* createSystem(string,double);
00029         System* getSystem(string name);
00030         vector<Flow*> getFlows();
00031         string getId();
00032         void setId(string);
00033         void add(System*);
00034         void add(Flow*);
00035         void add(Model* model);
00036 };
00037
00038 class ModelHandle : public Handle<ModelBody>, public Model{
00039     public:
00040         ModelHandle() : Handle() {}
00041         virtual ~ModelHandle() {}
00042         static Model* createModel(string id){
00043             ModelHandle* m = new ModelHandle();
00044             m->setId(id);
00045             m->pImpl_->add(m);
00046             return m;
00047         }
00048         double run(int start,int finish){
00049             return pImpl_->run(start, finish);
00050         }
00051         System* createSystem(string name,double value){
00052             return pImpl_->createSystem(name, value);
00053         }
00054         System* getSystem(string name){
00055             return pImpl_->getSystem(name);
00056         }
00057         vector<Flow*> getFlows(){
00058             return pImpl_->getFlows();
00059         }
00060         string getId(){
00061             return pImpl_->getId();
00062         }
00063         void setId(string id){
00064             pImpl_->setId(id);
00065         }
00066         void add(System* system){
00067             pImpl_->add(system);
00068         }
00069         void add(Flow* flow){
00070             pImpl_->add(flow);
00071         }
00072         void add(Model* model) {
00073             pImpl_->add(model);
00074         }
00075 };
00076
00077 #endif
```

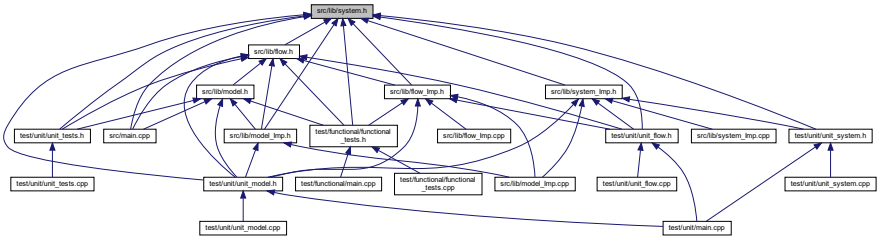
5.15 src/lib/system.h File Reference

```
#include <iostream>
#include <cstdlib>
```

Include dependency graph for system.h:



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



Classes

- class System

  File responsible for project systems.

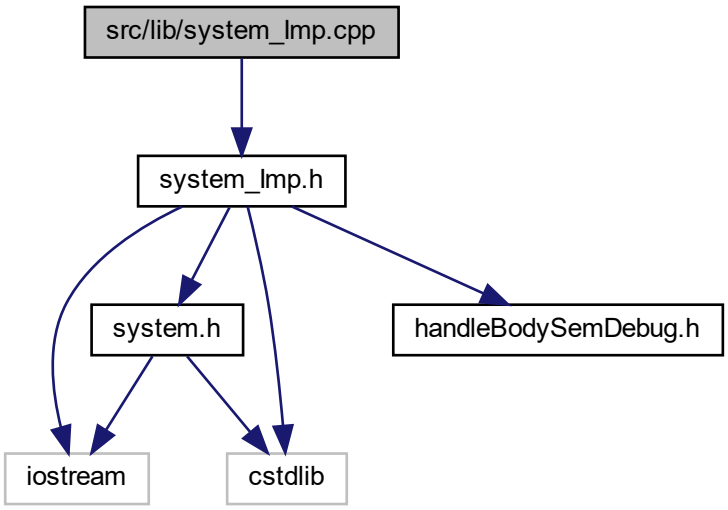
5.16 system.h

Go to the documentation of this file.

```
00001 #ifndef SYSTEM_H
00002 #define SYSTEM_H
00003
00004 #include <iostream>
00005 #include <cstdlib>
00006
00007 using namespace std;
00008
00015 class System{
00016     public:
00021         virtual ~System(){};
00022
00027         virtual void setName(string) = 0;
00028
00033         virtual void setValue(double) = 0;
00034
00040         virtual double getValue() = 0;
00041
00047         virtual string getName() = 0;
00048 };
00049
00050 #endif
```

5.17 src/lib/system_Imp.cpp File Reference

`#include "system_Imp.h"`
Include dependency graph for system_Imp.cpp:



5.18 system_Imp.cpp

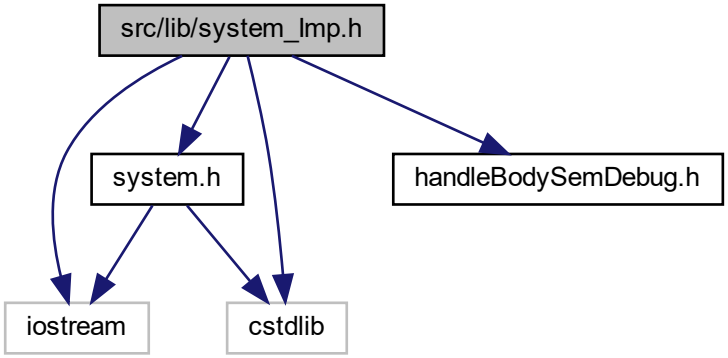
Go to the documentation of this file.

```
00001 #include "system_Imp.h"
00002
00003 SystemBody::SystemBody(){};
00004
00005 SystemBody::~SystemBody(){};
00006
00007 SystemBody::SystemBody(string n, double v){
00008     this->name = n;
00009     this->value = v;
00010 }
00011
00012 void SystemBody::setName(string name){
00013     this->name = name;
00014 }
00015
00016 void SystemBody::setValue(double value){
00017     this->value = value;
00018 }
00019
00020 string SystemBody::getName(){
00021     return this->name;
00022 }
00023
00024 double SystemBody::getValue(){
00025     return this->value;
00026 }
00027
00028 SystemBody* SystemBody::operator=(SystemBody* system){
00029     if (this == system)
00030         return this;
00031     this->name = system->getName();
00032     this->value = system->getValue();
00033     return this;
00034 }
```

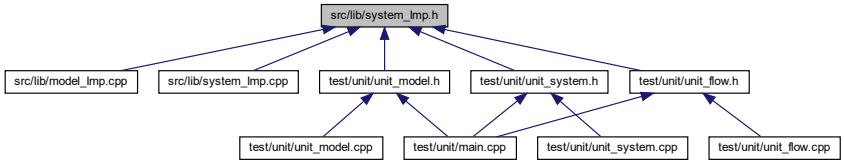
5.19 src/lib/system_Imp.h File Reference

```
#include "system.h"
#include "handleBodySemDebug.h"
#include <iostream>
#include <cstdlib>
```

Include dependency graph for system_Imp.h:



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



Classes

- class SystemBody
- class SystemHandle

5.20 system_Imp.h

Go to the documentation of this file.

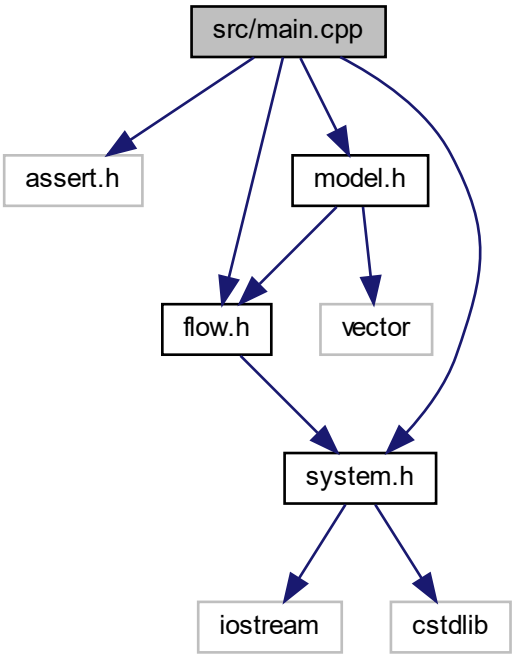
```
00001 #ifndef SYSTEM_IMP_H
00002 #define SYSTEM_IMP_H
00003
00004 #include "system.h"
00005 #include "handleBodySemDebug.h"
00006
00007 #include <iostream>
```

```
00008 #include <cstdlib>
00009
00010 using namespace std;
00011
00012 class SystemBody : public Body{
00013     protected:
00014         string name;
00015         double value;
00016     public:
00017         SystemBody();
00018         SystemBody(string n, double v);
00019         virtual ~SystemBody();
00020         void setName(string);
00021         void setValue(double);
00022         double getValue();
00023         string getName();
00024         SystemBody* operator=(SystemBody* system);
00025 };
00026
00027 class SystemHandle : public Handle<SystemBody>, public System {
00028     public:
00029         SystemHandle() : Handle(){};
00030         SystemHandle(string n, double v) : Handle(){
00031             pImpl_->setName(n);
00032             pImpl_->setValue(v);
00033         }
00034         virtual ~SystemHandle() {};
00035         void setName(string name){
00036             pImpl_->setName(name);
00037         }
00038         void setValue(double value){
00039             pImpl_->setValue(value);
00040         }
00041         double getValue(){
00042             return pImpl_->getValue();
00043         }
00044         string getName(){
00045             return pImpl_->getName();
00046         }
00047 };
00048
00049
00050 #endif
```

5.21 src/main.cpp File Reference

```
#include <assert.h>
#include "model.h"
#include "flow.h"
#include "system.h"
```

Include dependency graph for main.cpp:



Functions

- int main ()

5.21.1 Function Documentation

5.21.1.1 main()

```
int main ( )
```

Definition at line 8 of file main.cpp.

5.22 main.cpp

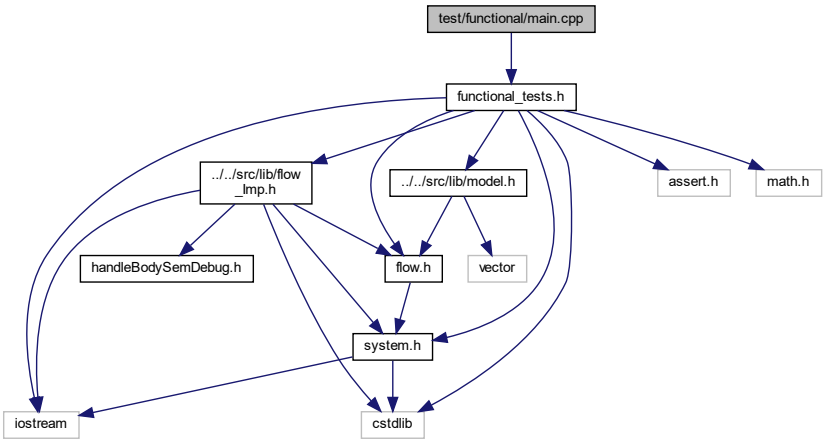
Go to the documentation of this file.

```
00001 #include <assert.h>
00002 #include "model.h"
00003 #include "flow.h"
00004 #include "system.h"
00005
00006 using namespace std;
00007
00008 int main (){  return 0; }
```

5.23 test/functional/main.cpp File Reference

```
#include "functional_tests.h"
```

Include dependency graph for main.cpp:



Macros

- #define DEBUGING

Functions

- int main ()

Variables

- int numHandleCreated = 0
- int numHandleDeleted = 0
- int numBodyCreated = 0
- int numBodyDeleted = 0

5.23.1 Macro Definition Documentation

5.23.1.1 DEBUGING

```
#define DEBUGING
```

Definition at line 5 of file main.cpp.

5.23.2 Function Documentation

5.23.2.1 main()

```
int main ( )
```

Definition at line 13 of file main.cpp.

5.23.3 Variable Documentation

5.23.3.1 numBodyCreated

```
int numBodyCreated = 0
```

Definition at line 9 of file main.cpp.

5.23.3.2 numBodyDeleted

```
int numBodyDeleted = 0
```

Definition at line 10 of file main.cpp.

5.23.3.3 numHandleCreated

```
int numHandleCreated = 0
```

Definition at line 7 of file main.cpp.

5.23.3.4 numHandleDeleted

```
int numHandleDeleted = 0
```

Definition at line 8 of file main.cpp.

5.24 main.cpp

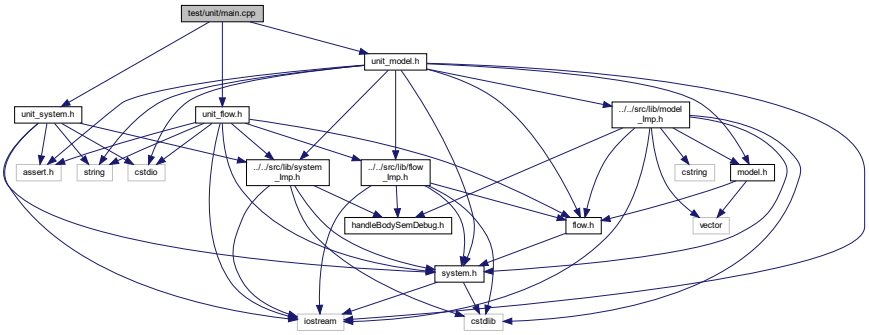
Go to the documentation of this file.

```
00001 #include "functional_tests.h"
00002
00003 using namespace std;
00004
00005 #define DEBUGING
00006 #ifdef DEBUGING
00007     int numHandleCreated = 0;
00008     int numHandleDeleted = 0;
00009     int numBodyCreated = 0;
00010     int numBodyDeleted = 0;
00011 #endif
00012
00013 int main (){
00014
00015     exponentialFuncionalTest();
00016     logisticalFuncionalTest();
00017     complexFuncionalTest();
00018
00019     cout « "RUNNING FUNCTIONAL TESTS" « endl;
00020
00021     printf("-> Created handles: %4d\n", numHandleCreated);
00022     printf("-> Deleted handles: %4d\n", numHandleDeleted);
00023     printf("-> Created bodies: %5d\n", numBodyCreated);
00024     printf("-> Deleted bodies: %5d\n", numBodyDeleted);
00025
00026     cout « "Everything is running!" « endl;
00027
00028     return 0;
00029 }
```

5.25 test/unit/main.cpp File Reference

```
#include "unit_system.h"
#include "unit_flow.h"
#include "unit_model.h"
```

Include dependency graph for main.cpp:



Macros

- #define DEBUGING

Functions

- int main ()

Variables

- int numHandleCreated = 0
- int numHandleDeleted = 0
- int numBodyCreated = 0
- int numBodyDeleted = 0

5.25.1 Macro Definition Documentation

5.25.1.1 DEBUGING

```
#define DEBUGING
```

Definition at line 6 of file main.cpp.

5.25.2 Function Documentation

5.25.2.1 main()

```
int main ( )
```

Definition at line 16 of file main.cpp.

5.25.3 Variable Documentation

5.25.3.1 numBodyCreated

```
int numBodyCreated = 0
```

Definition at line 10 of file main.cpp.

5.25.3.2 numBodyDeleted

```
int numBodyDeleted = 0
```

Definition at line 11 of file main.cpp.

5.25.3.3 numHandleCreated

```
int numHandleCreated = 0
```

Definition at line 8 of file main.cpp.

5.25.3.4 numHandleDeleted

```
int numHandleDeleted = 0
```

Definition at line 9 of file main.cpp.

5.26 main.cpp

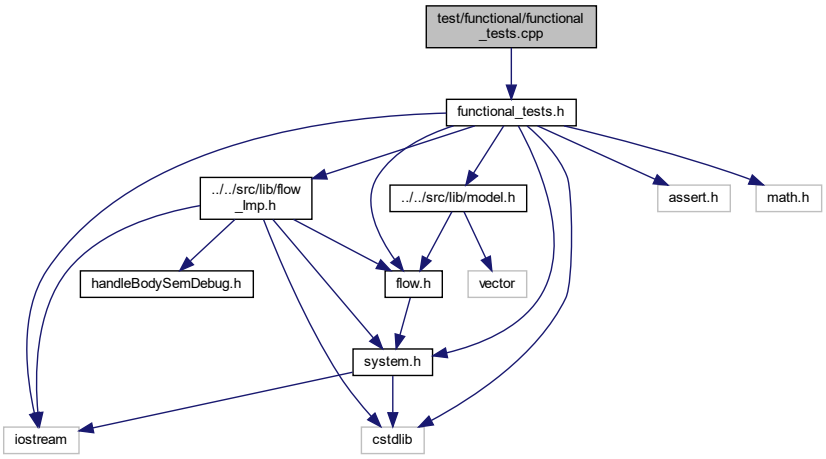
Go to the documentation of this file.

```
00001 //#include "unit_tests.h"
00002 #include "unit_system.h"
00003 #include "unit_flow.h"
00004 #include "unit_model.h"
00005
00006 #define DEBUGING
00007 #ifdef DEBUGING
00008     int numHandleCreated = 0;
00009     int numHandleDeleted = 0;
00010     int numBodyCreated = 0;
00011     int numBodyDeleted = 0;
00012 #endif
00013
00014 using namespace std;
00015
00016 int main (){
00017
00018     run_unit_test_System();
00019     run_unit_test_Flow();
00020     run_unit_test_Model();
00021
00022     cout « "RUNNING UNIT TESTS" « endl;
00023
00024     printf("-> Created handles: %4d\n", numHandleCreated);
00025     printf("-> Deleted handles: %4d\n", numHandleDeleted);
00026     printf("-> Created bodies: %5d\n", numBodyCreated);
00027     printf("-> Deleted bodies: %5d\n", numBodyDeleted);
00028
00029     cout « "All unit tests passed!" « endl;
00030
00031     return 0;
00032 }
```

5.27 test/functional/functional_tests.cpp File Reference

```
#include "functional_tests.h"
```

Include dependency graph for functional_tests.cpp:



Functions

- void exponentialFuncionalTest ()

  Exponential functional test.
- void logisticalFuncionalTest ()

  Logistics functional test.
- void complexFuncionalTest ()

  Complex functional test.

5.27.1 Function Documentation

5.27.1.1 complexFuncionalTest()

```
void complexFuncionalTest ( )
```

Complex functional test.

Definition at line 29 of file functional_tests.cpp.

5.27.1.2 exponentialFuncionalTest()

```
void exponentialFuncionalTest ( )
```

Exponential functional test.

Definition at line 3 of file functional_tests.cpp.

5.27.1.3 logisticalFuncionalTest()

```
void logisticalFuncionalTest ( )
```

Logistics functional test.

Definition at line 15 of file functional_tests.cpp.

5.28 functional_tests.cpp

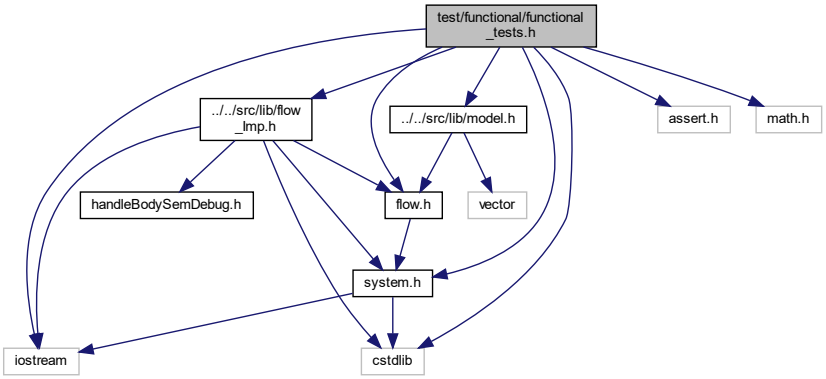
Go to the documentation of this file.

```
00001 #include "functional_tests.h"
00002
00003 void exponentialFuncionalTest(){
00004     Model* Modelexponential = Model::createModel("Model pops");
00005     System* pop1 =  Modelexponential->createSystem("pop1", 100.0);
00006     System* pop2 =  Modelexponential->createSystem("pop2", 0.0);
00007     Flow* f = Modelexponential->createFlow<EXPONENTIAL>(pop1,pop2);
00008     Modelexponential->run(0,100);
00009     assert(abs(pop1->getValue() - 36.6032) < 0.0001);
00010     assert(abs(pop2->getValue() - 63.3968) < 0.0001);
00011
00012     delete Modelexponential;
00013 }
00014
00015 void logisticalFuncionalTest(){
00016     Model* ModelLogistic = Model::createModel("Model Logistic");
00017     System* p1 = ModelLogistic->createSystem("p1", 100.0);
00018     System* p2 = ModelLogistic->createSystem("p2", 10.0);
00019     Flow* l = ModelLogistic->createFlow<LOGISTIC>(p1,p2);
00020     ModelLogistic->run(0,100);
00021
00022     assert(abs(p1->getValue() - 88.2167) < 0.0001);
00023     assert(abs(p2->getValue() - 21.7834) < 0.0001);
00024
00025     delete ModelLogistic;
00026 }
00027
00028
00029 void complexFuncionalTest(){
00030     Model* model = Model::createModel("Model Complex");
00031     System* q1 =  model->createSystem("q1", 100.0);
00032     System* q2 =  model->createSystem("q2", 0.0);
00033     System* q3 =  model->createSystem("q3", 100.0);
00034     System* q4 =  model->createSystem("q4", 0.0);
00035     System* q5 =  model->createSystem("q5", 0.0);
00036     Flow* f = model->createFlow<EXPONENTIAL>(q1,q2);
00037     Flow* g = model->createFlow<EXPONENTIAL>(q1,q3);
00038     Flow* r = model->createFlow<EXPONENTIAL>(q2,q5);
00039     Flow* t = model->createFlow<EXPONENTIAL>(q2,q3);
00040     Flow* u = model->createFlow<EXPONENTIAL>(q3,q4);
00041     Flow* v = model->createFlow<EXPONENTIAL>(q4,q1);
00042
00043     model->run(0,100);
00044
00045     assert(abs((q1->getValue() - 31.8513)) < 0.0001);
00046     assert(abs((q2->getValue() - 18.4003)) < 0.0001);
00047     assert(abs((q3->getValue() - 77.1143)) < 0.0001);
00048     assert(abs((q4->getValue() - 56.1728)) < 0.0001);
00049     assert(abs((q5->getValue() - 16.4612)) < 0.0001);
00050
00051     delete model;
00052 }
```

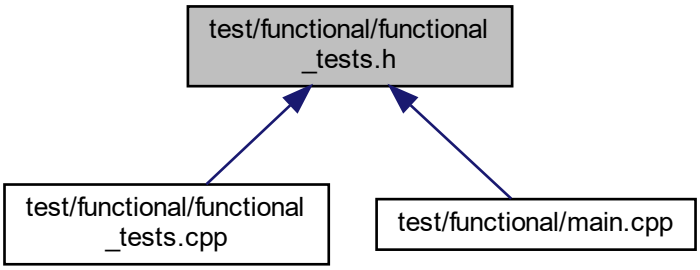
5.29 test/functional/functional_tests.h File Reference

```
#include "../../src/lib/model.h"
#include "../../src/lib/flow.h"
#include "../../src/lib/system.h"
#include "../../src/lib/flow_Imp.h"
#include <iostream>
#include <cstdlib>
#include <assert.h>
#include <math.h>
```

Include dependency graph for functional_tests.h:



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



Classes

- class Exponential
- class Logistic

Macros

- #define EXPONENTIAL FlowHandle<Exponential>
- #define LOGISTIC FlowHandle<Logistic>

Functions

- void exponentialFuncionalTest ()

  Exponential functional test.
- void logisticalFuncionalTest ()

  Logistics functional test.
- void complexFuncionalTest ()

  Complex functional test.

5.29.1 Macro Definition Documentation

5.29.1.1 EXPONENTIAL

```
#define EXPONENTIAL FlowHandle<Exponential>
```

Definition at line 8 of file functional_tests.h.

5.29.1.2 LOGISTIC

```
#define LOGISTIC FlowHandle<Logistic>
```

Definition at line 9 of file functional_tests.h.

5.29.2 Function Documentation

5.29.2.1 complexFuncionalTest()

```
void complexFuncionalTest ( )
```

Complex functional test.

Definition at line 29 of file functional_tests.cpp.

5.29.2.2 exponentialFuncionalTest()

```
void exponentialFuncionalTest ( )
```

Exponential functional test.

Definition at line 3 of file functional_tests.cpp.

5.29.2.3 logisticalFuncionalTest()

```
void logisticalFuncionalTest ( )
```

Logistics functional test.

Definition at line 15 of file functional_tests.cpp.

5.30 functional_tests.h

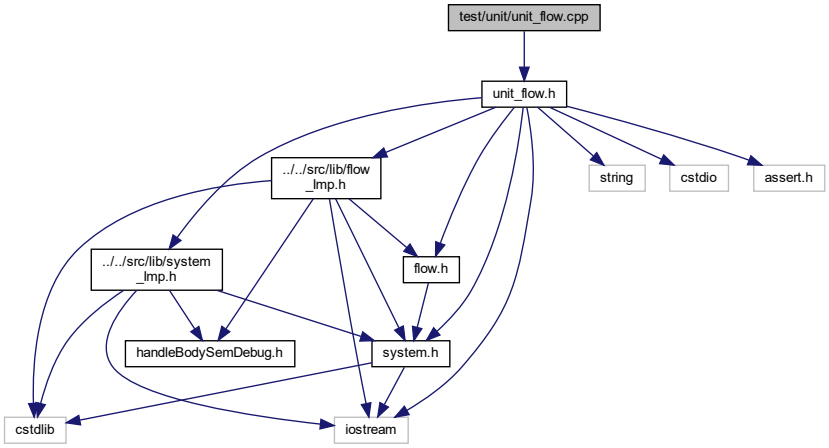
Go to the documentation of this file.

```
00001 #ifndef FUNCTIONAL_TESTS_H
00002 #define FUNCTIONAL_TESTS_H
00003 #include "../../src/lib/model.h"
00004 #include "../../src/lib/flow.h"
00005 #include "../../src/lib/system.h"
00006 #include "../../src/lib/flow_Imp.h"
00007
00008 #define EXPONENTIAL FlowHandle<Exponential>
00009 #define LOGISTIC  FlowHandle<Logistic>
00010
00011 #include <iostream>
00012 #include <cstdlib>
00013 #include <assert.h>
00014 #include <math.h>
00015
00016
00022 using namespace std;
00023
00028     void exponentialFuncionalTest();
00029
00034     void logisticalFuncionalTest();
00035
00040     void complexFuncionalTest();
00041
00042 class Exponential: public FlowBody {
00043     public:
00044         Exponential(){}
00045         ~Exponential() {}
00046         double run(){
00047             return getSource()->getValue()*0.01;
00048         }
00049 };
00050
00051 class Logistic: public FlowBody{
00052     public:
00053         Logistic(){}
00054         ~Logistic() {}
00055         double run(){
00056             return getDestination()->getValue()*0.01*(1-(getDestination()->getValue())/70);
00057         }
00058 };
00059 #endif
```

5.31 test/unit/unit_flow.cpp File Reference

```
#include "unit_flow.h"
```

Include dependency graph for unit_flow.cpp:



Functions

- void unit_Flow_constructor (void)
- void unit_Flow_destructor (void)
- void unit_Flow_setSource (void)
- void unit_Flow_setDestination (void)
- void unit_Flow_getSource (void)
- void unit_Flow_getDestination (void)
- void unit_Flow_operator (void)
- void run_unit_test_Flow (void)

5.31.1 Function Documentation

5.31.1.1 run_unit_test_Flow()

```
void run_unit_test_Flow (
            void  )
```

Definition at line 57 of file unit_flow.cpp.

5.31.1.2 unit_Flow_constructor()

```
void unit_Flow_constructor (
            void  )
```

Definition at line 3 of file unit_flow.cpp.

5.31.1.3 unit_Flow_destructor()

```
void unit_Flow_destructor (
            void  )
```

Definition at line 9 of file unit_flow.cpp.

5.31.1.4 unit_Flow_getDestination()

```
void unit_Flow_getDestination (
            void  )
```

Definition at line 38 of file unit_flow.cpp.

5.31.1.5 unit_Flow_getSource()

```
void unit_Flow_getSource (
            void  )
```

Definition at line 29 of file unit_flow.cpp.

5.31.1.6 unit_Flow_operator()

```
void unit_Flow_operator (
            void  )
```

Definition at line 47 of file unit_flow.cpp.

5.31.1.7 unit_Flow_setDestination()

```
void unit_Flow_setDestination (
            void  )
```

Definition at line 20 of file unit_flow.cpp.

5.31.1.8 unit_Flow_setSource()

```
void unit_Flow_setSource (
            void  )
```

Definition at line 11 of file unit_flow.cpp.

5.32 unit_flow.cpp

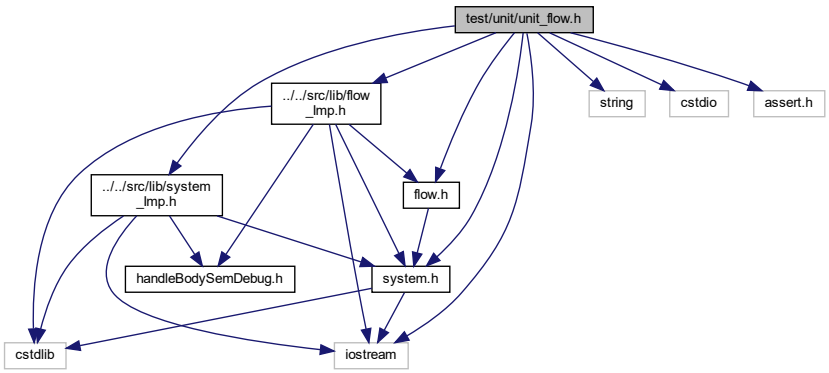
Go to the documentation of this file.

```
00001 #include "unit_flow.h"
00002
00003 void unit_Flow_constructor(void){
00004     Flow* f = new FlowHandle<FlowUnit>();
00005     assert(f->getSource() == nullptr);
00006     delete f;
00007 }
00008
00009 void unit_Flow_destructor(void){}
00010
00011 void unit_Flow_setSource(void){
00012     Flow* f = new FlowHandle<FlowUnit>();
00013     System* source = new SystemHandle();
00014     f->setSources(source);
00015     assert(f->getSource() == source);
00016     delete f;
00017     delete source;
00018 }
00019
00020 void unit_Flow_setDestination(void){
00021     Flow* f = new FlowHandle<FlowUnit>();
00022     System* destination = new SystemHandle();
00023     f->setDestination(destination);
00024     assert(f->getDestination() == destination);
00025     delete f;
00026     delete destination;
00027 }
00028
00029 void unit_Flow_getSource(void){
00030     System* s1 = new SystemHandle();
00031     Flow* f = new FlowHandle<FlowUnit>();
00032     f->setSources(s1);
00033     assert(f->getSource() == s1);
00034     delete s1;
00035     delete f;
00036 }
00037
00038 void unit_Flow_getDestination(void){
00039     System* s1 = new SystemHandle();
00040     Flow* f = new FlowHandle<FlowUnit>();
00041     f->setDestination(s1);
00042     assert(f->getDestination() == s1);
00043     delete s1;
00044     delete f;
00045 }
00046
00047 void unit_Flow_operator(void){
00048     System* s1 = new SystemHandle();
00049     Flow* f = new FlowHandle<FlowUnit>();
00050     f->setDestination(s1);
00051     Flow* test = f;
00052     assert(f->getDestination() == test->getDestination());
00053     delete s1;
00054     delete f;
00055 }
00056
00057 void run_unit_test_Flow(void){
00058     unit_Flow_constructor();
00059     unit_Flow_destructor();
00060     unit_Flow_setSource();
00061     unit_Flow_setDestination();
00062     unit_Flow_getSource();
00063     unit_Flow_getDestination();
00064     unit_Flow_operator();
00065 }
```

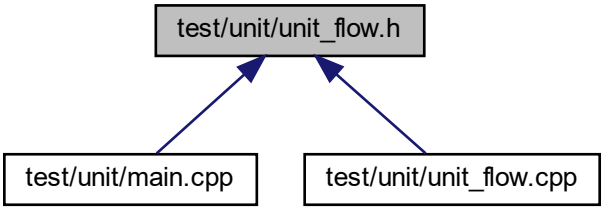
5.33 test/unit/unit_flow.h File Reference

```
#include "../../src/lib/system_Imp.h"
#include "../../src/lib/system.h"
#include "../../src/lib/flow_Imp.h"
#include "../../src/lib/flow.h"
#include <string>
#include <cstdio>
#include <iostream>
#include <assert.h>
```

Include dependency graph for unit_flow.h:



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



Classes

- class FlowUnit

Functions

- void unit_Flow_constructor (void)
- void unit_Flow_destructor (void)

- void unit_Flow_setSource (void)
- void unit_Flow_setDestination (void)
- void unit_Flow_getSource (void)
- void unit_Flow_getDestination (void)
- void unit_Flow_operator (void)
- void run_unit_test_Flow (void)

5.33.1 Function Documentation

5.33.1.1 run_unit_test_Flow()

```
void run_unit_test_Flow (
             void  )
```

Definition at line 57 of file unit_flow.cpp.

5.33.1.2 unit_Flow_constructor()

```
void unit_Flow_constructor (
             void  )
```

Definition at line 3 of file unit_flow.cpp.

5.33.1.3 unit_Flow_destructor()

```
void unit_Flow_destructor (
             void  )
```

Definition at line 9 of file unit_flow.cpp.

5.33.1.4 unit_Flow_getDestination()

```
void unit_Flow_getDestination (
             void  )
```

Definition at line 38 of file unit_flow.cpp.

5.33.1.5 unit_Flow_getSource()

```
void unit_Flow_getSource (
            void  )
```

Definition at line 29 of file unit_flow.cpp.

5.33.1.6 unit_Flow_operator()

```
void unit_Flow_operator (
            void  )
```

Definition at line 47 of file unit_flow.cpp.

5.33.1.7 unit_Flow_setDestination()

```
void unit_Flow_setDestination (
            void  )
```

Definition at line 20 of file unit_flow.cpp.

5.33.1.8 unit_Flow_setSource()

```
void unit_Flow_setSource (
            void  )
```

Definition at line 11 of file unit_flow.cpp.

5.34 unit_flow.h

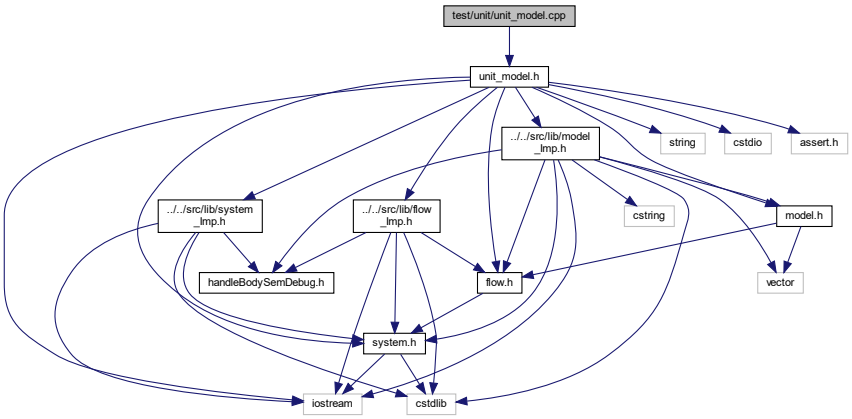
Go to the documentation of this file.

```
00001 #ifndef UNIT_FLOW_H
00002 #define UNIT_FLOW_H
00003
00004 #include "../../src/lib/system_Imp.h"
00005 #include "../../src/lib/system.h"
00006 #include "../../src/lib/flow_Imp.h"
00007 #include "../../src/lib/flow.h"
00008
00009 #include <string>
00010 #include <cstdio>
00011 #include <iostream>
00012 #include <assert.h>
00013
00014 using namespace std;
00015
00016 void unit_Flow_constructor(void);
00017 void unit_Flow_destructor(void);
00018 void unit_Flow_setSource(void);
00019 void unit_Flow_setDestination(void);
00020 void unit_Flow_getSource(void);
00021 void unit_Flow_getDestination(void);
00022 void unit_Flow_operator(void);
00023 void run_unit_test_Flow(void);
00024
00025 class FlowUnit : public FlowBody{
00026     public:
00027         FlowUnit(): FlowBody() {}
00028         ~FlowUnit() {}
00029         double run(){
00030             return 0;
00031         }
00032 };
00033
00034 #endif
```

5.35 test/unit/unit_model.cpp File Reference

```
#include "unit_model.h"
```

Include dependency graph for unit_model.cpp:



Functions

- void unit_Model_constructor (void)
- void unit_Model_destructor (void)
- void unit_Model_run (void)
- void unit_Model_add_System (void)
- void unit_Model_add_Flow (void)
- void run_unit_test_Model ()

5.35.1 Function Documentation

5.35.1.1 run_unit_test_Model()

```
void run_unit_test_Model (
            void  )
```

Definition at line 51 of file unit_model.cpp.

5.35.1.2 unit_Model_add_Flow()

```
void unit_Model_add_Flow (
            void  )
```

Definition at line 39 of file unit_model.cpp.

5.35.1.3 unit_Model_add_System()

```
void unit_Model_add_System (
            void  )
```

Definition at line 30 of file unit_model.cpp.

5.35.1.4 unit_Model_constructor()

```
void unit_Model_constructor (
            void  )
```

Definition at line 3 of file unit_model.cpp.

5.35.1.5 unit_Model_destructor()

```
void unit_Model_destructor (
            void  )
```

Definition at line 11 of file unit_model.cpp.

5.35.1.6 unit_Model_run()

```
void unit_Model_run (
            void  )
```

Definition at line 13 of file unit_model.cpp.

5.36 unit_model.cpp

Go to the documentation of this file.

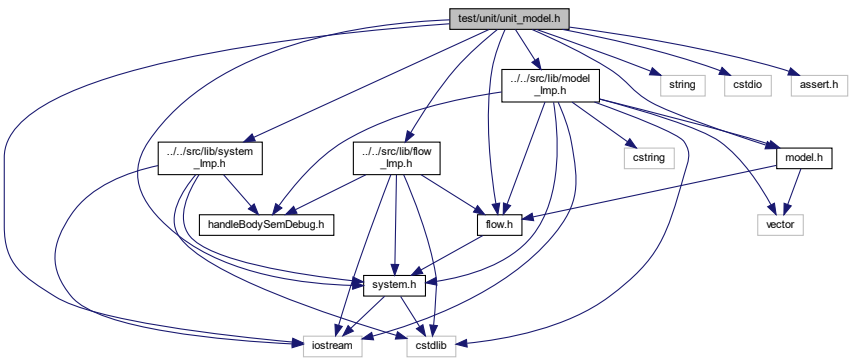
```
00001 #include "unit_model.h"
00002
00003 void unit_Model_constructor(void){
00004     Model* m = Model::createModel("test model");
00005     System* s;
00006     s = m->createSystem("system",10.0 );
00007     assert(m->getSystem("system") == s);
00008     delete m;
00009 }
00010
00011 void unit_Model_destructor(void){}
00012
00013 void unit_Model_run(void){
00014     Model* Modelexponential = Model::createModel("Model pops");
00015     System* pop1;
00016     System* pop2;
00017     Flow* f;
00018     pop1 =  Modelexponential->createSystem("pop1", 100.0);
00019     pop2 =  Modelexponential->createSystem("pop2", 0.0);
00020     f = Modelexponential->createFlow<EXPONENTIAL>();
```

```
00021     f->setSources(pop1);
00022     f->setDestination(pop2);
00023     Modelexponential->run(0,100);
00024     assert(abs(pop1->getValue() - 36.6032) < 0.0001);
00025     assert(abs(pop2->getValue() - 63.3968) < 0.0001);
00026
00027     delete Modelexponential;
00028 }
00029
00030 void unit_Model_add_System(void){
00031     Model* m = Model::createModel("test add");
00032     System* s;
00033     s = m->createSystem("testSystem",0);
00034     assert(m->getSystem("testSystem") == s);
00035
00036     delete m;
00037 }
00038
00039 void unit_Model_add_Flow(void){
00040     Model* m = Model::createModel("test add");
00041     System* s;
00042     System* s2;
00043     Flow* f = m->createFlow<LOGISTIC>();
00044     f->setSources(s);
00045     f->setDestination(s2);
00046     assert(m->getFlows().empty() == 0);
00047
00048     delete m;
00049 }
00050
00051 void run_unit_test_Model(){
00052     unit_Model_constructor();
00053     unit_Model_destructor();
00054     unit_Model_run();
00055     unit_Model_add_System();
00056     unit_Model_add_Flow();
00057 }
```

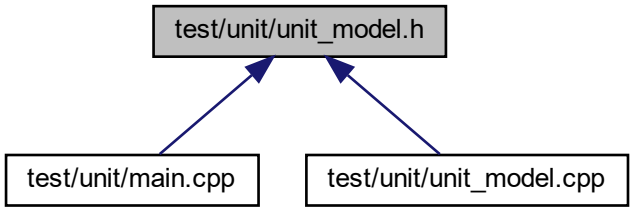
5.37 test/unit/unit_model.h File Reference

```
#include "../../src/lib/system_Imp.h"
#include "../../src/lib/system.h"
#include "../../src/lib/flow_Imp.h"
#include "../../src/lib/flow.h"
#include "../../src/lib/model_Imp.h"
#include "../../src/lib/model.h"
#include <string>
#include <cstdio>
#include <iostream>
#include <assert.h>
```

Include dependency graph for unit_model.h:



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



Classes

- class Exponential
- class Logistic

Macros

- #define EXPONENTIAL FlowHandle<Exponential>
- #define LOGISTIC FlowHandle<Logistic>

Functions

- void unit_Model_constructor (void)
- void unit_Model_destructor (void)
- void unit_Model_run (void)
- void unit_Model_add_System (void)
- void unit_Model_add_Flow (void)
- void run_unit_test_Model (void)

5.37.1 Macro Definition Documentation

5.37.1.1 EXPONENTIAL

```
#define EXPONENTIAL FlowHandle<Exponential>
```

Definition at line 11 of file unit_model.h.

5.37.1.2 LOGISTIC

```
#define LOGISTIC FlowHandle<Logistic>
```

Definition at line 12 of file unit_model.h.

5.37.2 Function Documentation

5.37.2.1 run_unit_test_Model()

```
void run_unit_test_Model (
            void  )
```

Definition at line 51 of file unit_model.cpp.

5.37.2.2 unit_Model_add_Flow()

```
void unit_Model_add_Flow (
            void  )
```

Definition at line 39 of file unit_model.cpp.

5.37.2.3 unit_Model_add_System()

```
void unit_Model_add_System (
            void  )
```

Definition at line 30 of file unit_model.cpp.

5.37.2.4 unit_Model_constructor()

```
void unit_Model_constructor (
            void  )
```

Definition at line 3 of file unit_model.cpp.

5.37.2.5 unit_Model_destructor()

```
void unit_Model_destructor (
            void  )
```

Definition at line 11 of file unit_model.cpp.

5.37.2.6 unit_Model_run()

```
void unit_Model_run (
            void  )
```

Definition at line 13 of file unit_model.cpp.

5.38 unit_model.h

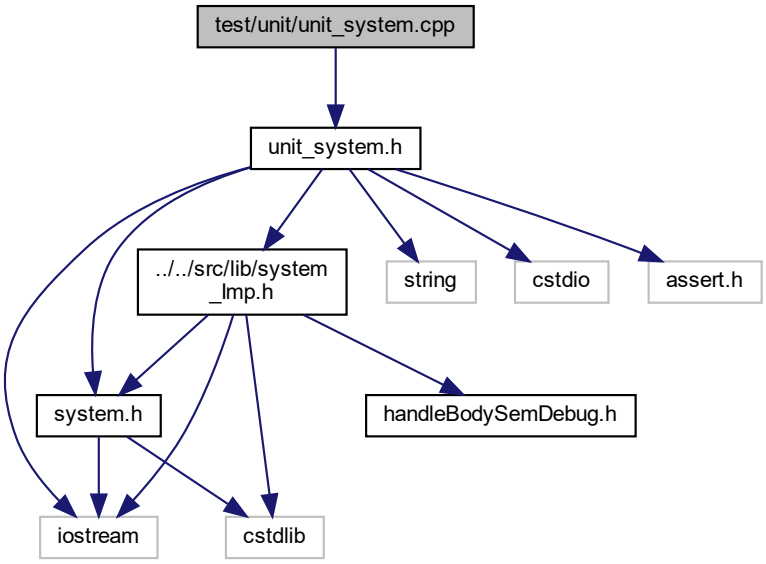
Go to the documentation of this file.

```
00001 #ifndef UNIT_MODEL_H
00002 #define UNIT_MODEL_H
00003
00004 #include "../../src/lib/system_Imp.h"
00005 #include "../../src/lib/system.h"
00006 #include "../../src/lib/flow_Imp.h"
00007 #include "../../src/lib/flow.h"
00008 #include "../../src/lib/model_Imp.h"
00009 #include "../../src/lib/model.h"
00010
00011 #define EXPONENTIAL FlowHandle<Exponential>
00012 #define LOGISTIC  FlowHandle<Logistic>
00013
00014 #include <string>
00015 #include <cstdio>
00016 #include <iostream>
00017 #include <assert.h>
00018
00019 using namespace std;
00020
00021 void unit_Model_constructor(void);
00022 void unit_Model_destructor(void);
00023 void unit_Model_run(void);
00024 void unit_Model_add_System(void);
00025 void unit_Model_add_Flow(void);
00026 void run_unit_test_Model(void);
00027
00028 class Exponential: public FlowBody {
00029     public:
00030         Exponential(){}
00031         ~Exponential() {}
00032         double run(){
00033             return getSource()->getValue()*0.01;
00034         }
00035 };
00036
00037 class Logistic: public FlowBody{
00038     public:
00039         Logistic(){}
00040         ~Logistic() {}
00041         double run(){
00042             return getDestination()->getValue()*0.01*(1-(getDestination()->getValue())/70);
00043         }
00044 };
00045 #endif
```

5.39 test/unit/unit_system.cpp File Reference

```
#include "unit_system.h"
```

Include dependency graph for unit_system.cpp:



Functions

- void unit_System_constructor (void)
- void unit_System_destructor (void)
- void unit_System_setName (void)
- void unit_System_setValue (void)
- void unit_System_getName (void)
- void unit_System_getValue (void)
- void unit_System_operator (void)
- void run_unit_test_System (void)

5.39.1 Function Documentation

5.39.1.1 run_unit_test_System()

```
void run_unit_test_System (
            void  )
```

Definition at line 50 of file unit_system.cpp.

5.39.1.2 unit_System_constructor()

```
void unit_System_constructor (
            void  )
```

Definition at line 3 of file unit_system.cpp.

5.39.1.3 unit_System_destructor()

```
void unit_System_destructor (
            void  )
```

Definition at line 10 of file unit_system.cpp.

5.39.1.4 unit_System_getName()

```
void unit_System_getName (
            void  )
```

Definition at line 28 of file unit_system.cpp.

5.39.1.5 unit_System_getValue()

```
void unit_System_getValue (
            void  )
```

Definition at line 35 of file unit_system.cpp.

5.39.1.6 unit_System_operator()

```
void unit_System_operator (
            void  )
```

Definition at line 42 of file unit_system.cpp.

5.39.1.7 unit_System_setName()

```
void unit_System_setName (
            void  )
```

Definition at line 12 of file unit_system.cpp.

5.39.1.8 unit_System_setValue()

```
void unit_System_setValue (
            void  )
```

Definition at line 20 of file unit_system.cpp.

5.40 unit_system.cpp

Go to the documentation of this file.

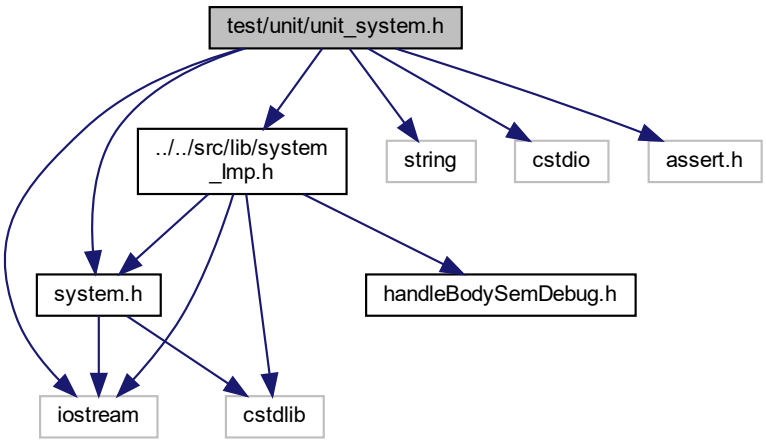
```
00001 #include "unit_system.h"
00002
00003 void unit_System_constructor(void){
00004     System* s1 = new SystemHandle();
00005     s1->setValue(0);
00006     assert(s1->getValue() == 0);
00007     delete s1;
00008 }
00009
00010 void unit_System_destructor(void){}
00011
00012 void unit_System_setName(void){
00013     System* s1 = new SystemHandle();
00014     s1->setName("test");
00015     assert(s1->getName() == "test");
00016
00017     delete s1;
00018 }
00019
00020 void unit_System_setValue(void){
00021     System* s1 = new SystemHandle();
00022     s1->setValue(10);
00023     assert(s1->getValue() == 10);
00024
00025     delete s1;
00026 }
00027
00028 void unit_System_getName(void){
00029     System* s1 = new SystemHandle("test", 10);
00030     assert(s1->getName() == "test" );
00031
00032     delete s1;
00033 }
00034
00035 void unit_System_getValue(void){
00036     System* s1 = new SystemHandle("test", 10);
00037     assert(s1->getValue() == 10 );
00038
00039     delete s1;
00040 }
00041
00042 void unit_System_operator(void){
00043     System* s1 = new SystemHandle("test", 10);
00044     System* s2 = s1;
00045     assert(s1->getValue() == s2->getValue());
00046
00047     delete s1, s2;
00048 }
00049
00050 void run_unit_test_System(void){
00051     unit_System_constructor();
00052     unit_System_destructor();
00053     unit_System_setName();
00054     unit_System_setValue();
00055     unit_System_getName();
00056     unit_System_getValue();
00057     unit_System_operator();
00058 }
```

5.41 test/unit/unit_system.h File Reference

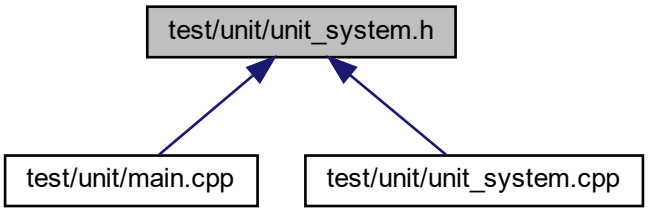
```
#include "../../src/lib/system_Imp.h"
#include "../../src/lib/system.h"
```

```
#include <string>
#include <cstdio>
#include <iostream>
#include <assert.h>
```

Include dependency graph for unit_system.h:



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



Functions

- void unit_System_constructor (void)
- void unit_System_destructor (void)
- void unit_System_setName (void)
- void unit_System_setValue (void)
- void unit_System_getName (void)
- void unit_System_getValue (void)
- void unit_System_operator (void)
- void run_unit_test_System (void)

5.41.1 Function Documentation

5.41.1.1 run_unit_test_System()

```
void run_unit_test_System (
            void  )
```

Definition at line 50 of file unit_system.cpp.

5.41.1.2 unit_System_constructor()

```
void unit_System_constructor (
            void  )
```

Definition at line 3 of file unit_system.cpp.

5.41.1.3 unit_System_destructor()

```
void unit_System_destructor (
            void  )
```

Definition at line 10 of file unit_system.cpp.

5.41.1.4 unit_System_getName()

```
void unit_System_getName (
            void  )
```

Definition at line 28 of file unit_system.cpp.

5.41.1.5 unit_System_getValue()

```
void unit_System_getValue (
            void  )
```

Definition at line 35 of file unit_system.cpp.

5.41.1.6 unit_System_operator()

```
void unit_System_operator (
            void  )
```

Definition at line 42 of file unit_system.cpp.

5.41.1.7 unit_System_setName()

```
void unit_System_setName (
            void  )
```

Definition at line 12 of file unit_system.cpp.

5.41.1.8 unit_System_setValue()

```
void unit_System_setValue (
            void  )
```

Definition at line 20 of file unit_system.cpp.

5.42 unit_system.h

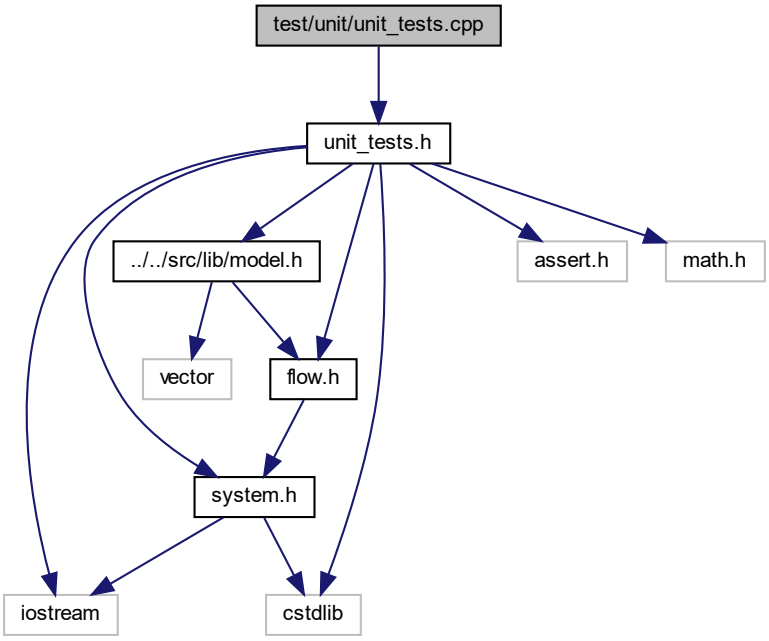
Go to the documentation of this file.

```
00001 #ifndef UNIT_SYSTEM_H
00002 #define UNIT_SYSTEM_H
00003
00004 #include "../../src/lib/system_Imp.h"
00005 #include "../../src/lib/system.h"
00006
00007 #include <string>
00008 #include <cstdio>
00009 #include <iostream>
00010 #include <assert.h>
00011
00012 using namespace std;
00013
00014 void unit_System_constructor(void);
00015 void unit_System_destructor(void);
00016 void unit_System_setName(void);
00017 void unit_System_setValue(void);
00018 void unit_System_getName(void);
00019 void unit_System_getValue(void);
00020 void unit_System_operator(void);
00021 void run_unit_test_System(void);
00022
00023 #endif
```

5.43 test/unit/unit_tests.cpp File Reference

```
#include "unit_tests.h"
```

Include dependency graph for unit_tests.cpp:



5.44 unit_tests.cpp

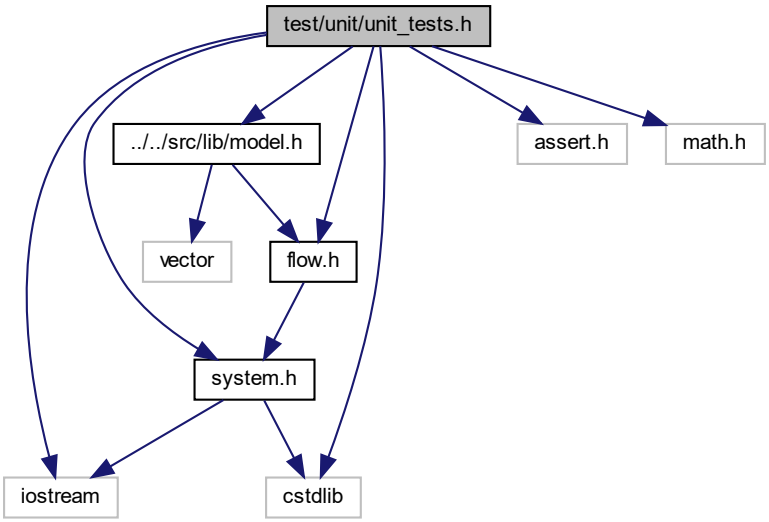
Go to the documentation of this file.

```
00001 #include "unit_tests.h"
00002
```

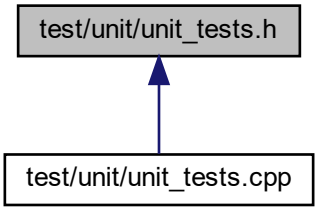
5.45 test/unit/unit_tests.h File Reference

```
#include "../../src/lib/model.h"
#include "../../src/lib/flow.h"
#include "../../src/lib/system.h"
#include <iostream>
#include <cstdlib>
#include <assert.h>
```

`#include <math.h>`
Include dependency graph for unit_tests.h:



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



5.46 unit_tests.h

Go to the documentation of this file.

```
00001 #ifndef UNIT_TESTS_H
00002 #define UNIT_TESTS_H
00003 #include "../../src/lib/model.h"
00004 #include "../../src/lib/flow.h"
00005 #include "../../src/lib/system.h"
00006
00007 #include <iostream>
00008 #include <cstdlib>
00009 #include <assert.h>
00010 #include <math.h>
00011
00012 using namespace std;
00013
00014
00015 #endif
```

Index