

MissionControlStarBase

Generated by Doxygen 1.8.17

1 Todo List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 Class Documentation	7
4.1 ActiveLaunchS Class Reference	7
4.1.1 Member Function Documentation	8
4.1.1.1 changeState()	8
4.1.1.2 handle()	9
4.2 Aggregate Class Reference	9
4.2.1 Member Function Documentation	10
4.2.1.1 add()	10
4.2.1.2 createIterator()	10
4.2.1.3 remove()	11
4.3 Booster Class Reference	11
4.3.1 Detailed Description	14
4.3.2 Constructor & Destructor Documentation	14
4.3.2.1 ~Booster()	14
4.3.3 Member Function Documentation	14
4.3.3.1 add()	14
4.3.3.2 clone()	15
4.3.3.3 getBoosterId()	15
4.3.3.4 getChildBooster()	15
4.3.3.5 getLOXfuelLevel()	16
4.3.3.6 getName()	16
4.3.3.7 getRP1fuelLevel()	16
4.3.3.8 remove()	16
4.3.3.9 setBoosterId()	17
4.3.3.10 setLOXfuelLevel()	17
4.3.3.11 setRP1fuelLevel()	17
4.4 Builder Class Reference	18
4.4.1 Detailed Description	20
4.4.2 Member Function Documentation	20
4.4.2.1 buildRocket()	20
4.4.2.2 getRocketType()	20
4.4.2.3 setFirstStageBoosters()	20
4.4.2.4 setPayloadType()	20
4.4.2.5 setRocketType()	21
4.4.2.6 setSecondStage()	21

4.5 BuildS Class Reference	22
4.5.1 Member Function Documentation	23
4.5.1.1 changeState()	23
4.5.1.2 handle()	23
4.6 Caretaker Class Reference	24
4.6.1 Member Function Documentation	24
4.6.1.1 batchStore()	24
4.6.1.2 createIterator()	25
4.6.1.3 RestoreLast()	25
4.6.1.4 storeRocket()	25
4.7 ConcreteBuilder Class Reference	26
4.7.1 Detailed Description	28
4.7.2 Constructor & Destructor Documentation	28
4.7.2.1 ConcreteBuilder()	28
4.7.3 Member Function Documentation	28
4.7.3.1 buildRocket()	28
4.7.3.2 setFirstStageBoosters()	29
4.7.3.3 setPayload() [1/3]	29
4.7.3.4 setPayload() [2/3]	29
4.7.3.5 setPayload() [3/3]	30
4.7.3.6 setSecondStage()	30
4.8 Dragon Class Reference	31
4.8.1 Constructor & Destructor Documentation	32
4.8.1.1 Dragon() [1/2]	33
4.8.1.2 Dragon() [2/2]	33
4.8.2 Member Function Documentation	33
4.8.2.1 clone()	33
4.8.2.2 getPayloadDescription()	33
4.8.2.3 launchPayload()	34
4.8.2.4 printPayload()	34
4.9 DragonCrew Class Reference	34
4.9.1 Constructor & Destructor Documentation	35
4.9.1.1 DragonCrew() [1/2]	36
4.9.1.2 DragonCrew() [2/2]	36
4.9.2 Member Function Documentation	36
4.9.2.1 clone()	36
4.9.2.2 getPayloadDescription()	37
4.9.2.3 insertCrew()	37
4.9.2.4 launchPayload()	37
4.9.2.5 printPayload()	38
4.10 Engine Class Reference	38
4.10.1 Constructor & Destructor Documentation	40

4.10.1.1 Engine() [1/2]	40
4.10.1.2 Engine() [2/2]	40
4.10.2 Member Function Documentation	40
4.10.2.1 clone()	40
4.10.2.2 setType()	41
4.11 Factory Class Reference	41
4.12 Falcon Class Reference	43
4.12.1 Member Function Documentation	45
4.12.1.1 clone()	45
4.12.1.2 getName()	45
4.13 FalconHeavy Class Reference	46
4.13.1 Member Function Documentation	48
4.13.1.1 clone()	48
4.13.1.2 getName()	48
4.14 FuelObserver Class Reference	49
4.14.1 Detailed Description	50
4.14.2 Constructor & Destructor Documentation	50
4.14.2.1 FuelObserver() [1/2]	50
4.14.2.2 FuelObserver() [2/2]	50
4.14.3 Member Function Documentation	50
4.14.3.1 setSubjectBooster()	51
4.14.3.2 update()	51
4.15 FuelS Class Reference	52
4.15.1 Member Function Documentation	53
4.15.1.1 changeState()	53
4.15.1.2 handle()	53
4.16 Iterator Class Reference	54
4.16.1 Member Function Documentation	55
4.16.1.1 End()	55
4.16.1.2 getCurr()	55
4.16.1.3 isEnd()	55
4.16.1.4 next()	56
4.16.1.5 start()	56
4.17 LaunchS Class Reference	56
4.17.1 Member Function Documentation	57
4.17.1.1 changeState()	57
4.17.1.2 handle()	58
4.18 MissionControlStarbase Class Reference	58
4.18.1 Member Function Documentation	59
4.18.1.1 construct()	59
4.18.1.2 launch()	59
4.19 Observer Class Reference	59

4.19.1 Detailed Description	61
4.19.2 Member Function Documentation	61
4.19.2.1 setName()	61
4.20 Payload Class Reference	61
4.20.1 Member Enumeration Documentation	62
4.20.1.1 PayloadType	62
4.20.2 Member Function Documentation	63
4.20.2.1 launchPayload()	63
4.21 Propulsion Class Reference	64
4.21.1 Member Function Documentation	65
4.21.1.1 add()	65
4.21.1.2 attach()	66
4.21.1.3 clone()	66
4.21.1.4 detach()	66
4.21.1.5 getLOXfuelLevel()	67
4.21.1.6 getRP1fuelLevel()	67
4.21.1.7 notify()	67
4.21.1.8 remove()	67
4.21.1.9 setLOXfuelLevel()	68
4.21.1.10 setRP1fuelLevel()	68
4.22 Rocket Class Reference	69
4.22.1 Member Enumeration Documentation	70
4.22.1.1 RocketType	70
4.22.2 Member Function Documentation	70
4.22.2.1 getFirstStage()	70
4.22.2.2 getSecondStage()	71
4.22.2.3 setRocketType()	71
4.23 RocketAggregate Class Reference	72
4.23.1 Member Function Documentation	73
4.23.1.1 add()	73
4.23.1.2 createIterator()	73
4.23.1.3 remove()	73
4.24 RocketIterator Class Reference	74
4.24.1 Constructor & Destructor Documentation	76
4.24.1.1 RocketIterator()	76
4.24.2 Member Function Documentation	76
4.24.2.1 End()	76
4.24.2.2 getCurr()	76
4.24.2.3 isEnd()	77
4.24.2.4 next()	77
4.24.2.5 start()	77
4.25 RocketMemento Class Reference	78

4.26 Satellite Class Reference	79
4.26.1 Constructor & Destructor Documentation	80
4.26.1.1 Satellite() [1/2]	80
4.26.1.2 Satellite() [2/2]	81
4.26.2 Member Function Documentation	81
4.26.2.1 clone()	81
4.26.2.2 handleRequest()	81
4.27 SatelliteFactory Class Reference	82
4.27.1 Constructor & Destructor Documentation	83
4.27.1.1 SatelliteFactory() [1/2]	83
4.27.1.2 SatelliteFactory() [2/2]	83
4.27.2 Member Function Documentation	83
4.27.2.1 createComponent()	83
4.28 SatelliteLauncher Class Reference	84
4.28.1 Constructor & Destructor Documentation	85
4.28.1.1 SatelliteLauncher()	85
4.28.1.2 ~SatelliteLauncher()	86
4.28.2 Member Function Documentation	86
4.28.2.1 add()	86
4.28.2.2 count()	86
4.29 SecondStage Class Reference	87
4.29.1 Member Function Documentation	89
4.29.1.1 clone()	89
4.29.1.2 getChildBooster()	89
4.29.1.3 getName()	90
4.30 Starlink Class Reference	90
4.30.1 Constructor & Destructor Documentation	92
4.30.1.1 Starlink()	92
4.30.1.2 ~Starlink()	92
4.30.2 Member Function Documentation	92
4.30.2.1 addSat()	92
4.30.2.2 clone()	92
4.30.2.3 getPayloadDescription()	93
4.30.2.4 LaunchAllSatellites()	93
4.30.2.5 launchPayload()	93
4.30.2.6 printPayload()	94
4.31 State Class Reference	94
4.31.1 Member Function Documentation	95
4.31.1.1 changeState()	95
4.31.1.2 handle()	95

Chapter 1

Todo List

Member `ConcreteBuilder::setPayload` (string payloadDescription) override

see if should add

Member `ConcreteBuilder::setPayload` (vector< string > astronauts, vector< string > ranks) override

see if should add

Member `ConcreteBuilder::setPayload` (int numSatellites) override

see if should add

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Aggregate	9
RocketAggregate	72
Builder	18
ConcreteBuilder	26
Caretaker	24
Factory	41
SatelliteFactory	82
Iterator	54
RocketIterator	74
MissionControlStarbase	58
Observer	59
FuelObserver	49
Payload	61
Dragon	31
DragonCrew	34
Starlink	90
Propulsion	64
Booster	11
Falcon	43
FalconHeavy	46
SecondStage	87
Engine	38
Rocket	69
RocketMemento	78
SatelliteLauncher	84
Satellite	79
State	94
ActiveLaunchS	7
BuildS	22
FuelS	52
LaunchS	56

Chapter 3

Class Index

3.1 Class List

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

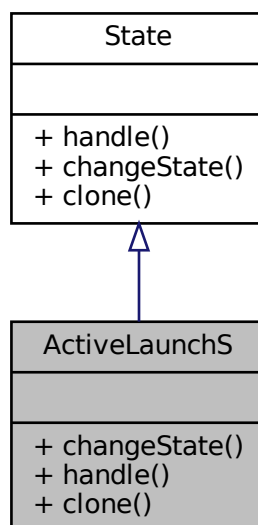
ActiveLaunchS	7
Aggregate	9
Booster	
Booster class, child of Propulsion	11
Builder	
Interface for a basic rocket builder class	18
BuildS	22
Caretaker	24
ConcreteBuilder	
Implements Builder, constructs Rockets according to a structure	26
Dragon	31
DragonCrew	34
Engine	38
Factory	41
Falcon	43
FalconHeavy	46
FuelObserver	
A class that observers fuel	49
FuelS	52
Iterator	54
LaunchS	56
MissionControlStarbase	58
Observer	
A Booster observer class	59
Payload	61
Propulsion	64
Rocket	69
RocketAggregate	72
RocketIterator	74
RocketMemento	78
Satellite	79
SatelliteFactory	82
SatelliteLauncher	84
SecondStage	87
Starlink	90
State	94

Chapter 4

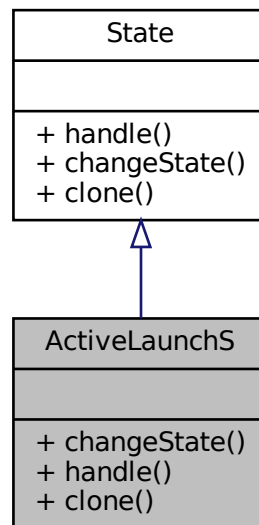
Class Documentation

4.1 ActiveLaunchS Class Reference

Inheritance diagram for ActiveLaunchS:



Collaboration diagram for ActiveLaunchS:



Public Member Functions

- void `changeState (Rocket *aR)`
Change rocket state from Active Launch state.
- void `handle (Rocket *aR)` override
Prints out a message explaining the.
- `State * clone ()` override
returns a clone of the current state

4.1.1 Member Function Documentation

4.1.1.1 changeState()

```
void ActiveLaunchS::changeState (
    Rocket * aR ) [virtual]
```

Change rocket state from Active Launch state.

You can not change state further while in Active Launch state.

Parameters

<i>aR</i>	
-----------	--

refuel

Implements [State](#).

4.1.1.2 handle()

```
void ActiveLaunchS::handle (
    Rocket * aR ) [override], [virtual]
```

Prints out a message explaining the.

Parameters

<i>aR</i>	The rocket being handled
-----------	--------------------------

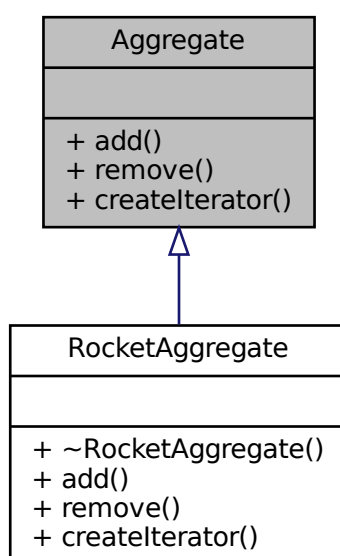
Implements [State](#).

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

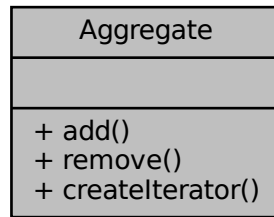
- State/ActiveLaunchS.h
- State/ActiveLaunchS.cpp

4.2 Aggregate Class Reference

Inheritance diagram for Aggregate:



Collaboration diagram for Aggregate:



Public Member Functions

- virtual void [add](#) ([RocketMemento](#) *aR)=0
Add a [RocketMemento](#) to the vector.
- virtual void [remove](#) ([RocketMemento](#) *aR)=0
Remove a [RocketMemento](#) from the vector.
- virtual [Iterator](#) * [createIterator](#) ()=0
Create an iterator.

4.2.1 Member Function Documentation

4.2.1.1 add()

```
virtual void Aggregate::add (  
    RocketMemento * aR ) [pure virtual]
```

Add a [RocketMemento](#) to the vector.

Parameters

in	<i>aR</i>	pointer to a RocketMemento object
----	-----------	---

Implemented in [RocketAggregate](#).

4.2.1.2 createIterator()

```
virtual Iterator* Aggregate::createIterator ( ) [pure virtual]
```

Create an iterator.

Returns

newly created iterator

Implemented in [RocketAggregate](#).

4.2.1.3 remove()

```
virtual void Aggregate::remove (
    RocketMemento * aR ) [pure virtual]
```

Remove a [RocketMemento](#) from the vector.

Parameters

in	<i>aR</i>	pointer to a RocketMemento object
----	-----------	---

Implemented in [RocketAggregate](#).

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

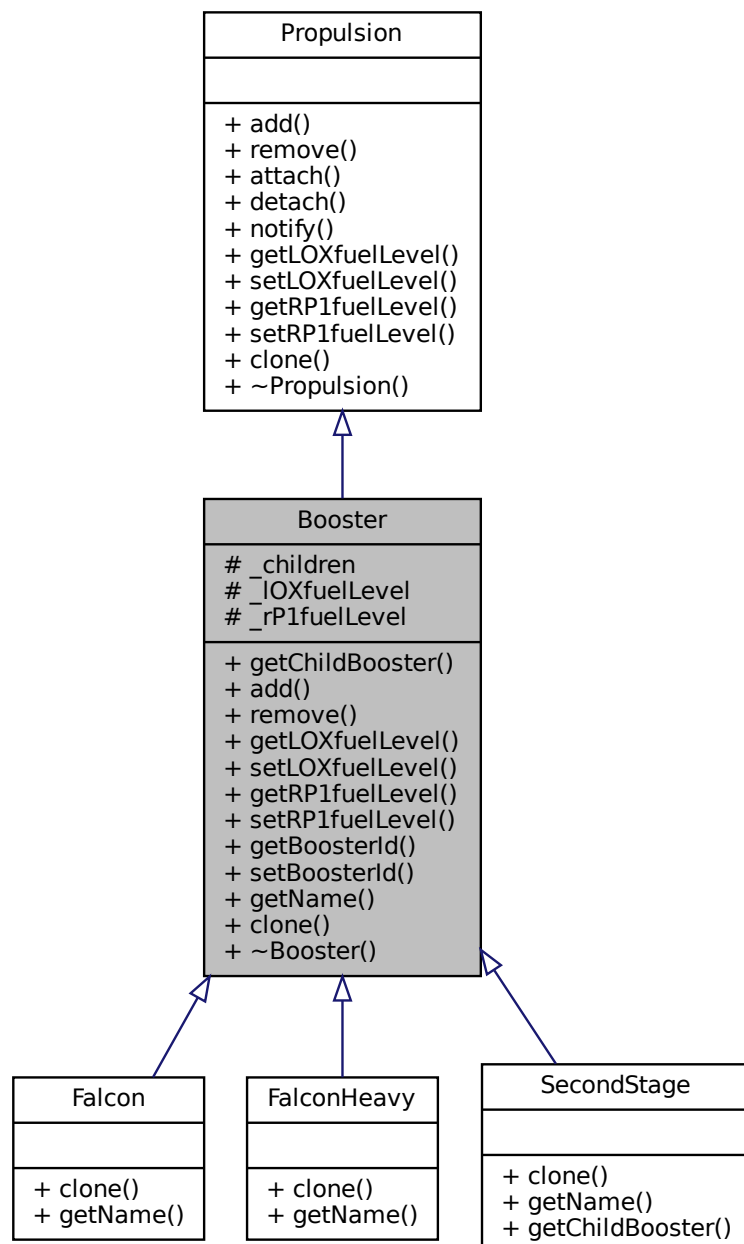
- Storage/Aggregate.h

4.3 Booster Class Reference

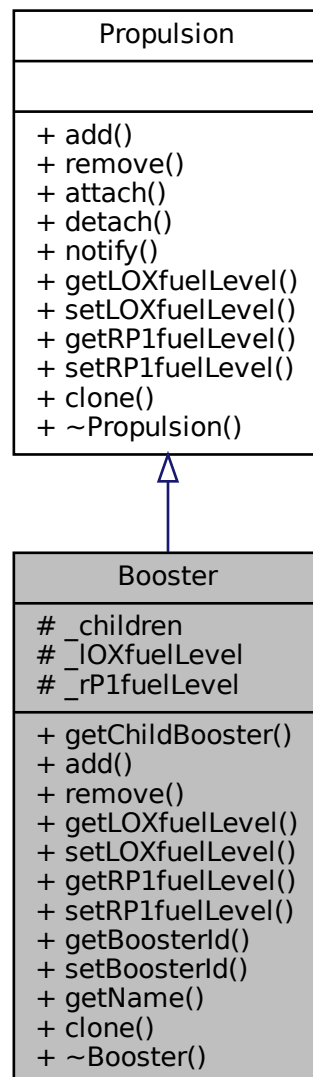
[Booster](#) class, child of [Propulsion](#).

```
#include <Booster.h>
```

Inheritance diagram for Booster:



Collaboration diagram for Booster:



Public Member Functions

- virtual [Booster](#) * [getChildBooster](#) (int index)
- void [add](#) ([Propulsion](#) *aP) override
adds a propulsion object to the composite structure
- void [remove](#) ([Propulsion](#) *aP) override
removes a propulsion object
- int [getLOXfuelLevel](#) ()
Getter for the LOXfuelLevel.
- void [setLOXfuelLevel](#) (int aLOXfuelLevel)
Sets the LOX fuel level.

- int `getRP1fuelLevel ()`
Getter for the RP1 fuel level.
- void `setRP1fuelLevel (int aRP1fuelLevel)`
Setter for the RP1 fuel level.
- int `getBoosterId ()`
Getter for the booster id.
- void `setBoosterId (int id)`
Sets the booster id.
- virtual string `getName ()`
Getter for the booster name.
- `Propulsion * clone ()`
Returns a clone of the entire tree structure.
- `~Booster ()` override
Destructor for `Booster` , deletes all branches.

Protected Attributes

- vector< `Propulsion *` > `_children`
Children in the composite structure.
- int `_IOXfuelLevel`
LOX Fuel level as a percentage.
- int `_rP1fuelLevel`
RP1 Fuel level as a percentage.

4.3.1 Detailed Description

`Booster` class, child of `Propulsion`.

Author

Jonathan Enslin - u19103345

4.3.2 Constructor & Destructor Documentation

4.3.2.1 `~Booster()`

```
Booster::~Booster ( ) [override]
```

Destructor for `Booster` , deletes all branches.

4.3.3 Member Function Documentation

4.3.3.1 `add()`

```
void Booster::add (
    Propulsion * aP ) [override], [virtual]
```

adds a propulsion object to the composite structure

Parameters

<code>in</code>	<code>aP</code>	the propulsion object to add to the composite
-----------------	-----------------	---

`aP` will be to the vector of propulsion objects

Implements [Propulsion](#).

4.3.3.2 clone()

```
Propulsion * Booster::clone ( ) [virtual]
```

Returns a clone of the entire tree structure.

Returns

Propulsion*

Implements [Propulsion](#).

Reimplemented in [Falcon](#), [FalconHeavy](#), and [SecondStage](#).

4.3.3.3 getBoosterId()

```
int Booster::getBoosterId ( )
```

Getter for the booster id.

Returns

booster id

4.3.3.4 getChildBooster()

```
Booster * Booster::getChildBooster (
    int index ) [virtual]
```

See child classes for more implementation This method returns null

Parameters

<code>index</code>	The index of the child booster to obtain
--------------------	--

Returns

Booster*

Reimplemented in [SecondStage](#).

4.3.3.5 getLOXfuelLevel()

```
int Booster::getLOXfuelLevel ( ) [virtual]
```

Getter for the LOXfuelLevel.

Returns

the LOX fuel level value

Implements [Propulsion](#).

4.3.3.6 getName()

```
string Booster::getName ( ) [virtual]
```

Getter for the booster name.

Returns

booster name as string

Reimplemented in [Falcon](#), [FalconHeavy](#), and [SecondStage](#).

4.3.3.7 getRP1fuelLevel()

```
int Booster::getRP1fuelLevel ( ) [virtual]
```

Getter for the RP1 fuel level.

Returns

RP1 fuel level

Implements [Propulsion](#).

4.3.3.8 remove()

```
void Booster::remove (
    Propulsion * aP ) [override], [virtual]
```

removes a propulsion object

Parameters

<i>aP</i>	Propulsion object to be removed
-----------	---

This will *aP* from the `_children` vector

Implements [Propulsion](#).

4.3.3.9 setBoosterId()

```
void Booster::setBoosterId (
    int id )
```

Sets the booster id.

Parameters

in	<i>id</i>	The value the booster id should be set to
----	-----------	---

4.3.3.10 setLOXfuelLevel()

```
void Booster::setLOXfuelLevel (
    int aLOXfuelLevel ) [virtual]
```

Sets the LOX fuel level.

Parameters

in	<i>aLOXfuelLevel</i>	The value LOX fuel level should be set to
----	----------------------	---

Implements [Propulsion](#).

4.3.3.11 setRP1fuelLevel()

```
void Booster::setRP1fuelLevel (
    int aRP1fuelLevel ) [virtual]
```

Setter for the RP1 fuel level.

Implements [Propulsion](#).

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

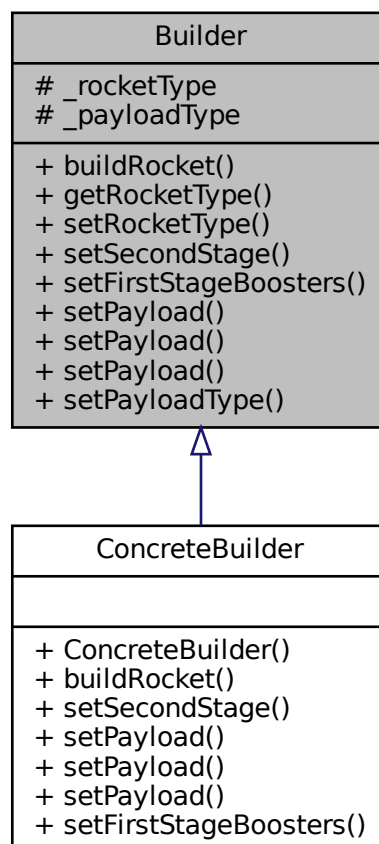
- `Propulsion/Booster.h`
- `Propulsion/Booster.cpp`

4.4 Builder Class Reference

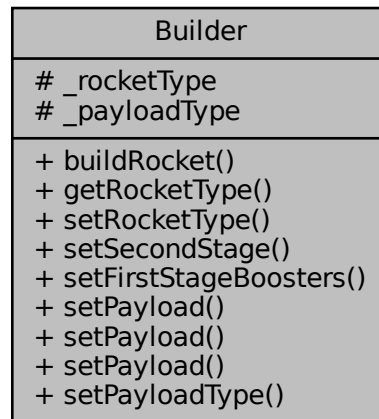
Interface for a basic rocket builder class.

```
#include <Builder.h>
```

Inheritance diagram for Builder:



Collaboration diagram for Builder:



Public Member Functions

- virtual [Rocket](#) * [buildRocket](#) ()=0
This methods returns the rocket that was built.
- [Rocket::RocketType](#) [getRocketType](#) ()
Getter for the rocket type.
- void [setRocketType](#) ([Rocket::RocketType](#) aRocketType)
Setter for the rocket type.
- virtual void [setSecondStage](#) ()=0
Sets the second stage of the rocket.
- virtual void [setFirstStageBoosters](#) ()=0
Adder for the first stage of the rocket.
- virtual void [setPayload](#) (string payloadDescription)=0
see child class for details
- virtual void [setPayload](#) (int numSatellites)=0
see child class for details
- virtual void [setPayload](#) (vector< string > astronauts, vector< string > ranks)=0
see child class for details
- virtual void [setPayloadType](#) ([Payload::PayloadType](#) aPayloadType)
Setter for the payload type of the rocket being built.

Protected Attributes

- [Rocket::RocketType](#) _rocketType
The type of the rocket.
- [Payload::PayloadType](#) _payloadType
The type of the payload.

4.4.1 Detailed Description

Interface for a basic rocket builder class.

Note

- Abstract class

4.4.2 Member Function Documentation

4.4.2.1 buildRocket()

```
virtual Rocket* Builder::buildRocket ( ) [pure virtual]
```

This methods returns the rocket that was built.

Returns

A pointer to the rocket that was built

see the children class for implementation details

Implemented in [ConcreteBuilder](#).

4.4.2.2 getRocketType()

```
Rocket::RocketType Builder::getRocketType ( )
```

Getter for the rocket type.

Returns

`_rocketType` - A that is equal to the rockets type

4.4.2.3 setFirstStageBoosters()

```
virtual void Builder::setFirstStageBoosters ( ) [pure virtual]
```

Adder for the first stage of the rocket.

See the the children classes for implementation details

Implemented in [ConcreteBuilder](#).

4.4.2.4 setPayloadType()

```
void Builder::setPayloadType (
    Payload::PayloadType aPayloadType ) [virtual]
```

Setter for the payload type of the rocket being built.

Parameters

in	<i>aPayload</i>	- The payload type to be produced
----	-----------------	-----------------------------------

4.4.2.5 setRocketType()

```
void Builder::setRocketType (
    Rocket::RocketType aRocketType )
```

Setter for the rocket type.

Parameters

<i>aRocketType</i>	The type of the rocket
--------------------	------------------------

4.4.2.6 setSecondStage()

```
virtual void Builder::setSecondStage ( ) [pure virtual]
```

Sets the second stage of the rocket.

See children classess for implementation details

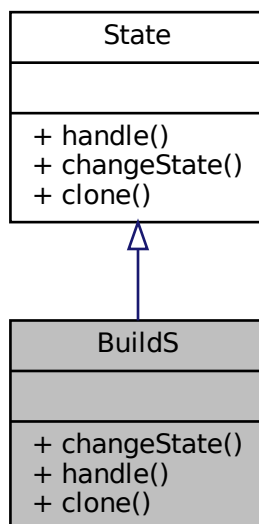
Implemented in [ConcreteBuilder](#).

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

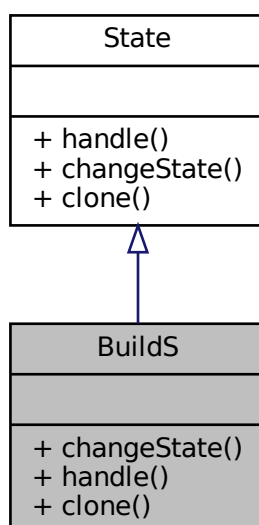
- Builder/Builder.h
- Builder/Builder.cpp

4.5 BuildS Class Reference

Inheritance diagram for BuildS:



Collaboration diagram for BuildS:



Public Member Functions

- void `changeState` (`Rocket *aR`) override
Changes to the Fuel `State`.
- void `handle` (`Rocket *aR`) override
Prints out a message explaining the.
- `State * clone` () override
returns a clone of the current state

4.5.1 Member Function Documentation

4.5.1.1 `changeState()`

```
void BuildS::changeState (  
    Rocket * aR ) [override], [virtual]
```

Changes to the Fuel `State`.

Changes the state of the rocket from Build state.

Parameters

<code>aR</code>	The rocket changing state
-----------------	---------------------------

Changes the rocket state from being built to Fueling state. This just simply calls the rocket set state and passes the new Fuel state.

Parameters

<code>aR</code>	
-----------------	--

Implements `State`.

4.5.1.2 `handle()`

```
void BuildS::handle (  
    Rocket * aR ) [override], [virtual]
```

Prints out a message explaining the.

Parameters

<code>aR</code>	The rocket being handled
-----------------	--------------------------

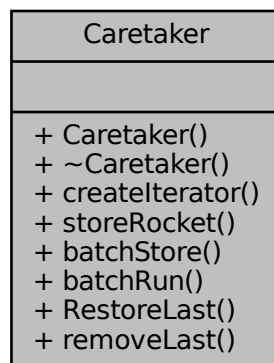
Implements [State](#).

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

- State/BuildS.h
- State/BuildS.cpp

4.6 Caretaker Class Reference

Collaboration diagram for Caretaker:



Public Member Functions

- [Iterator](#) * [createIterator](#) ()
Creates an iterator that is used to traverse the memento list.
- void [storeRocket](#) ([RocketMemento](#) *rockMem)
Adds a rocket to the memento list.
- void [batchStore](#) ([RocketAggregate](#) *aBatch)
Adds in a batch of rockets.
- void [batchRun](#) ()
- [RocketMemento](#) * [RestoreLast](#) ()
Returns lastly stored rocket's state.
- void [removeLast](#) ()

4.6.1 Member Function Documentation

4.6.1.1 batchStore()

```
void Caretaker::batchStore (
    RocketAggregate * aBatch )
```

Adds in a batch of rockets.

Parameters

<i>aBatch</i>	a Batch t of rockets that will server as the new memento list
---------------	---

4.6.1.2 createIterator()

```
Iterator * Caretaker::createIterator ( )
```

Creates an iterator that is used to traverse the memento list.

Returns

return the iterator

4.6.1.3 RestoreLast()

```
RocketMemento * Caretaker::RestoreLast ( )
```

Returns lastly stored rocket's state.

Returns

returns lastly added [RocketMemento](#)

4.6.1.4 storeRocket()

```
void Caretaker::storeRocket (
    RocketMemento * rockMem )
```

Adds a rocket to the memento list.

Parameters

<i>rockMem</i>	a pointer to a RocketMemento to be added to memento list
----------------	--

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

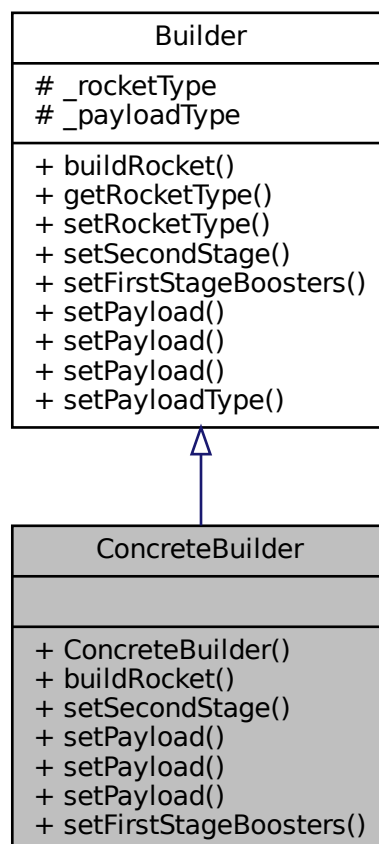
- Storage/Caretaker.h
- Storage/Caretaker.cpp

4.7 ConcreteBuilder Class Reference

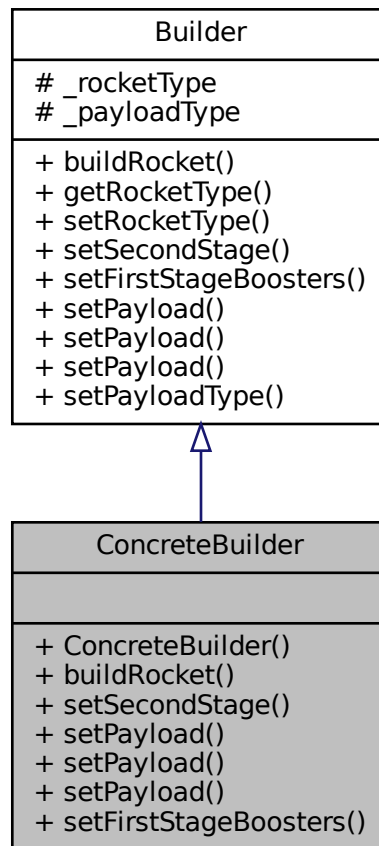
Implements [Builder](#), constructs Rockets according to a structure.

```
#include <ConcreteBuilder.h>
```

Inheritance diagram for ConcreteBuilder:



Collaboration diagram for ConcreteBuilder:



Public Member Functions

- [ConcreteBuilder](#) ([Rocket::RocketType](#) aRocketType, [Payload::PayloadType](#) aPayloadType=[Payload::CARGO](#))
A parameterized constructor.
- [Rocket](#) * [buildRocket](#) () override
Returns the built rocket.
- void [setSecondStage](#) () override
Sets the second stage of the rocket.
- virtual void [setPayload](#) (string payloadDescription) override
*Sets the payload when using **CARGO** payload type.*
- virtual void [setPayload](#) (int numSatellites) override
*Set the [Payload](#) object when using **STARLINK** payload type.*
- virtual void [setPayload](#) (vector< string > astronauts, vector< string > ranks) override
*Set the [Payload](#) object when using **CREW** payload type.*
- void [setFirstStageBoosters](#) () override
Adds a first stage to the [Rocket](#).

Additional Inherited Members

4.7.1 Detailed Description

Implements [Builder](#), constructs Rockets according to a structure.

This class implements the interface of the [Builder](#) class, it constructs [FalconHeavy](#) and [Falcon](#) 9 rockets, the structure followed is as follows: SecondStage->(engine,[firstStages[Engines]])

Note

- Uses the `PayloadType` enum from

4.7.2 Constructor & Destructor Documentation

4.7.2.1 ConcreteBuilder()

```
ConcreteBuilder::ConcreteBuilder (
    Rocket::RocketType aRocketType,
    Payload::PayloadType aPayloadType = Payload::CARGO )
```

A parameterized constructor.

Parameters

<i>aRocketType</i>	The type of the rocket
<i>aPayloadType</i>	The type of the rockets payload

A default constructor that sets `secondStageSet` to `false` and `numFirstStage` to 0 It also creates a new rocket with no parts set

4.7.3 Member Function Documentation

4.7.3.1 buildRocket()

```
Rocket * ConcreteBuilder::buildRocket ( ) [override], [virtual]
```

Returns the built rocket.

Returns

- A pointer to the rocket that has been built

This method returns a pointer to the rocket that has been constructed.

Implements [Builder](#).

4.7.3.2 setFirstStageBoosters()

```
void ConcreteBuilder::setFirstStageBoosters ( ) [override], [virtual]
```

Adds a first stage to the [Rocket](#).

This method will add a first stage [Booster](#) to the rocket. it will also automatically add engines to the rocket

Implements [Builder](#).

4.7.3.3 setPayload() [1/3]

```
void ConcreteBuilder::setPayload (
    int numSatellites ) [override], [virtual]
```

Set the [Payload](#) object when using **STARLINK** payload type.

Parameters

in	<i>numSatellites</i>	The number of satellits to add to the chain of satellites (chain of responsibility)
----	----------------------	---

This function will create a starlink payload, for **STARLINK** payload type containing the number of satellites specified as a parameter

Note

- Only to be used when payload type is **STARLNK** . If it is not this type the function will throw a `string`

Todo see if should add

Implements [Builder](#).

4.7.3.4 setPayload() [2/3]

```
void ConcreteBuilder::setPayload (
    string payloadDescription ) [override], [virtual]
```

Sets the payload when using **CARGO** payload type.

Parameters

in	<i>payloadDescription</i>	- The payload description to give to the payload
----	---------------------------	--

Sets the payload for payload type CARGO, will automatically create payload an initialise with parameters

Note

- This functions is only to be used for setting the payload when `payLoadType` is **CARGO** . If it is not this type, the function will throw a `string`

Todo see if should add

Implements [Builder](#).

4.7.3.5 setPayload() [3/3]

```
void ConcreteBuilder::setPayload (
    vector< string > astronauts,
    vector< string > ranks ) [override], [virtual]
```

Set the [Payload](#) object when using **CREW** payload type.

Parameters

<i>astronauts</i>	A vector of astronauts to add to the payload
<i>ranks</i>	A vector of the respective ranks of the astronauts

This will create a payload containing the passed astronauts, the vector of ranks should ideally be the same length as the astronaut array, but differences in length will be ignored

Note

- Only to be used when payload type is **CREW** . If it is not this type the function will throw a `string`

Todo see if should add

Implements [Builder](#).

4.7.3.6 setSecondStage()

```
void ConcreteBuilder::setSecondStage ( ) [override], [virtual]
```

Sets the second stage of the rocket.

This function will add a second stage to the rocket, it will also automatically add an engine to the rocket

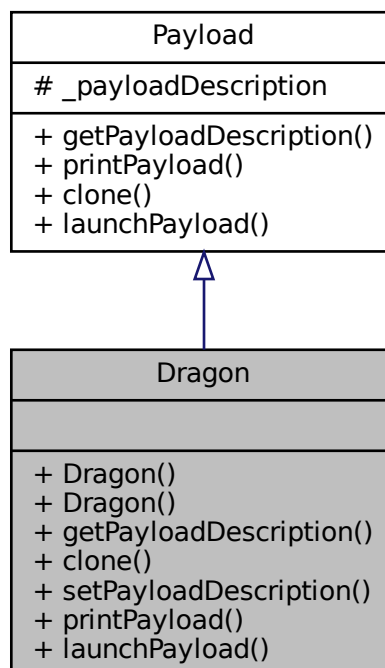
Implements [Builder](#).

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

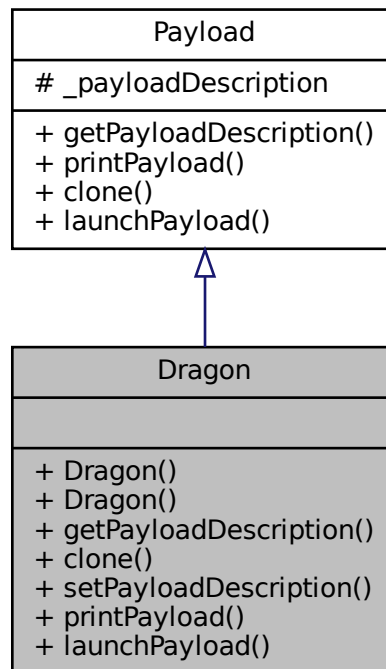
- Builder/ConcreteBuilder.h
- Builder/ConcreteBuilder.cpp

4.8 Dragon Class Reference

Inheritance diagram for Dragon:



Collaboration diagram for Dragon:



Public Member Functions

- `Dragon` (const `Dragon` &obj)
Construct a new `Dragon::Dragon` object.
- `Dragon` ()
Construct a new `Dragon::Dragon` object.
- string `getPayloadDescription` ()
Gets the description of the `Dragon` payload as a string.
- `Payload * clone` ()
clone function for `Dragon`
- void `setPayloadDescription` (string aDescription)
- void `printPayload` ()
Displays the description of the `Dragon` payload to the user.
- void `launchPayload` ()
Launches the payload.

Additional Inherited Members

4.8.1 Constructor & Destructor Documentation

4.8.1.1 Dragon() [1/2]

```
Dragon::Dragon (
    const Dragon & obj )
```

Construct a new [Dragon::Dragon](#) object.

Copy construct from an existing [Dragon](#) payload.

Parameters

<i>obj</i>	
------------	--

4.8.1.2 Dragon() [2/2]

```
Dragon::Dragon ( )
```

Construct a new [Dragon::Dragon](#) object.

Create the [Dragon](#) payload.

4.8.2 Member Function Documentation

4.8.2.1 clone()

```
Payload * Dragon::clone ( ) [virtual]
```

clone function for [Dragon](#)

Returns

a pointer to payload

Implements [Payload](#).

4.8.2.2 getPayloadDescription()

```
string Dragon::getPayloadDescription ( ) [virtual]
```

Gets the description of the [Dragon](#) payload as a string.

Every [Dragon](#) payload has a preset payload description. This function simply returns the description as a string.

Returns

string

Implements [Payload](#).

4.8.2.3 launchPayload()

```
void Dragon::launchPayload ( ) [virtual]
```

Launches the payload.

Implements [Payload](#).

4.8.2.4 printPayload()

```
void Dragon::printPayload ( ) [virtual]
```

Displays the description of the [Dragon](#) payload to the user.

Every [Dragon](#) payload has a preset payload description. This function simply takes the string return of [getPayloadDescription\(\)](#) and gives it to the user.

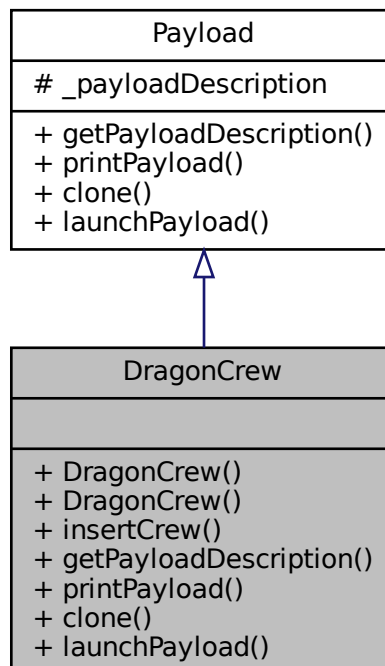
Implements [Payload](#).

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

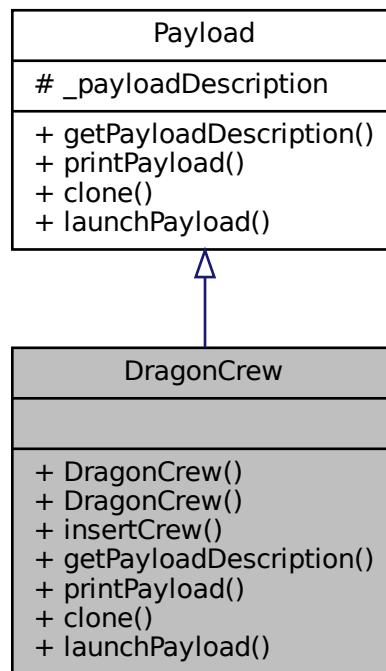
- Payload/Dragon.h
- Payload/Dragon.cpp

4.9 DragonCrew Class Reference

Inheritance diagram for DragonCrew:



Collaboration diagram for DragonCrew:



Public Member Functions

- `DragonCrew` (const `DragonCrew` &obj)
Copy constructor for a new `DragonCrew::DragonCrew` object.
- `DragonCrew` ()
Construct a new `DragonCrew::DragonCrew` object.
- void `insertCrew` (string Name, string Rank)
Insert a crew member into the `DragonCrew` object.
- string `getPayloadDescription` ()
Gets the description of the `DragonCrew` payload as a string.
- void `printPayload` ()
Displays the `DragonCrew` payload to the user.
- `Payload` * `clone` ()
A clone for the dragon crew.
- void `launchPayload` ()
Launches payload.

Additional Inherited Members

4.9.1 Constructor & Destructor Documentation

4.9.1.1 DragonCrew() [1/2]

```
DragonCrew::DragonCrew (
    const DragonCrew & obj )
```

Copy constructor for a new [DragonCrew::DragonCrew](#) object.

Every [DragonCrew](#) payload has a preset payload description. The crew member array is then copied over to the new [DragonCrew](#). Then the description is copied aswell.

Parameters

<i>obj</i>	
------------	--

4.9.1.2 DragonCrew() [2/2]

```
DragonCrew::DragonCrew ( )
```

Construct a new [DragonCrew::DragonCrew](#) object.

Every [DragonCrew](#) payload has a preset payload description. This function simply returns the description as a string. [DragonCrew](#) has a permanent 7 size for 7 crew members.

4.9.2 Member Function Documentation

4.9.2.1 clone()

```
Payload * DragonCrew::clone ( ) [virtual]
```

A clone for the dragon crew.

Returns

A [Payload](#) Pointer

Implements [Payload](#).

4.9.2.2 getPayloadDescription()

```
string DragonCrew::getPayloadDescription ( ) [virtual]
```

Gets the description of the [DragonCrew](#) payload as a string.

Builds a description string from the given info. Builds the list of crew members. Then adds the description to the out string.

Returns

string

Implements [Payload](#).

4.9.2.3 insertCrew()

```
void DragonCrew::insertCrew (
    string Name,
    string Rank )
```

Insert a crew member into the [DragonCrew](#) object.

When inserting into the array. "" represents empty places. We use this to add crew members in line with their Name and Rank. Displays a message if the insertion was successful or a other message if payload is full.

Parameters

<i>Name</i>	
<i>Rank</i>	

4.9.2.4 launchPayload()

```
void DragonCrew::launchPayload ( ) [virtual]
```

Launches payload.

See child classes for implementation

Implements [Payload](#).

4.9.2.5 printPayload()

```
void DragonCrew::printPayload ( ) [virtual]
```

Displays the [DragonCrew](#) payload to the user.

Calls [getPayloadDescription\(\)](#) to receive the out string. The string is built with tabs and newlines for output. The string is displayed to the user.

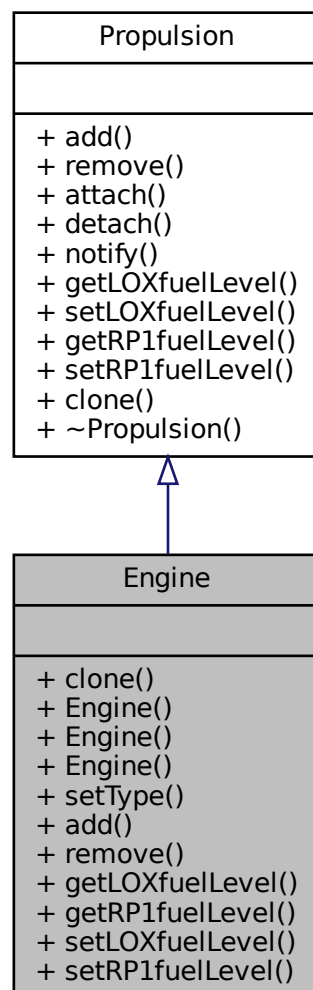
Implements [Payload](#).

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

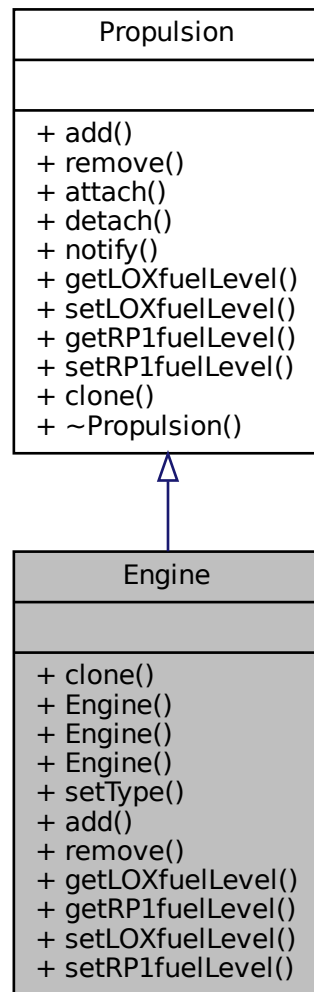
- Payload/DragonCrew.h
- Payload/DragonCrew.cpp

4.10 Engine Class Reference

Inheritance diagram for Engine:



Collaboration diagram for Engine:



Public Member Functions

- [Propulsion * clone \(\)](#)
Creates a clone of the engine.
- [Engine \(\)](#)
- [Engine \(string aType\)](#)
Parameterized constructor.
- [Engine \(const Engine &eng\)](#)
Copy constructor.
- void [setType](#) (string aType)
Sets the type of the engine.
- virtual void [add](#) ([Propulsion *aP](#))
Does nothing.
- virtual void [remove](#) ([Propulsion *aP](#))

- Does nothing.*
- int [getLOXfuelLevel](#) ()
- Does nothing.*
- int [getRP1fuelLevel](#) ()
- Does nothing.*
- void [setLOXfuelLevel](#) (int)
- Does nothing.*
- void [setRP1fuelLevel](#) (int)
- Does nothing.*

4.10.1 Constructor & Destructor Documentation

4.10.1.1 Engine() [1/2]

```
Engine::Engine ( )
```

Default constructor for [Engine](#), sets type to an empty string

4.10.1.2 Engine() [2/2]

```
Engine::Engine (
    string aType )
```

Parameterized constructor.

Parameters

in	<i>aType</i>	the type of the engine
----	--------------	------------------------

4.10.2 Member Function Documentation

4.10.2.1 clone()

```
Propulsion * Engine::clone ( ) [virtual]
```

Creates a clone of the engine.

Returns

Pointer to the clone of this engine

Implements [Propulsion](#).

4.10.2.2 setType()

```
void Engine::setType (
    string aType )
```

Sets the type of the engine.

Parameters

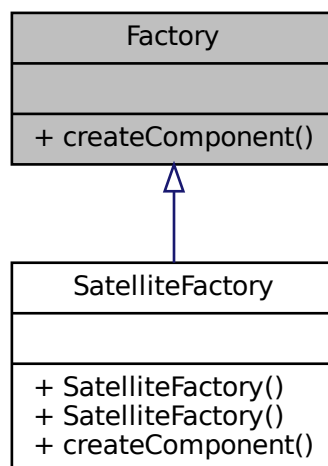
in	<i>aType</i>	The type of the rocket
----	--------------	------------------------

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

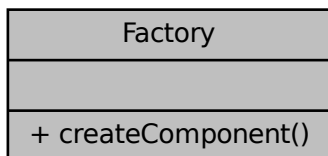
- Propulsion/Engine.h
- Propulsion/Engine.cpp

4.11 Factory Class Reference

Inheritance diagram for Factory:



Collaboration diagram for Factory:



Public Member Functions

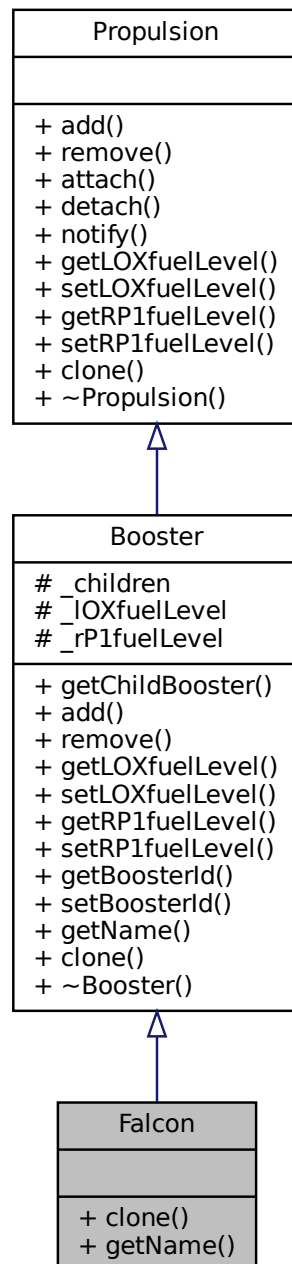
- virtual [SatelliteLauncher](#) * **createComponent** ()=0

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

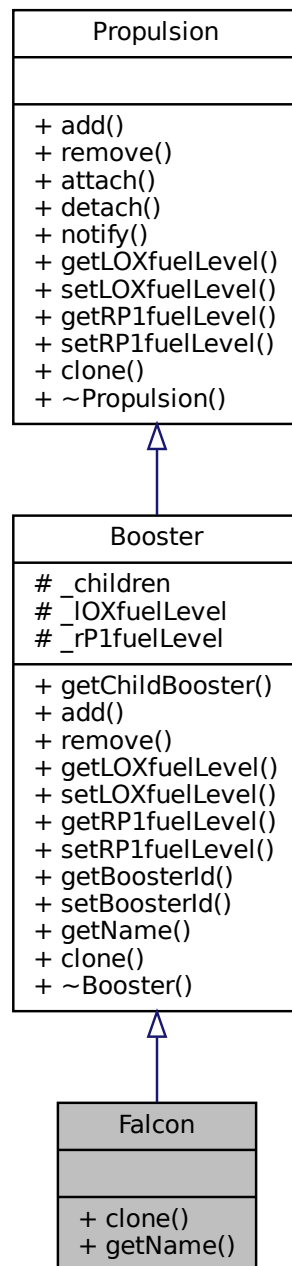
- `Payload/Factory.h`

4.12 Falcon Class Reference

Inheritance diagram for Falcon:



Collaboration diagram for Falcon:



Public Member Functions

- **Propulsion** * `clone()`
Returns a clone of the entire tree structure.
- string `getName()` override
Getter for the booster name.

Additional Inherited Members

4.12.1 Member Function Documentation

4.12.1.1 clone()

```
Propulsion * Falcon::clone ( ) [virtual]
```

Returns a clone of the entire tree structure.

Returns

Propulsion*

Reimplemented from [Booster](#).

4.12.1.2 getName()

```
string Falcon::getName ( ) [override], [virtual]
```

Getter for the booster name.

Returns

booster name as string

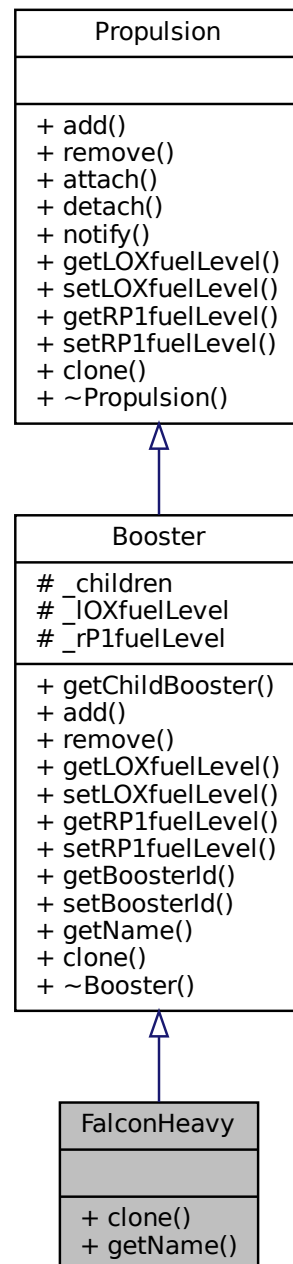
Reimplemented from [Booster](#).

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

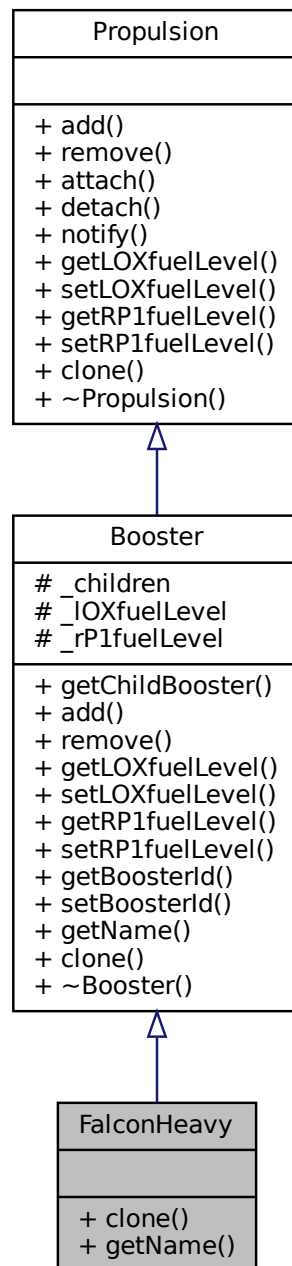
- Propulsion/Falcon.h
- Propulsion/Falcon.cpp

4.13 FalconHeavy Class Reference

Inheritance diagram for FalconHeavy:



Collaboration diagram for FalconHeavy:



Public Member Functions

- [Propulsion](#) * [clone](#) ()
Returns a clone of the entire tree structure.
- string [getName](#) () override
Getter for the booster name.

Additional Inherited Members

4.13.1 Member Function Documentation

4.13.1.1 clone()

```
Propulsion * FalconHeavy::clone ( ) [virtual]
```

Returns a clone of the entire tree structure.

Returns

Propulsion*

Reimplemented from [Booster](#).

4.13.1.2 getName()

```
string FalconHeavy::getName ( ) [override], [virtual]
```

Getter for the booster name.

Returns

booster name as string

Reimplemented from [Booster](#).

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

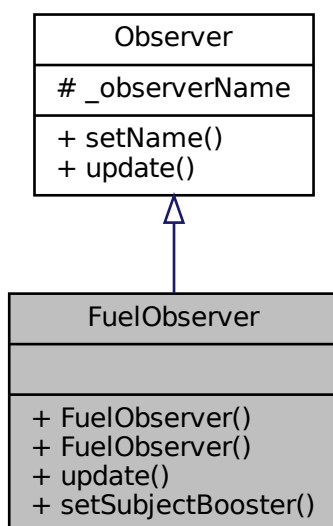
- Propulsion/FalconHeavy.h
- Propulsion/FalconHeavy.cpp

4.14 FuelObserver Class Reference

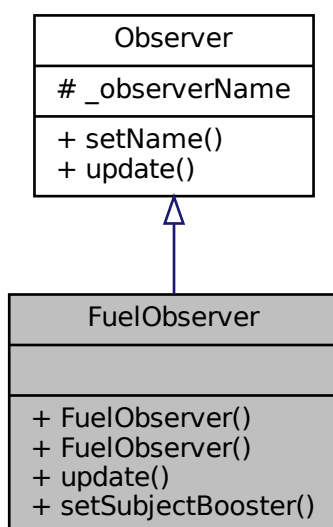
A class that observes fuel.

```
#include <FuelObserver.h>
```

Inheritance diagram for FuelObserver:



Collaboration diagram for FuelObserver:



Public Member Functions

- [FuelObserver](#) ()
- [FuelObserver](#) (string aName)
Constructor that initialises name.
- void [update](#) ()
Called by a subject object This will update the values of `_lOXfuelState` and `_rP1fuelState` and then call the.
- void [setSubjectBooster](#) ([Booster](#) *aBooster)

Additional Inherited Members

4.14.1 Detailed Description

A class that observers fuel.

Author

Jonathan Enslin - u19103345

4.14.2 Constructor & Destructor Documentation

4.14.2.1 [FuelObserver\(\)](#) [1/2]

```
FuelObserver::FuelObserver ( )
```

Default constructor for [FuelObserver](#), sets name to empty string

4.14.2.2 [FuelObserver\(\)](#) [2/2]

```
FuelObserver::FuelObserver (
    string aName )
```

Constructor that initialises name.

Parameters

<i>aName</i>	The name of the Observer
--------------	--

4.14.3 Member Function Documentation

4.14.3.1 setSubjectBooster()

```
void FuelObserver::setSubjectBooster (
    Booster * aBooster )
```

Sets the booster that will be observed

Parameters

in	<i>aBooster</i>	Pointer to the booster being observed
----	-----------------	---------------------------------------

Note

- method could be moved to parent class, in which case `_concreteBooster` could also be moved to parent Class

4.14.3.2 update()

```
void FuelObserver::update ( ) [virtual]
```

Called by a subject object This will update the values of `_lOXfuelState` and `_rP1fuelState` and then call the.

See also

`FuelObserver::assessFuel()` for the remaining functionality

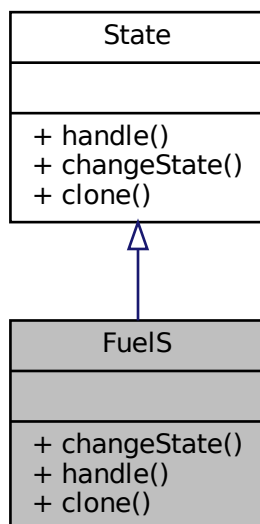
Implements [Observer](#).

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

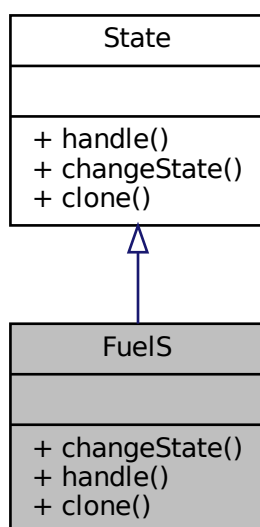
- Propulsion/FuelObserver.h
- Propulsion/FuelObserver.cpp

4.15 FuelS Class Reference

Inheritance diagram for FuelS:



Collaboration diagram for FuelS:



Public Member Functions

- void [changeState](#) ([Rocket](#) *aR)
Change the rocket state from Fueling state.
- void [handle](#) ([Rocket](#) *aR) override
Refuels the rocket.
- [State](#) * [clone](#) () override
returns a clone of the current state

4.15.1 Member Function Documentation

4.15.1.1 [changeState\(\)](#)

```
void FuelS::changeState (  
    Rocket * aR ) [virtual]
```

Change the rocket state from Fueling state.

Changes the rocket state from being fueled to launching state. This just simply calls the rocket set state and passes the new Launch state.

Parameters

<i>aR</i>	
-----------	--

Implements [State](#).

4.15.1.2 [handle\(\)](#)

```
void FuelS::handle (  
    Rocket * aR ) [override], [virtual]
```

Refuels the rocket.

Parameters

<i>aR</i>	The rocket being handled
-----------	--------------------------

This state refuels the rocket to 100%, it will also notify the observers of the rocket

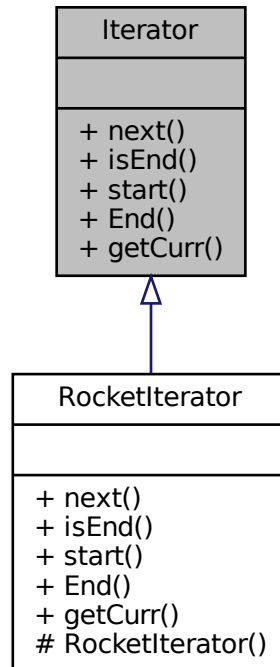
Implements [State](#).

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

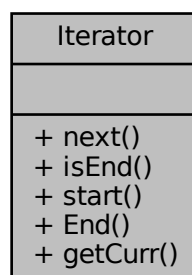
- State/FuelS.h
- State/FuelS.cpp

4.16 Iterator Class Reference

Inheritance diagram for Iterator:



Collaboration diagram for Iterator:



Public Member Functions

- virtual [RocketMemento](#) * `next` ()=0

- Move on to next element.*
 - virtual bool `isEnd()`=0
- Check if currently at last element.*
 - virtual `RocketMemento` * `start()`=0
- Move to the first element.*
 - virtual `RocketMemento` * `End()`=0
- Move to the last element.*
 - virtual `RocketMemento` * `getCurr()`=0
- Returns the current element.*

4.16.1 Member Function Documentation

4.16.1.1 End()

```
virtual RocketMemento* Iterator::End ( ) [pure virtual]
```

Move to the last element.

Returns

The last element

Implemented in `RocketIterator`.

4.16.1.2 getCurr()

```
virtual RocketMemento* Iterator::getCurr ( ) [pure virtual]
```

Returns the current element.

Returns

element that is currently being pointed to

Implemented in `RocketIterator`.

4.16.1.3 isEnd()

```
virtual bool Iterator::isEnd ( ) [pure virtual]
```

Check if currently at last element.

Returns

return true if at last element, else return false

Implemented in `RocketIterator`.

4.16.1.4 next()

```
virtual RocketMemento* Iterator::next ( ) [pure virtual]
```

Move on to next element.

Returns

element that is being pointed to after move

Implemented in [RocketIterator](#).

4.16.1.5 start()

```
virtual RocketMemento* Iterator::start ( ) [pure virtual]
```

Move to the first element.

Returns

The first element

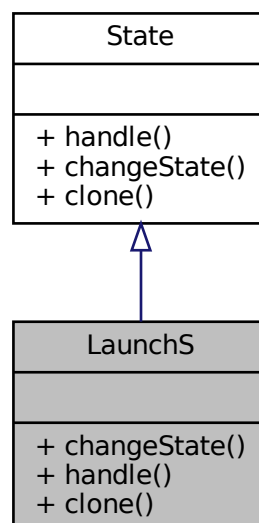
Implemented in [RocketIterator](#).

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

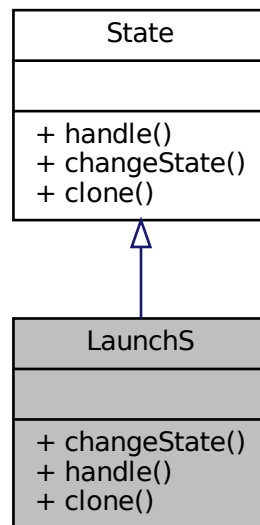
- Storage/Iterator.h

4.17 LaunchS Class Reference

Inheritance diagram for LaunchS:



Collaboration diagram for LaunchS:



Public Member Functions

- void `changeState` (`Rocket *aR`)
Change rocket state from Launch prep state.
- void `handle` (`Rocket *aR`) override
Launches the rocket.
- `State * clone` () override
returns a clone of the current state

4.17.1 Member Function Documentation

4.17.1.1 `changeState()`

```
void LaunchS::changeState (
    Rocket * aR ) [virtual]
```

Change rocket state from Launch prep state.

Changes the rocket state from being ready for launch to Active Launch state. This just simply calls the rocket set state and passes the new Active Launch state.

Parameters

<i>aR</i>	
-----------	--

Implements [State](#).

4.17.1.2 handle()

```
void LaunchS::handle (
    Rocket * aR ) [override], [virtual]
```

Launches the rocket.

Parameters

<i>aR</i>	The rocket being handled
-----------	--------------------------

Launches the rocket by modifying the fuel values, also notifies all observers

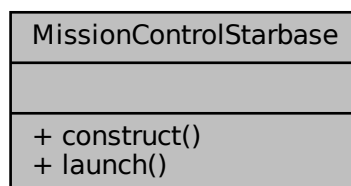
Implements [State](#).

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

- State/LaunchS.h
- State/LaunchS.cpp

4.18 MissionControlStarbase Class Reference

Collaboration diagram for MissionControlStarbase:



Public Member Functions

- [Rocket](#) * [construct](#) ([Builder](#) *aBuilder)
Construct a rocket.
- void [launch](#) ([Rocket](#) *R)
Launch [Rocket](#).

4.18.1 Member Function Documentation

4.18.1.1 `construct()`

```
Rocket * MissionControlStarbase::construct (
    Builder * aBuilder )
```

Construct a rocket.

Use a passed builder to build and put a rocket together for use. Also allow user to change builder parameters.

Parameters

<i>aBuilder</i>	
-----------------	--

Returns

Rocket*

4.18.1.2 `launch()`

```
void MissionControlStarbase::launch (
    Rocket * R )
```

Launch [Rocket](#).

Use rocket's states to go through it preparations and then launch.

Parameters

<i>R</i>	
----------	--

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

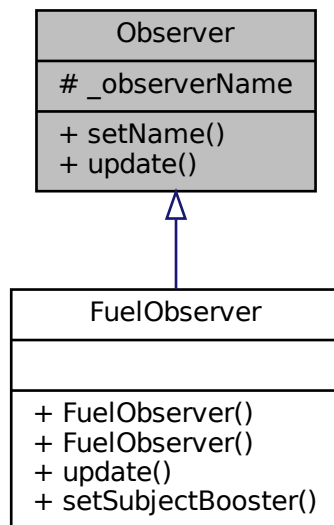
- `MissionControlStarbase.h`
- `MissionControlStarbase.cpp`

4.19 Observer Class Reference

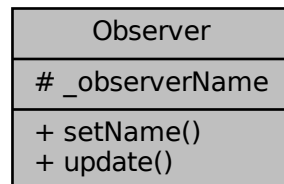
A [Booster](#) observer class.

```
#include <Observer.h>
```

Inheritance diagram for Observer:



Collaboration diagram for Observer:



Public Member Functions

- void `setName` (string name)
Sets the name of the observer.
- virtual void `update` ()=0

Protected Attributes

- string `_observerName`
Name of the observer.

4.19.1 Detailed Description

A [Booster](#) observer class.

Author

Jonathan Enslin - u19103345

4.19.2 Member Function Documentation

4.19.2.1 setName()

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

Sets the name of the observer.

Parameters

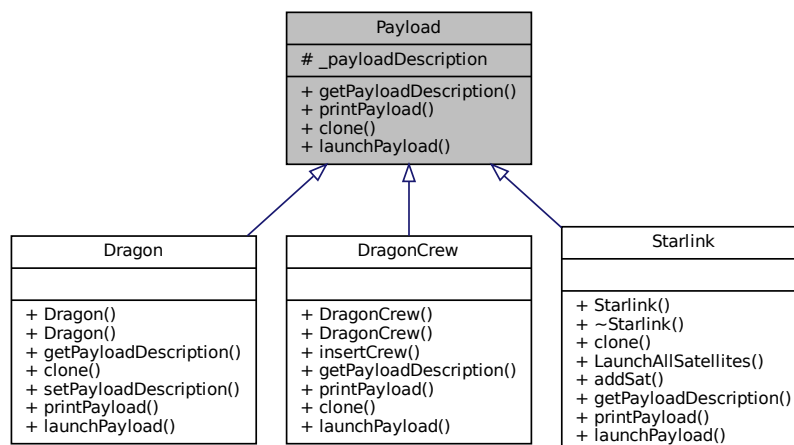
in	<i>name</i>	Name to be assigned
----	-------------	---------------------

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

- Propulsion/Observer.h
- Propulsion/Observer.cpp

4.20 Payload Class Reference

Inheritance diagram for Payload:



Collaboration diagram for Payload:

Payload
_payloadDescription
+ getPayloadDescription() + printPayload() + clone() + launchPayload()

Public Types

- enum [PayloadType](#) { [CREW](#), [CARGO](#), [STARLINK](#) }
Specifies different payload types.

Public Member Functions

- virtual string **getPayloadDescription** ()=0
- virtual void **printPayload** ()=0
- virtual [Payload](#) * **clone** ()=0
- virtual void **launchPayload** ()=0
Launches payload.

Protected Attributes

- string **_payloadDescription**

4.20.1 Member Enumeration Documentation

4.20.1.1 PayloadType

enum [Payload::PayloadType](#)

Specifies different payload types.

- [CREW](#) specifies that the payload carries crew
 - [CARGO](#) specifies that the payload carries cargo
 - [STARLINK](#) specifies that the payload carries starlink satellites

Enumerator

CREW	Crew type.
CARGO	Cargo type.
STARLINK	Starlink .

4.20.2 Member Function Documentation

4.20.2.1 launchPayload()

```
virtual void Payload::launchPayload ( ) [pure virtual]
```

Launches payload.

See child classes for implementation

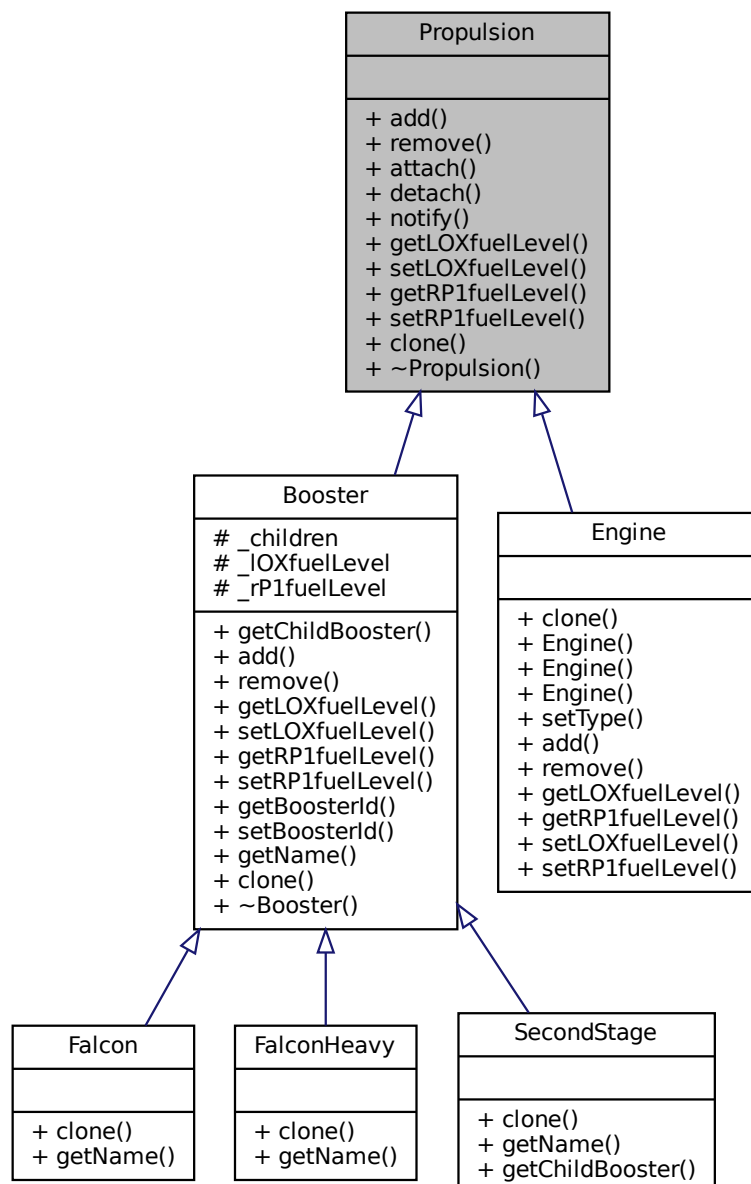
Implemented in [Starlink](#), [Dragon](#), and [DragonCrew](#).

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

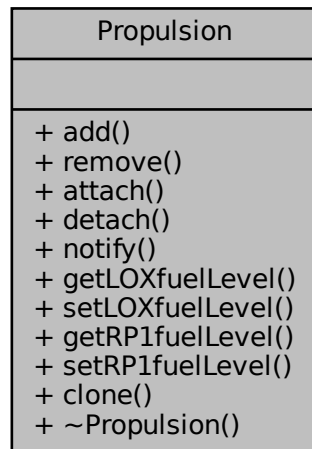
- Payload/Payload.h
- Payload/Payload.cpp

4.21 Propulsion Class Reference

Inheritance diagram for Propulsion:



Collaboration diagram for Propulsion:



Public Member Functions

- virtual void `add (Propulsion *aP)=0`
- virtual void `remove (Propulsion *aP)=0`
- void `attach (Observer *aO)`
- void `detach (Observer *aO)`
Removes an observer from the observer list.
- void `notify ()`
- virtual int `getLOXfuelLevel ()=0`
Getter for the LOXfuelLevel.
- virtual void `setLOXfuelLevel (int aLOXfuelLevel)=0`
Sets the LOX fuel level.
- virtual int `getRP1fuelLevel ()=0`
Getter for the RP1 fuel level.
- virtual void `setRP1fuelLevel (int aRP1fuelLevel)=0`
Setter for the RP1 fuel level.
- virtual `Propulsion * clone ()=0`
Clone function.
- virtual `~Propulsion ()`
default virtual destructor

4.21.1 Member Function Documentation

4.21.1.1 add()

```
void Propulsion::add (
    Propulsion * aP ) [pure virtual]
```

Parameters

in	<i>aP</i>	- The propulsion object to be added
----	-----------	-------------------------------------

see child classes for implementation details

Implemented in [Booster](#), and [Engine](#).

4.21.1.2 attach()

```
void Propulsion::attach (
    Observer * aO )
```

Adds an observer to the observer list

Parameters

in	<i>aO</i>	A pointer to the observer to be added to the observer list
----	-----------	--

Note

- [Observer](#) also has to register with respective object to function correctly

4.21.1.3 clone()

```
virtual Propulsion* Propulsion::clone ( ) [pure virtual]
```

Clone function.

Returns

Propulsion pointer

Implemented in [Booster](#), [Engine](#), [Falcon](#), [FalconHeavy](#), and [SecondStage](#).

4.21.1.4 detach()

```
void Propulsion::detach (
    Observer * aO )
```

Removes an observer from the observer list.

Parameters

<i>aO</i>	The observer to be removed from the observer list
-----------	---

4.21.1.5 getLOXfuelLevel()

```
virtual int Propulsion::getLOXfuelLevel ( ) [pure virtual]
```

Getter for the LOXfuelLevel.

Returns

the LOX fuel level value

Implemented in [Booster](#), and [Engine](#).

4.21.1.6 getRP1fuelLevel()

```
virtual int Propulsion::getRP1fuelLevel ( ) [pure virtual]
```

Getter for the RP1 fuel level.

Returns

RP1 fuel level

Implemented in [Booster](#), and [Engine](#).

4.21.1.7 notify()

```
void Propulsion::notify ( )
```

Notifies all observers in the observer list

4.21.1.8 remove()

```
void Propulsion::remove (
    Propulsion * aP ) [pure virtual]
```

Parameters

<i>in</i>	<i>aP</i>	- The propulsion object to be removed
-----------	-----------	---------------------------------------

see child classes for implementation details

Implemented in [Booster](#), and [Engine](#).

4.21.1.9 setLOXfuelLevel()

```
virtual void Propulsion::setLOXfuelLevel (
    int aLOXfuelLevel ) [pure virtual]
```

Sets the LOX fuel level.

Parameters

in	<i>aLOXfuelLevel</i>	The value LOX fuel level should be set to
----	----------------------	---

Implemented in [Engine](#), and [Booster](#).

4.21.1.10 setRP1fuelLevel()

```
virtual void Propulsion::setRP1fuelLevel (
    int aRP1fuelLevel ) [pure virtual]
```

Setter for the RP1 fuel level.

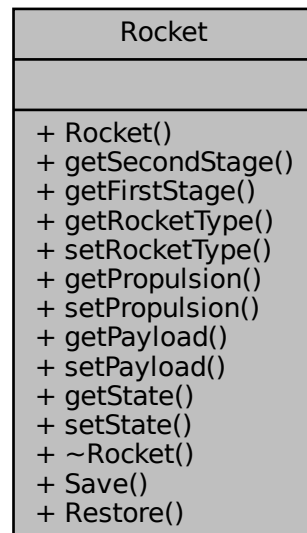
Implemented in [Engine](#), and [Booster](#).

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

- Propulsion/Propulsion.h
- Propulsion/Propulsion.cpp

4.22 Rocket Class Reference

Collaboration diagram for Rocket:



Public Types

- enum [RocketType](#) { [FALCON9](#), [FALCONHEAVY](#) }
Specifies the type of.

Public Member Functions

- [Booster](#) * [getSecondStage](#) ()
Get the Second Stage object.
- [Booster](#) * [getFirstStage](#) (int index)
Get the First Stage object at a certain index.
- [RocketType](#) [getRocketType](#) ()
- void [setRocketType](#) ([RocketType](#) type)
Sets the rocket type.
- [Propulsion](#) * [getPropulsion](#) ()
Returns the propulsion object of the rocket.
- void [setPropulsion](#) ([Propulsion](#) *aPropulsion)
sets the propulsion object of the rocket
- [Payload](#) * [getPayload](#) ()
gets the rocket's payload
- void [setPayload](#) ([Payload](#) *aPayload)
sets the rocket's payload
- [State](#) * [getState](#) ()

- gets the state object of the rocket*
- void **setState** ([State](#) *aState)
- sets the state object of the rocket*
- [RocketMemento](#) * **Save** ()
- void **Restore** ([RocketMemento](#) *aRockMem)

4.22.1 Member Enumeration Documentation

4.22.1.1 RocketType

enum [Rocket::RocketType](#)

Specifies the type of.

FALCON9 Rockets will have 1 first stage boosters FALCONHEAVY Rockets will have 3 first stage boosters

Enumerator

FALCON9	Specifies that the rocket is a Falcon-9 Rocket .
FALCONHEAVY	Specifies that the rocket is a Falcon Heavy Rocket .

4.22.2 Member Function Documentation

4.22.2.1 getFirstStage()

```
Booster * Rocket::getFirstStage (
    int index )
```

Get the First Stage object at a certain index.

Returns the index'th first stage booster

Parameters

<i>index</i>	
--------------	--

Returns

[Booster](#)*

4.22.2.2 getSecondStage()

```
Booster * Rocket::getSecondStage ( )
```

Get the Second Stage object.

Returns a pointer to the second stage of the rocket

Returns

SecondStage*

4.22.2.3 setRocketType()

```
void Rocket::setRocketType (
    RocketType type )
```

Sets the rocket type.

Note

- This should not be changed after rocket has been created

Parameters

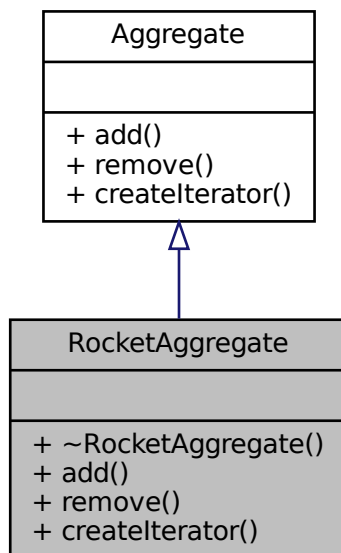
in	type	The type of the rocket
----	------	------------------------

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

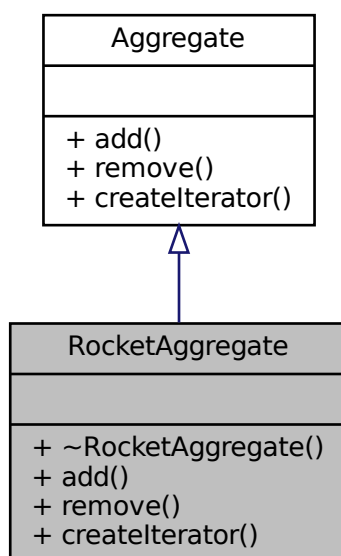
- Rocket.h
- Rocket.cpp

4.23 RocketAggregate Class Reference

Inheritance diagram for RocketAggregate:



Collaboration diagram for RocketAggregate:



Public Member Functions

- [~RocketAggregate](#) ()
Destruct the [RocketAggregate](#).
- void [add](#) ([RocketMemento](#) *aR)
Add a [RocketMemento](#) to the vector.
- void [remove](#) ([RocketMemento](#) *aR)
Remove a [RocketMemento](#) from the vector.
- [Iterator](#) * [createIterator](#) ()
Create an iterator.

4.23.1 Member Function Documentation

4.23.1.1 add()

```
void RocketAggregate::add (
    RocketMemento * aR ) [virtual]
```

Add a [RocketMemento](#) to the vector.

Parameters

in	<i>aR</i>	pointer to a RocketMemento object
----	-----------	---

Implements [Aggregate](#).

4.23.1.2 createIterator()

```
Iterator * RocketAggregate::createIterator ( ) [virtual]
```

Create an iterator.

Returns

newly created iterator

Implements [Aggregate](#).

4.23.1.3 remove()

```
void RocketAggregate::remove (
    RocketMemento * aR ) [virtual]
```

Remove a [RocketMemento](#) from the vector.

Parameters

in	<i>aR</i>	pointer to a RocketMemento object
----	-----------	---

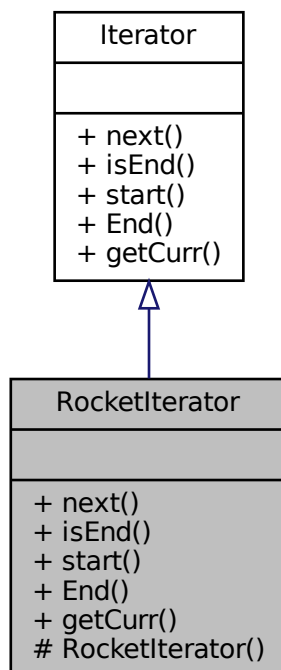
Implements [Aggregate](#).

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

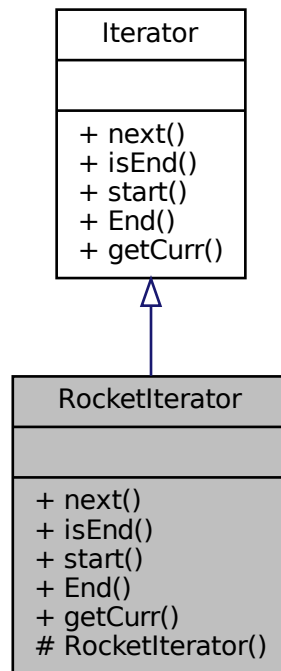
- Storage/RocketAggregate.h
- Storage/RocketAggregate.cpp

4.24 RocketIterator Class Reference

Inheritance diagram for RocketIterator:



Collaboration diagram for RocketIterator:



Public Member Functions

- [RocketMemento](#) * [next](#) ()
Move on to next element.
- bool [isEnd](#) ()
Check if currently at last element.
- [RocketMemento](#) * [start](#) ()
Move to the first element.
- [RocketMemento](#) * [End](#) ()
Move to the last element.
- [RocketMemento](#) * [getCurr](#) ()
Returns the current element.

Protected Member Functions

- [RocketIterator](#) (vector< [RocketMemento](#) * > *c)
A parameterized constructor.

Friends

- class [RocketAggregate](#)

4.24.1 Constructor & Destructor Documentation

4.24.1.1 RocketIterator()

```
RocketIterator::RocketIterator (
    vector< RocketMemento * > * c ) [protected]
```

A parameterized constructor.

Parameters

<code>c</code>	a pointer to a vector containing pointers to RocketMemento objects
----------------	--

4.24.2 Member Function Documentation

4.24.2.1 End()

```
RocketMemento * RocketIterator::End ( ) [virtual]
```

Move to the last element.

Returns

The last element

Implements [Iterator](#).

4.24.2.2 getCurr()

```
RocketMemento * RocketIterator::getCurr ( ) [virtual]
```

Returns the current element.

Returns

element that is currently being pointed to

Implements [Iterator](#).

4.24.2.3 isEnd()

```
bool RocketIterator::isEnd ( ) [virtual]
```

Check if currently at last element.

Returns

return true if at last element, else return false

Implements [Iterator](#).

4.24.2.4 next()

```
RocketMemento * RocketIterator::next ( ) [virtual]
```

Move on to next element.

Returns

element that is being pointed to after move

Implements [Iterator](#).

4.24.2.5 start()

```
RocketMemento * RocketIterator::start ( ) [virtual]
```

Move to the first element.

Returns

The first element

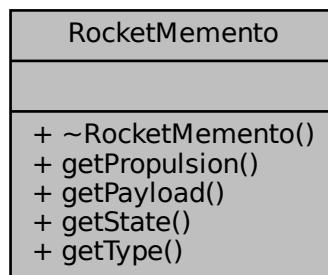
Implements [Iterator](#).

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

- Storage/RocketIterator.h
- Storage/RocketIterator.cpp

4.25 RocketMemento Class Reference

Collaboration diagram for RocketMemento:



Public Member Functions

- [Propulsion](#) * **getPropulsion** ()
- [Payload](#) * **getPayload** ()
- [State](#) * **getState** ()
- [Rocket::RocketType](#) **getType** ()

Friends

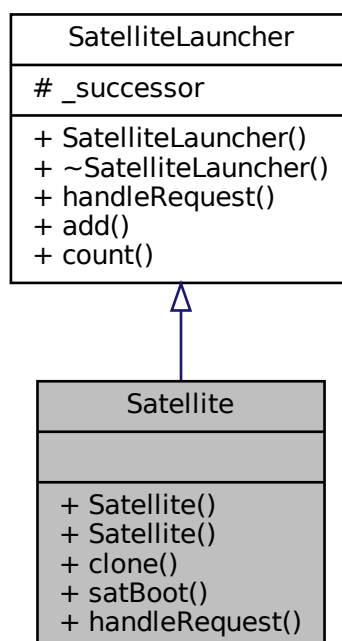
- class **Rocket**

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

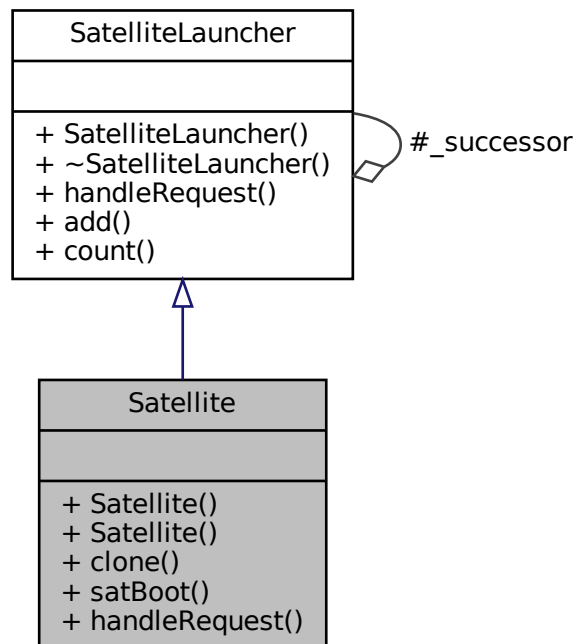
- Storage/RocketMemento.h
- Storage/RocketMemento.cpp

4.26 Satellite Class Reference

Inheritance diagram for Satellite:



Collaboration diagram for Satellite:



Public Member Functions

- [Satellite](#) ()
Construct a new [Satellite::Satellite](#) object.
- [Satellite](#) (const [Satellite](#) &obj)
Construct a new [Satellite::Satellite](#) object.
- [Satellite](#) * [clone](#) ()
Clones satellite and returns new one.
- void [satBoot](#) (int v)
- void [handleRequest](#) (int number=0)
[SatelliteLauncher](#) goes down the chain of Satellites. Once it reaches the end they detach and delete.

Additional Inherited Members

4.26.1 Constructor & Destructor Documentation

4.26.1.1 [Satellite](#)() [1/2]

`Satellite::Satellite ()`

Construct a new [Satellite::Satellite](#) object.

Creates normal [Satellite](#).

4.26.1.2 Satellite() [2/2]

```
Satellite::Satellite (
    const Satellite & obj )
```

Construct a new [Satellite::Satellite](#) object.

Copy constructor to make copied Satellites from existing Satellites.

Parameters

<i>obj</i>	
------------	--

4.26.2 Member Function Documentation

4.26.2.1 clone()

```
Satellite * Satellite::clone ( )
```

Clones satellite and returns new one.

Creates a cloned [Satellite](#) when a [Satellite](#) calls to clone itself.

Returns

[Satellite](#)*

4.26.2.2 handleRequest()

```
void Satellite::handleRequest (
    int number = 0 ) [virtual]
```

[SatelliteLauncher](#) goes down the chain of Satellites. Once it reaches the end they detach and delete.

Once [handleRequest\(\)](#) is called. Starting from the head the list of Satellites is traversed until the very end. At the end through recursion the ends get deleted

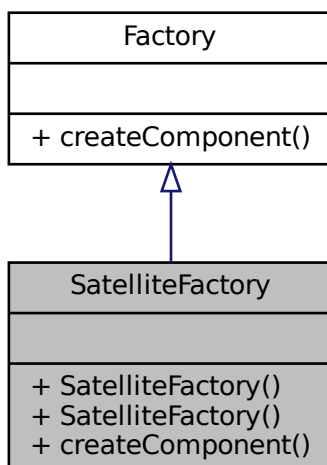
Implements [SatelliteLauncher](#).

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

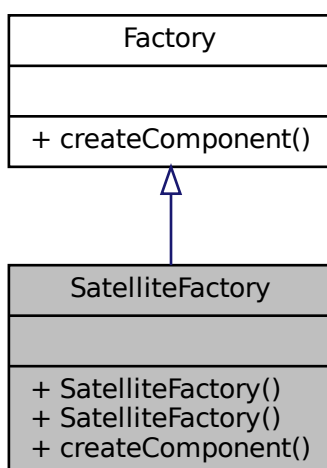
- Payload/Satellite.h
- Payload/Satellite.cpp

4.27 SatelliteFactory Class Reference

Inheritance diagram for SatelliteFactory:



Collaboration diagram for SatelliteFactory:



Public Member Functions

- [SatelliteFactory](#) ()

- Construct a new [SatelliteFactory::SatelliteFactory](#) object.
- [SatelliteFactory](#) (const [SatelliteFactory](#) &obj)
Construct a new [SatelliteFactory::SatelliteFactory](#) object.
- [SatelliteLauncher](#) * [createComponent](#) ()
Factory makes new satellite for user.

4.27.1 Constructor & Destructor Documentation

4.27.1.1 [SatelliteFactory\(\)](#) [1/2]

```
SatelliteFactory::SatelliteFactory ( )
```

Construct a new [SatelliteFactory::SatelliteFactory](#) object.

Creates normal [SatelliteFactory](#) object used to construct other Satellites.

4.27.1.2 [SatelliteFactory\(\)](#) [2/2]

```
SatelliteFactory::SatelliteFactory (
    const SatelliteFactory & obj )
```

Construct a new [SatelliteFactory::SatelliteFactory](#) object.

Copy constructor used to copy existing SatelliteFactories Simply creates another instance.

Parameters

<i>obj</i>	
------------	--

4.27.2 Member Function Documentation

4.27.2.1 [createComponent\(\)](#)

```
SatelliteLauncher * SatelliteFactory::createComponent ( ) [virtual]
```

Factory makes new satellite for user.

Used to create satellites on demand for use. Creates a satellite and returns it to user.

Returns

SatelliteLauncher*

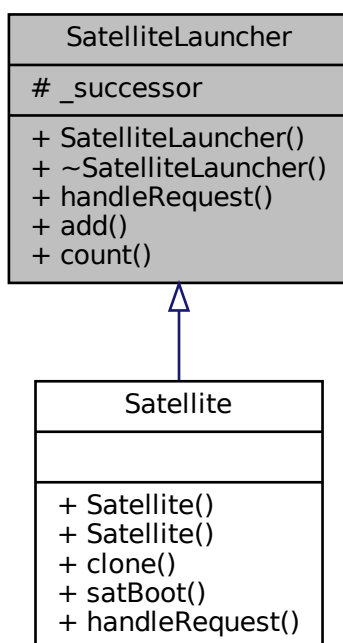
Implements [Factory](#).

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

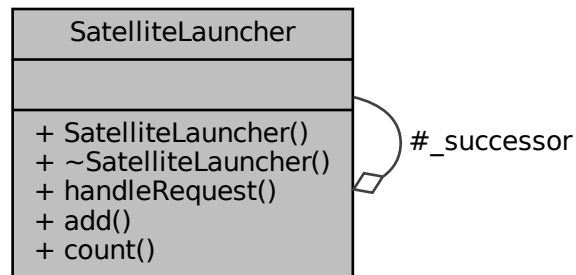
- Payload/SatelliteFactory.h
- Payload/SatelliteFactory.cpp

4.28 SatelliteLauncher Class Reference

Inheritance diagram for SatelliteLauncher:



Collaboration diagram for SatelliteLauncher:



Public Member Functions

- [SatelliteLauncher](#) ()
Construct a new [SatelliteLauncher::SatelliteLauncher](#) object.
- [~SatelliteLauncher](#) ()
Destroy the [SatelliteLauncher::SatelliteLauncher](#) object.
- virtual void **handleRequest** (int number=0)=0
- void [add](#) ([SatelliteLauncher](#) *satellite)
Adds a satellite to the chain.
- int [count](#) ()
Counts all satellites in the chain.

Protected Attributes

- [SatelliteLauncher](#) * **_successor**

4.28.1 Constructor & Destructor Documentation

4.28.1.1 SatelliteLauncher()

```
SatelliteLauncher::SatelliteLauncher ( )
```

Construct a new [SatelliteLauncher::SatelliteLauncher](#) object.

Parent Constructor for [Satellite](#) children. Sets successor to NULL.

4.28.1.2 ~SatelliteLauncher()

```
SatelliteLauncher::~SatelliteLauncher ( )
```

Destroy the [SatelliteLauncher::SatelliteLauncher](#) object.

Destructor used to delete successor objects. Used for deletion when object may no longer be used.

4.28.2 Member Function Documentation

4.28.2.1 add()

```
void SatelliteLauncher::add (
    SatelliteLauncher * satellite )
```

Adds a satellite to the chain.

Adds a satellite to the current list of satellites with use of recursion to find end. Once the end is found, new satellite is added to the end.

Parameters

<i>satellite</i>	
------------------	--

4.28.2.2 count()

```
int SatelliteLauncher::count ( )
```

Counts all satellites in the chain.

Through recursion the list is traversed incremented by 1 on each successful function call. Allows the satellites to be counted.

Returns

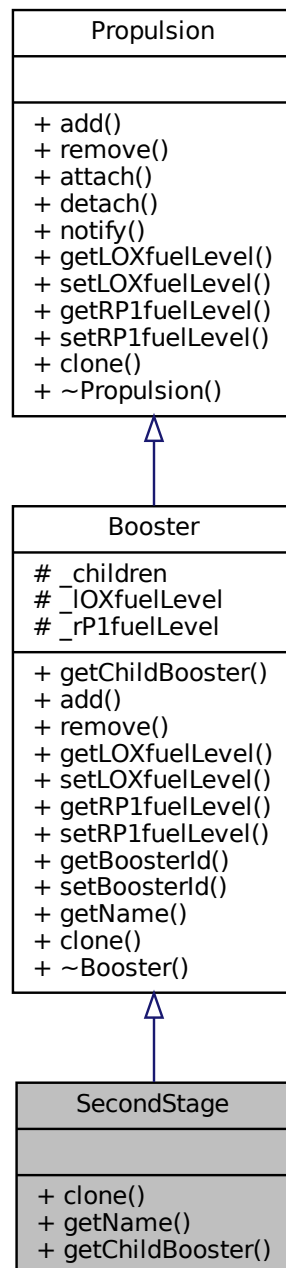
int

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

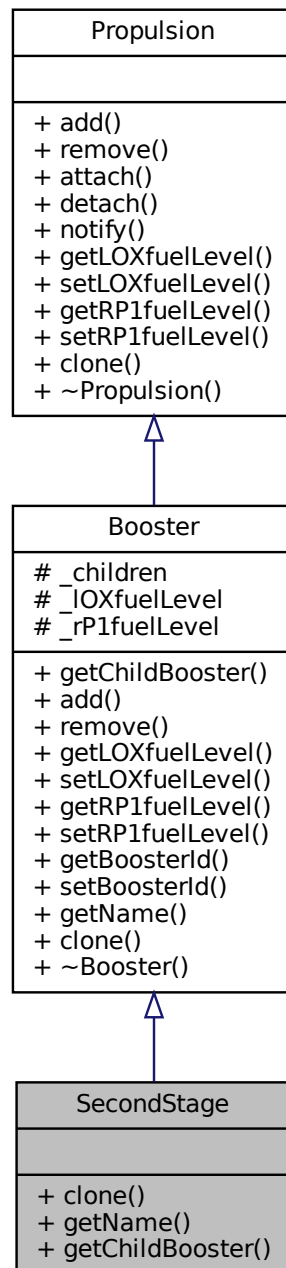
- Payload/SatelliteLauncher.h
- Payload/SatelliteLauncher.cpp

4.29 SecondStage Class Reference

Inheritance diagram for SecondStage:



Collaboration diagram for SecondStage:



Public Member Functions

- [Propulsion](#) * [clone](#) ()
Returns a clone of the entire tree structure.
- string [getName](#) () override
Getter for the booster name.
- [Booster](#) * [getChildBooster](#) (int index) override
Get the Child [Booster](#) object.

Additional Inherited Members

4.29.1 Member Function Documentation

4.29.1.1 clone()

```
Propulsion * SecondStage::clone ( ) [virtual]
```

Returns a clone of the entire tree structure.

Returns

Propulsion*

Reimplemented from [Booster](#).

4.29.1.2 getChildBooster()

```
Booster * SecondStage::getChildBooster (
    int index ) [override], [virtual]
```

Get the Child [Booster](#) object.

Gets a booster from the child, taken from children[index + 1], thus skipping over engine

Parameters

<i>index</i>	The index of the child booster to obtain
--------------	--

Returns

Booster*

Note

- This should only be used on a a Rocket/Booster that adheres to the structure made by a [ConcreteBuilder](#)

Reimplemented from [Booster](#).

4.29.1.3 getName()

```
string SecondStage::getName ( ) [override], [virtual]
```

Getter for the booster name.

Returns

booster name as string

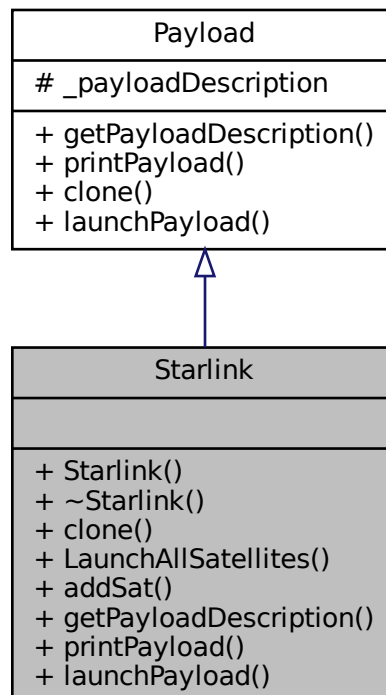
Reimplemented from [Booster](#).

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

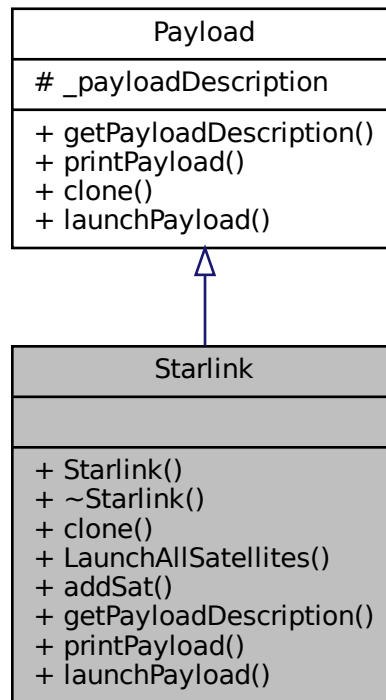
- Propulsion/SecondStage.h
- Propulsion/SecondStage.cpp

4.30 Starlink Class Reference

Inheritance diagram for Starlink:



Collaboration diagram for Starlink:



Public Member Functions

- [Starlink \(\)](#)
Construct a new [Starlink::Starlink](#) object.
- [~Starlink \(\)](#)
Destroy the [Starlink::Starlink](#) object Delete the array of satelliteLaunchers.
- [Payload * clone \(\)](#)
[Starlink](#) clones itself.
- void [LaunchAllSatellites \(\)](#)
Simulates the the ejection by calling the head [Satellite](#) to Handle the request which goes down the chain deleting and detaching satellites.
- void [addSat \(SatelliteLauncher *Sat\)](#)
Add satellite to the chain.
- string [getPayloadDescription \(\)](#)
Gets the [Starlink](#) payload as a string.
- void [printPayload \(\)](#)
Displays the [Starlink](#) payload to the user.
- void [launchPayload \(\)](#)
Launches the starlink payload.

Additional Inherited Members

4.30.1 Constructor & Destructor Documentation

4.30.1.1 Starlink()

```
Starlink::Starlink ( )
```

Construct a new [Starlink::Starlink](#) object.

Creates the [Starlink](#) payload. Presets the payload description accordingly and then head of the list is set to NULL.

4.30.1.2 ~Starlink()

```
Starlink::~~Starlink ( )
```

Destroy the [Starlink::Starlink](#) object Delete the array of satelliteLaunchers.

Destructor of [Starlink](#) payload. Since [Starlink](#) used dynamic objects we have to delete each object. Call `handleRequest()` to delete each object and nullify all locations.

4.30.2 Member Function Documentation

4.30.2.1 addSat()

```
void Starlink::addSat (
    SatelliteLauncher * Sat )
```

Add satellite to the chain.

Add satellites to the chain. With use of recursion the addition goes down the chain until free position is taken.

Parameters

<i>Sat</i>	
------------	--

4.30.2.2 clone()

```
Payload * Starlink::clone ( ) [virtual]
```

[Starlink](#) clones itself.

[Starlink](#) clones itself and makes a brand new [Starlink](#) with all the same variables that are not shallow copied.

Returns

[Starlink](#)*

Implements [Payload](#).

4.30.2.3 `getPayloadDescription()`

```
string Starlink::getPayloadDescription ( ) [virtual]
```

Gets the [Starlink](#) payload as a string.

Makes a string of the description with use of some count() to find the current number of Satellites.

Returns

string

Implements [Payload](#).

4.30.2.4 `LaunchAllSatellites()`

```
void Starlink::LaunchAllSatellites ( )
```

Simulates the the ejection by calling the head [Satellite](#) to Handle the request which goes down the chain deleting and detaching satellites.

Simulates the launch of all Satellites by calling `handleRequest()`, but with the added functionality of checking whether satellites exist or not. This empties the payload of satellites and detaches aswell as deletes each object. Displays to the user whether or not Satellites released or not.

4.30.2.5 `launchPayload()`

```
void Starlink::launchPayload ( ) [virtual]
```

Launches the starlink payload.

This will deploy all starlink satellites

Implements [Payload](#).

4.30.2.6 printPayload()

```
void Starlink::printPayload ( ) [virtual]
```

Displays the [Starlink](#) payload to the user.

Calls [getPayloadDescription\(\)](#) and takes the already structured string to display to the user.

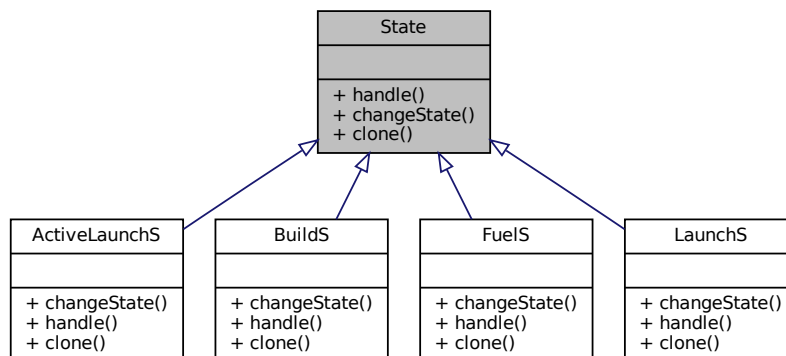
Implements [Payload](#).

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

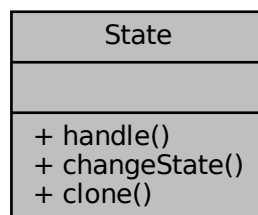
- Payload/Starlink.h
- Payload/Starlink.cpp

4.31 State Class Reference

Inheritance diagram for State:



Collaboration diagram for State:



Public Member Functions

- virtual void [handle](#) ([Rocket](#) *aR)=0
This function handles state specific functionality.
- virtual void [changeState](#) ([Rocket](#) *aR)=0
changes the state of the rocket
- virtual [State](#) * [clone](#) ()=0
returns a clone of the current state

4.31.1 Member Function Documentation

4.31.1.1 [changeState\(\)](#)

```
virtual void State::changeState (  
    Rocket * aR ) [pure virtual]
```

changes the state of the rocket

Parameters

<i>aR</i>	The rocket which is changing state
-----------	------------------------------------

See children classes for details

Implemented in [BuildS](#), [ActiveLaunchS](#), [FuelS](#), and [LaunchS](#).

4.31.1.2 [handle\(\)](#)

```
virtual void State::handle (  
    Rocket * aR ) [pure virtual]
```

This function handles state specific functionality.

Parameters

<i>aR</i>	The rocket that should be used for handling
-----------	---

see children classes for details

Implemented in [BuildS](#), [ActiveLaunchS](#), [FuelS](#), and [LaunchS](#).

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

- [State/State.h](#)

Index

- ~Booster
 - Booster, [14](#)
- ~SatelliteLauncher
 - SatelliteLauncher, [85](#)
- ~Starlink
 - Starlink, [92](#)
- ActiveLaunchS, [7](#)
 - changeState, [8](#)
 - handle, [9](#)
- add
 - Aggregate, [10](#)
 - Booster, [14](#)
 - Propulsion, [65](#)
 - RocketAggregate, [73](#)
 - SatelliteLauncher, [86](#)
- addSat
 - Starlink, [92](#)
- Aggregate, [9](#)
 - add, [10](#)
 - createIterator, [10](#)
 - remove, [11](#)
- attach
 - Propulsion, [66](#)
- batchStore
 - Caretaker, [24](#)
- Booster, [11](#)
 - ~Booster, [14](#)
 - add, [14](#)
 - clone, [15](#)
 - getBoosterId, [15](#)
 - getChildBooster, [15](#)
 - getLOXfuelLevel, [16](#)
 - getName, [16](#)
 - getRP1fuelLevel, [16](#)
 - remove, [16](#)
 - setBoosterId, [17](#)
 - setLOXfuelLevel, [17](#)
 - setRP1fuelLevel, [17](#)
- Builder, [18](#)
 - buildRocket, [20](#)
 - getRocketType, [20](#)
 - setFirstStageBoosters, [20](#)
 - setPayloadType, [20](#)
 - setRocketType, [21](#)
 - setSecondStage, [21](#)
- buildRocket
 - Builder, [20](#)
 - ConcreteBuilder, [28](#)
- BuildS, [22](#)
 - changeState, [23](#)
 - handle, [23](#)
- Caretaker, [24](#)
 - batchStore, [24](#)
 - createIterator, [25](#)
 - RestoreLast, [25](#)
 - storeRocket, [25](#)
- CARGO
 - Payload, [63](#)
- changeState
 - ActiveLaunchS, [8](#)
 - BuildS, [23](#)
 - FuelS, [53](#)
 - LaunchS, [57](#)
 - State, [95](#)
- clone
 - Booster, [15](#)
 - Dragon, [33](#)
 - DragonCrew, [36](#)
 - Engine, [40](#)
 - Falcon, [45](#)
 - FalconHeavy, [48](#)
 - Propulsion, [66](#)
 - Satellite, [81](#)
 - SecondStage, [89](#)
 - Starlink, [92](#)
- ConcreteBuilder, [26](#)
 - buildRocket, [28](#)
 - ConcreteBuilder, [28](#)
 - setFirstStageBoosters, [28](#)
 - setPayload, [29, 30](#)
 - setSecondStage, [30](#)
- construct
 - MissionControlStarbase, [59](#)
- count
 - SatelliteLauncher, [86](#)
- createComponent
 - SatelliteFactory, [83](#)
- createIterator
 - Aggregate, [10](#)
 - Caretaker, [25](#)
 - RocketAggregate, [73](#)
- CREW
 - Payload, [63](#)
- detach
 - Propulsion, [66](#)
- Dragon, [31](#)

- clone, 33
- Dragon, 32, 33
- getPayloadDescription, 33
- launchPayload, 33
- printPayload, 34
- DragonCrew, 34
 - clone, 36
 - DragonCrew, 35, 36
 - getPayloadDescription, 36
 - insertCrew, 37
 - launchPayload, 37
 - printPayload, 37
- End
 - Iterator, 55
 - RocketIterator, 76
- Engine, 38
 - clone, 40
 - Engine, 40
 - setType, 40
- Factory, 41
- Falcon, 43
 - clone, 45
 - getName, 45
- FALCON9
 - Rocket, 70
- FALCONHEAVY
 - Rocket, 70
- FalconHeavy, 46
 - clone, 48
 - getName, 48
- FuelObserver, 49
 - FuelObserver, 50
 - setSubjectBooster, 50
 - update, 51
- FuelS, 52
 - changeState, 53
 - handle, 53
- getBoosterId
 - Booster, 15
- getChildBooster
 - Booster, 15
 - SecondStage, 89
- getCurr
 - Iterator, 55
 - RocketIterator, 76
- getFirstStage
 - Rocket, 70
- getLOXfuelLevel
 - Booster, 16
 - Propulsion, 67
- getName
 - Booster, 16
 - Falcon, 45
 - FalconHeavy, 48
 - SecondStage, 89
- getPayloadDescription
 - Dragon, 33
 - DragonCrew, 36
 - Starlink, 93
- getRocketType
 - Builder, 20
- getRP1fuelLevel
 - Booster, 16
 - Propulsion, 67
- getSecondStage
 - Rocket, 70
- handle
 - ActiveLaunchS, 9
 - BuildS, 23
 - FuelS, 53
 - LaunchS, 58
 - State, 95
- handleRequest
 - Satellite, 81
- insertCrew
 - DragonCrew, 37
- isEnd
 - Iterator, 55
 - RocketIterator, 76
- Iterator, 54
 - End, 55
 - getCurr, 55
 - isEnd, 55
 - next, 55
 - start, 56
- launch
 - MissionControlStarbase, 59
- LaunchAllSatellites
 - Starlink, 93
- launchPayload
 - Dragon, 33
 - DragonCrew, 37
 - Payload, 63
 - Starlink, 93
- LaunchS, 56
 - changeState, 57
 - handle, 58
- MissionControlStarbase, 58
 - construct, 59
 - launch, 59
- next
 - Iterator, 55
 - RocketIterator, 77
- notify
 - Propulsion, 67
- Observer, 59
 - setName, 61
- Payload, 61
 - CARGO, 63

- CREW, 63
- launchPayload, 63
- PayloadType, 62
- STARLINK, 63
- PayloadType
 - Payload, 62
- printPayload
 - Dragon, 34
 - DragonCrew, 37
 - Starlink, 93
- Propulsion, 64
 - add, 65
 - attach, 66
 - clone, 66
 - detach, 66
 - getLOXfuelLevel, 67
 - getRP1fuelLevel, 67
 - notify, 67
 - remove, 67
 - setLOXfuelLevel, 68
 - setRP1fuelLevel, 68
- remove
 - Aggregate, 11
 - Booster, 16
 - Propulsion, 67
 - RocketAggregate, 73
- RestoreLast
 - Caretaker, 25
- Rocket, 69
 - FALCON9, 70
 - FALCONHEAVY, 70
 - getFirstStage, 70
 - getSecondStage, 70
 - RocketType, 70
 - setRocketType, 71
- RocketAggregate, 72
 - add, 73
 - createIterator, 73
 - remove, 73
- RocketIterator, 74
 - End, 76
 - getCurr, 76
 - isEnd, 76
 - next, 77
 - RocketIterator, 76
 - start, 77
- RocketMemento, 78
- RocketType
 - Rocket, 70
- Satellite, 79
 - clone, 81
 - handleRequest, 81
 - Satellite, 80
- SatelliteFactory, 82
 - createComponent, 83
 - SatelliteFactory, 83
- SatelliteLauncher, 84
 - ~SatelliteLauncher, 85
 - add, 86
 - count, 86
 - SatelliteLauncher, 85
- SecondStage, 87
 - clone, 89
 - getChildBooster, 89
 - getName, 89
- setBoosterId
 - Booster, 17
- setFirstStageBoosters
 - Builder, 20
 - ConcreteBuilder, 28
- setLOXfuelLevel
 - Booster, 17
 - Propulsion, 68
- setName
 - Observer, 61
- setPayload
 - ConcreteBuilder, 29, 30
- setPayloadType
 - Builder, 20
- setRocketType
 - Builder, 21
 - Rocket, 71
- setRP1fuelLevel
 - Booster, 17
 - Propulsion, 68
- setSecondStage
 - Builder, 21
 - ConcreteBuilder, 30
- setSubjectBooster
 - FuelObserver, 50
- setType
 - Engine, 40
- STARLINK
 - Payload, 63
- Starlink, 90
 - ~Starlink, 92
 - addSat, 92
 - clone, 92
 - getPayloadDescription, 93
 - LaunchAllSatellites, 93
 - launchPayload, 93
 - printPayload, 93
 - Starlink, 92
- start
 - Iterator, 56
 - RocketIterator, 77
- State, 94
 - changeState, 95
 - handle, 95
- storeRocket
 - Caretaker, 25
- update
 - FuelObserver, 51