# VehicleTestbed

Generated by Doxygen 1.8.14

# Contents

# 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

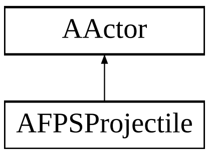
Here is a list of all documented files with brief descriptions:

# Chapter 4

# Class Documentation

## 4.1 AFPSProjectile Class Reference

Inheritance diagram for AFPSProjectile:



### Public Member Functions

- AFPSProjectile ()

    *Constructor for the AFPSProjectile class*
- USphereComponent ∗ GetCollisionComp () const
- UProjectileMovementComponent ∗ **GetProjectileMovement** () const

### Protected Attributes

- USphereComponent ∗ **CollisionComp**
- UProjectileMovementComponent ∗ **ProjectileMovement**

### 4.1.1 Constructor & Destructor Documentation

#### 4.1.1.1 AFPSProjectile()

```
AFPSProjectile::AFPSProjectile ( )
```

Constructor for the AFPSProjectile class

summary>Gets the collision component of the projectile

returns>A pointer to the CollisionComp

### 4.1.2 Member Function Documentation

#### 4.1.2.1 GetCollisionComp()

```
USphereComponent* AFPSProjectile::GetCollisionComp ( ) const  [inline]
```

summary>Gets the projectile movement of the projectile

returns>A pointer to the ProjectileMovement

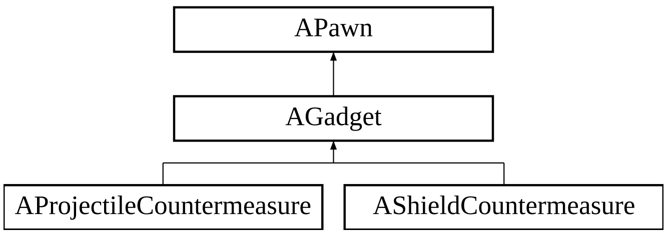
The documentation for this class was generated from the following files:

- Source/VehicleTestbed/Public/FPSProjectile.h
- Source/VehicleTestbed/Private/FPSProjectile.cpp

## 4.2 AGadget Class Reference

```
#include <Gadget.h>
```

Inheritance diagram for AGadget:



### Public Member Functions

- AGadget ()
    *Default Constructor*
- ∼AGadget ()
    *summary>Activates the countermeasure based on desired behaviour*
- virtual void **Activate** ()

### Protected Member Functions

- virtual void **InitialiseMesh** ()

### Protected Attributes

- wchar_t ∗ **MeshLocation**
- wchar_t ∗ PhysicsAssetLocation
    *summary>Initializes the asset to use the appropriate mesh and physics assets*

### 4.2.1 Detailed Description

Base class for mounting/dismounting to an AMountablePAwn

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 AGadget()

`AGadget::AGadget ( )`

Default Constructor

summary>Default Deconstructor

#### 4.2.2.2 ∼AGadget()

`AGadget::∼AGadget ( )`

summary>Activates the countermeasure based on desired behaviour

### 4.2.3 Member Data Documentation

#### 4.2.3.1 PhysicsAssetLocation

`wchar_t* AGadget::PhysicsAssetLocation [protected]`

summary>Initializes the asset to use the appropriate mesh and physics assets

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

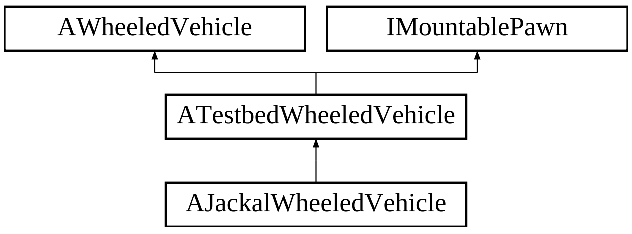
- Source/VehicleTestbed/Public/Gadgets/Gadget.h
- Source/VehicleTestbed/Private/Gadgets/Gadget.cpp

## 4.3 AJackalWheeledVehicle Class Reference

Class to represent the Clearpath Jackal.

`#include <JackalWheeledVehicle.h>`

Inheritance diagram for AJackalWheeledVehicle:



### Public Member Functions

- AJackalWheeledVehicle ()

    *Default Constructor*
- ∼AJackalWheeledVehicle ()

    *summary>Allow actors to initialize themselves on the C++ side*
- void **PostInitializeComponents** ()

### Additional Inherited Members

### 4.3.1 Detailed Description

Class to represent the Clearpath Jackal.

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 AJackalWheeledVehicle()

```
AJackalWheeledVehicle::AJackalWheeledVehicle ( )
```

Default Constructor

summary>Default Deconstructor

#### 4.3.2.2 ∼AJackalWheeledVehicle()

```
AJackalWheeledVehicle::∼AJackalWheeledVehicle ( )
```

summary>Allow actors to initialize themselves on the C++ side

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

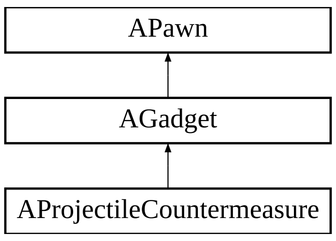
- Source/VehicleTestbed/Public/Vehicles/JackalWheeledVehicle.h
- Source/VehicleTestbed/Private/Vehicles/JackalWheeledVehicle.cpp

## 4.4 AProjectileCountermeasure Class Reference

Gadget that represents a projectile firing countermeasure that can be mounted to a MountablePawn

```
#include <ProjectileCountermeasure.h>
```

Inheritance diagram for AProjectileCountermeasure:



## Public Member Functions

- AProjectileCountermeasure ()

  *Default Constructor*
- ∼AProjectileCountermeasure ()

  *summary>Activates the countermeasure based on desired behaviour*
- virtual void **Activate** ()

## Protected Member Functions

- virtual void **InitialiseMesh** () override

## Protected Attributes

- wchar_t ∗ **MeshLocation** = TEXT("SkeletalMesh'/Game/Vehicle/Countermeasures/JackalProjectileCM.←JackalProjectileCM'")
- wchar_t ∗ PhysicsAssetLocation = TEXT("PhysicsAsset'/Game/Vehicle/Countermeasures/JackalProjectile←CM_PhysicsAsset.JackalProjectileCM_PhysicsAsset'")

  *summary>Initializes the asset to use the appropriate mesh and physics assets*

## 4.4.1 Detailed Description

Gadget that represents a projectile firing countermeasure that can be mounted to a MountablePawn

## 4.4.2 Constructor & Destructor Documentation

### 4.4.2.1 AProjectileCountermeasure()

```
AProjectileCountermeasure::AProjectileCountermeasure ( )
```

Default Constructor

summary>Default Deconstructor

### 4.4.2.2 ∼AProjectileCountermeasure()

```
AProjectileCountermeasure::∼AProjectileCountermeasure ( )
```

summary>Activates the countermeasure based on desired behaviour

## 4.4.3 Member Data Documentation

#### 4.4.3.1 PhysicsAssetLocation

```
wchar_t* AProjectileCountermeasure::PhysicsAssetLocation = TEXT("PhysicsAsset'/Game/Vehicle/Countermeasures/Ja
ProjectileCM_PhysicsAsset.JackalProjectileCM_PhysicsAsset'")  [protected]
```

summary>Initializes the asset to use the appropriate mesh and physics assets

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

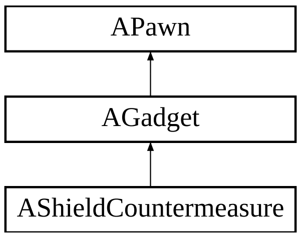
- Source/VehicleTestbed/Public/Gadgets/ProjectileCountermeasure.h
- Source/VehicleTestbed/Private/Gadgets/ProjectileCountermeasure.cpp

## 4.5 AShieldCountermeasure Class Reference

Gadget that represents a shield countermeasure that can be mounted to a MountablePawn

`#include <ShieldCountermeasure.h>`

Inheritance diagram for AShieldCountermeasure:



### Public Member Functions

- AShieldCountermeasure ()

  *Default Constructor*

### Protected Member Functions

- virtual void **InitialiseMesh** () override

### Protected Attributes

- wchar_t ∗ **MeshLocation** = TEXT("StaticMesh'/Game/Vehicle/Countermeasures/JackalShield.Jackal↩Shield'")
- wchar_t ∗ PhysicsAssetLocation = TEXT("PhysicsAsset'/Game/Vehicle/Jackal/Jackal_PhysicsAsset.Jackal↩_PhysicsAsset'")

  *summary>Initializes the asset to use the appropriate mesh and physics assets*

### 4.5.1 Detailed Description

Gadget that represents a shield countermeasure that can be mounted to a MountablePawn

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 AShieldCountermeasure()

```
AShieldCountermeasure::AShieldCountermeasure ( )
```

Default Constructor

summary>Default Deconstructor

### 4.5.3 Member Data Documentation

#### 4.5.3.1 PhysicsAssetLocation

```
wchar_t* AShieldCountermeasure::PhysicsAssetLocation = TEXT("PhysicsAsset'/Game/Vehicle/Jackal/Jackal↩
_PhysicsAsset.Jackal_PhysicsAsset'")  [protected]
```

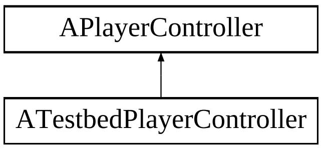
summary>Initializes the asset to use the appropriate mesh and physics assets

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

- Source/VehicleTestbed/Public/Gadgets/ShieldCountermeasure.h
- Source/VehicleTestbed/Private/Gadgets/ShieldCountermeasure.cpp

## 4.6 ATestbedPlayerController Class Reference

Inheritance diagram for ATestbedPlayerController:



### Public Member Functions

- ATestbedPlayerController ()

  *summary>Allows the PlayerController to set up custom input bindings*
- virtual void SetupInputComponent () override

  *summary>Event that is called when play begins for this actor*
- virtual void **BeginPlay** () override

## Protected Attributes

- UPauseMenuComponent ∗ **PauseMenuComponent**
- UPawnSwapComponent ∗ **PawnSwapComponent**
- UJackalControlComponent ∗ **JackalControlComponent**

### 4.6.1 Constructor & Destructor Documentation

#### 4.6.1.1 ATestbedPlayerController()

```
ATestbedPlayerController::ATestbedPlayerController ( )
```

summary>Allows the PlayerController to set up custom input bindings

### 4.6.2 Member Function Documentation

#### 4.6.2.1 SetupInputComponent()

```
void ATestbedPlayerController::SetupInputComponent ( )  [override], [virtual]
```

summary>Event that is called when play begins for this actor

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

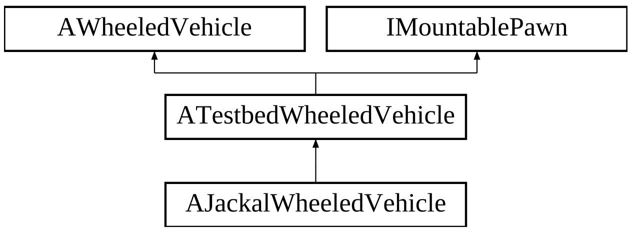
- Source/VehicleTestbed/Public/PlayerController/TestbedPlayerController.h
- Source/VehicleTestbed/Private/PlayerController/TestbedPlayerController.cpp

## 4.7 ATestbedWheeledVehicle Class Reference

Base class for all wheeled vehicle actors. Sets up cameras and controls for player usage.

```
#include <TestbedWheeledVehicle.h>
```

Inheritance diagram for ATestbedWheeledVehicle:



## Public Member Functions

- ATestbedWheeledVehicle ()

  *Default Constructor*
- ∼ATestbedWheeledVehicle ()

  *summary>Allow actors to initialize themselves on the C++ side*
- void PostInitializeComponents ()
- void SetThrottleInput (float Value)
- void SetSteeringInput (float Value)
- void SetBrakeInput (float Value)
- float GetVehicleForwardSpeed () const

  *summary>Gets first person view camera for the vehicle*
- void SwitchToOverheadCamera ()

  *summary>Gets first person view camera for the vehicle*
- void SwitchToInternalCamera ()

  *summary>Gets first person view camera for the vehicle*
- void SwitchToChaseCamera ()

  *summary>Returns true if the actor can be possessed in game*
- bool **IsPossessable** ()
- virtual TArray< UGadgetMountingNode ∗ > GetMountingNodes () override
- virtual UGadgetMountingNode ∗ GetMountingNodeBySocketName (FName SocketName) override
- virtual void **MountGadget** (TSubclassOf< AGadget > GadgetClass, USkeletalMeshSocket ∗Socket) override

## Protected Attributes

- USpringArmComponent ∗ **CameraSpringArm**
- UCameraComponent ∗ **ChaseCamera**
- UCameraComponent ∗ **InternalCamera**
- UCameraComponent ∗ **OverheadCamera**
- UCameraComponent ∗ **ActiveCamera**
- bool **bIsPosessable** = true
- TArray< UGadgetMountingNode ∗ > **GadgetMountingNodes**

## 4.7.1 Detailed Description

Base class for all wheeled vehicle actors. Sets up cameras and controls for player usage.

## 4.7.2 Constructor & Destructor Documentation

### 4.7.2.1 ATestbedWheeledVehicle()

```
ATestbedWheeledVehicle::ATestbedWheeledVehicle ( )
```

Default Constructor

summary>Default Deconstructor

#### 4.7.2.2 ∼ATestbedWheeledVehicle()

```
ATestbedWheeledVehicle::∼ATestbedWheeledVehicle ( )
```

summary>Allow actors to initialize themselves on the C++ side

### 4.7.3 Member Function Documentation

#### 4.7.3.1 GetMountingNodeBySocketName()

```
UGadgetMountingNode * ATestbedWheeledVehicle::GetMountingNodeBySocketName (
            FName SocketName ) [override], [virtual]
```

summary> GadgetMountingNodes related to the socket denoted by FName toAddTo

params name = 'GadgetClass'>A static class of the gadget that will be attached to the pawn</params> params name = 'Socket'>The socket on the skeletal mesh of the pawn</params>

Implements IMountablePawn.

#### 4.7.3.2 GetMountingNodes()

```
TArray< UGadgetMountingNode * > ATestbedWheeledVehicle::GetMountingNodes ( ) [override],
[virtual]
```

summary>Get a TArray of pointers to the UGadgetMouningNodes on the vehicle

returns>A TArray of pointers to the UGadgetMountingNodes

summary>Search for and returns a GadgetMountingNode that is associated with a socket name. Can return a nullptr if nothing is found

params name = 'SocketName'>The socket name that is associated with the GadgetMountingNode being searched for</params> returns>The GadgetMountingNode that is associated with the SocketName

Implements IMountablePawn.

#### 4.7.3.3 GetVehicleForwardSpeed()

```
float ATestbedWheeledVehicle::GetVehicleForwardSpeed ( ) const
```

summary>Gets first person view camera for the vehicle

#### 4.7.3.4 PostInitializeComponents()

```
void ATestbedWheeledVehicle::PostInitializeComponents ( )
```

summary>Sets the current throttle applied to the vehicle by the player

param name='Value'>Value of the throttle applied

#### 4.7.3.5 SetBrakeInput()

```
void ATestbedWheeledVehicle::SetBrakeInput (
             float Value )
```

summary>Gets the current speed of the vehicle

returns>Current vehicle speed

#### 4.7.3.6 SetSteeringInput()

```
void ATestbedWheeledVehicle::SetSteeringInput (
             float Value )
```

summary>Sets the current braking applied to the vehicle by the player

param name='Value'>Value of the brakes applied

#### 4.7.3.7 SetThrottleInput()

```
void ATestbedWheeledVehicle::SetThrottleInput (
             float Value )
```

summary>Sets the steering direction and magnitude of it to the vehicle

param name='Value'>Value of the steering applied, positive and negative give steering direction

#### 4.7.3.8 SwitchToChaseCamera()

```
void ATestbedWheeledVehicle::SwitchToChaseCamera ( )
```

summary>Returns true if the actor can be possessed in game

#### 4.7.3.9 SwitchToInternalCamera()

```
void ATestbedWheeledVehicle::SwitchToInternalCamera ( )
```

summary>Gets first person view camera for the vehicle

#### 4.7.3.10 SwitchToOverheadCamera()

```
void ATestbedWheeledVehicle::SwitchToOverheadCamera ( )
```

summary>Gets first person view camera for the vehicle

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

- Source/VehicleTestbed/Public/Vehicles/TestbedWheeledVehicle.h
- Source/VehicleTestbed/Private/Vehicles/TestbedWheeledVehicle.cpp

## 4.8 rapidxml::attribute_iterator< Ch > Class Template Reference

Iterator of child attributes of xml_node.

```
#include <rapidxml_iterators.hpp>
```

### Public Types

- typedef xml_attribute< Ch > **value_type**
- typedef xml_attribute< Ch > & **reference**
- typedef xml_attribute< Ch > ∗ **pointer**
- typedef std::ptrdiff_t **difference_type**
- typedef std::bidirectional_iterator_tag **iterator_category**

### Public Member Functions

- **attribute_iterator** (xml_node< Ch > ∗node)
- reference **operator∗** () const
- pointer **operator->** () const
- attribute_iterator & **operator++** ()
- attribute_iterator **operator++** (int)
- attribute_iterator & **operator--** ()
- attribute_iterator **operator--** (int)
- bool **operator==** (const attribute_iterator< Ch > &rhs)
- bool **operator!=** (const attribute_iterator< Ch > &rhs)

### 4.8.1 Detailed Description

**template<class Ch>**
**class rapidxml::attribute_iterator< Ch >**

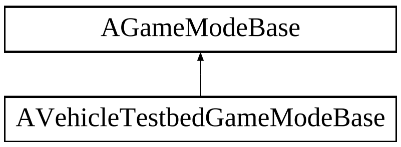
Iterator of child attributes of xml_node.

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

- Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml_iterators.hpp

## 4.9 AVehicleTestbedGameModeBase Class Reference

Inheritance diagram for AVehicleTestbedGameModeBase:



### Public Member Functions

- AVehicleTestbedGameModeBase ()

  *Defualt constructor for Game Mode*
- virtual void PostInitializeComponents () override

  *summary>Called when the game starts, starts dataRecording*
- virtual void BeginPlay () override

  *summary>Called the game ends, stops dataRecording*
- virtual void **EndPlay** (const EEndPlayReason::Type EndPlayReason) override
- UDataRecorder ∗ **GetDataRecorder** () const

### 4.9.1 Constructor & Destructor Documentation

#### 4.9.1.1 AVehicleTestbedGameModeBase()

```
AVehicleTestbedGameModeBase::AVehicleTestbedGameModeBase ( )
```

Defualt constructor for Game Mode

summary>Called just before game starts, initialise all bindings

### 4.9.2 Member Function Documentation

#### 4.9.2.1 BeginPlay()

```
void AVehicleTestbedGameModeBase::BeginPlay ( )  [override], [virtual]
```

summary>Called the game ends, stops dataRecording

#### 4.9.2.2 PostInitializeComponents()

```
void AVehicleTestbedGameModeBase::PostInitializeComponents ( )  [override], [virtual]
```

summary>Called when the game starts, starts dataRecording

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

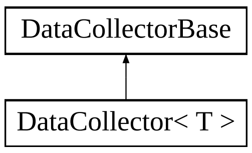
- Source/VehicleTestbed/Public/GameMode/VehicleTestbedGameModeBase.h
- Source/VehicleTestbed/Private/GameMode/VehicleTestbedGameModeBase.cpp

## 4.10 DataCollector< T > Class Template Reference

Templated DataCollector Class. Allows any datacollector type to be held in a single vector

```
#include <DataCollector.h>
```

Inheritance diagram for DataCollector< T >:



### Additional Inherited Members

### 4.10.1 Detailed Description

**template<typename T>**
**class DataCollector< T >**

Templated DataCollector Class. Allows any datacollector type to be held in a single vector

Usage example

```
<![CDATA[
DataCollector<int> myCollector = new DataCollector<int>();
myCollector->FGetDelegate.BindUObject(this, &SomeClass::GetSomeInt);
]]>
```

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

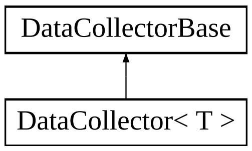
- Source/VehicleTestbed/Public/DataRecording/DataCollector.h

## 4.11 DataCollectorBase Class Reference

Abstract base DataCollector. Used to obtain pointers to concrete children without knowing their type

```
#include <DataCollector.h>
```

Inheritance diagram for DataCollectorBase:



### Public Member Functions

- DataCollectorBase ()

    *Default constructor, initialse with blank name and enabled = true*
- DataCollectorBase (FName name)

    *summary>Full constructor, initialse with called params*
- DataCollectorBase (FName name, bool bEnable)
- virtual std::unique_ptr< DataValueBase > Collect () const =0

    *summary>Setter method for enabled flag*
- virtual void SetEnabled (bool bEnable)
- virtual bool IsEnabled () const
- virtual FName **GetName** () const

### Protected Attributes

- bool **bEnabled**
- FName **Name**

### 4.11.1 Detailed Description

Abstract base DataCollector. Used to obtain pointers to concrete children without knowing their type

### 4.11.2 Constructor & Destructor Documentation

#### 4.11.2.1 DataCollectorBase() [1/3]

```
DataCollectorBase::DataCollectorBase ( )
```

Default constructor, initialse with blank name and enabled = true

summary>Name constructor, initialse with set name and enabled = true

#### 4.11.2.2 DataCollectorBase() [2/3]

```
DataCollectorBase::DataCollectorBase (
             FName name )
```

summary>Full constructor, initialse with called params

#### 4.11.2.3 DataCollectorBase() [3/3]

```
DataCollectorBase::DataCollectorBase (
             FName name,
             bool bEnable )
```

summary>Virtual Print Method, collects the datavalue para>PURE VIRTUAL METHOD: Must be overriden in derived classes

returns>Unique ptr to DataValueBase

### 4.11.3 Member Function Documentation

#### 4.11.3.1 Collect()

```
virtual std::unique_ptr<DataValueBase> DataCollectorBase::Collect ( ) const  [pure virtual]
```

summary>Setter method for enabled flag

#### 4.11.3.2 IsEnabled()

```
bool DataCollectorBase::IsEnabled ( ) const  [virtual]
```

summary>Gets the name of collector

returns>FName of the collector

#### 4.11.3.3 SetEnabled()

```
void DataCollectorBase::SetEnabled (
             bool bEnable )  [virtual]
```

summary>Get enabled condition

returns>Boolean enabled flag

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

- Source/VehicleTestbed/Public/DataRecording/DataCollector.h
- Source/VehicleTestbed/Private/DataRecording/DataCollector.cpp

## 4.12 DataPoint Class Reference

DataPoint class for data recording queue, holds a timestamp and list of dataValues

```
#include <DataPoint.h>
```

### Public Member Functions

- DataPoint ()
  *Default Constructor*
- DataPoint (const DataPoint &otherDataPoint)
- DataPoint & operator= (const DataPoint &otherDataPoint)
  *summary>Destructor*
- virtual ~DataPoint ()
- bool operator== (const DataPoint &other) const
- bool operator!= (const DataPoint &other) const
  *summary>Adds a DataValue to the Data Vector*
- void AddData (std::unique_ptr< DataValueBase > dataValue)
  *summary>Output operator, writes DataPoint to std::ostream*

### Friends

- std::ostream & **operator**<< (std::ostream &os, const DataPoint &dataPoint)

### 4.12.1 Detailed Description

DataPoint class for data recording queue, holds a timestamp and list of dataValues

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 DataPoint() [1/2]

```
DataPoint::DataPoint ( )
```

Default Constructor

summary>Copy Constructor, create deep copy of other datapoint

param name="otherDataPoint">Datapoint to copy

returns>New DataPoint with deep copy of otherDataPoint

#### 4.12.2.2 DataPoint() [2/2]

```
DataPoint::DataPoint (
            const DataPoint & otherDataPoint )
```

summary>Assignment Operator, deep copy all values from other datapoint

param name="otherDataPoint">Datapoint to copy

returns>This DataPoint with deep copy of otherDataPoint

#### 4.12.2.3 ∼DataPoint()

```
DataPoint::∼DataPoint ( ) [virtual]
```

summary>Equality operator, compares datapoints

param name="other">Datapoint to compare

returns>bool Equality

### 4.12.3 Member Function Documentation

#### 4.12.3.1 AddData()

```
void DataPoint::AddData (
            std::unique_ptr< DataValueBase > dataValue )
```

summary>Output operator, writes DataPoint to std::ostream

#### 4.12.3.2 operator"!=()

```
bool DataPoint::operator!= (
            const DataPoint & other ) const
```

summary>Adds a DataValue to the Data Vector

#### 4.12.3.3 operator=()

```
DataPoint & DataPoint::operator= (
            const DataPoint & otherDataPoint )
```

summary>Destructor

#### 4.12.3.4 operator==()

```
bool DataPoint::operator== (
              const DataPoint & other ) const
```

summary>Inequality operator, compares datapoints

param name="other">Datapoint to compare

returns>bool inequality

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

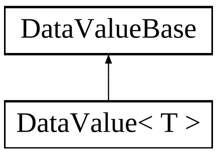
- Source/VehicleTestbed/Public/DataRecording/DataPoint.h
- Source/VehicleTestbed/Private/DataRecording/DataPoint.cpp

## 4.13 DataValue< T > Class Template Reference

Templated DataValue Class. Allows any datatype to be held in a single vector

`#include <DataValue.h>`

Inheritance diagram for DataValue< T >:



### Public Member Functions

- DataValue ()

  *Default constructor, initalises value*
- DataValue (T value)
- DataValue (DataValue< T > const &otherDataValue)

  *summary>Destructor*
- virtual std::unique_ptr< DataValueBase > Clone () const
- void Print (std::ostream &os) const override

### 4.13.1 Detailed Description

**template<typename T>**
**class DataValue< T >**

Templated DataValue Class. Allows any datatype to be held in a single vector

Usage example

```
<![CDATA[
DataValue<int> myIntValue(1);
DataValue<bool> myBoolValue(true);
cout << myIntValue << ' ' << myBoolValue; // returns "1 true"
]]>
```

## 4.13.2 Constructor & Destructor Documentation

#### 4.13.2.1 DataValue() [1/3]

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

Default constructor, initalises value

summary>Default constructor

param name="value">Value to assign

#### 4.13.2.2 DataValue() [2/3]

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

summary>Copy Constructor

param name="otherDataValue">DataValue to copy

#### 4.13.2.3 DataValue() [3/3]

```
template<typename T>
DataValue< T >::DataValue (
             DataValue< T > const & otherDataValue )  [inline]
```

summary>Destructor

## 4.13.3 Member Function Documentation

#### 4.13.3.1 Clone()

```
template<typename T>
virtual std::unique_ptr<DataValueBase> DataValue< T >::Clone ( ) const  [inline], [virtual]
```

summary>Virtual Clone Method, creates a deep copy of the current object

returns>New DataValue of current type

summary>Virtual Print Method, outputs to an std::ostream

param name="os">std::ostream object to write to

Implements DataValueBase.

#### 4.13.3.2 Print()

```
template<typename T>
void DataValue< T >::Print (
            std::ostream & os ) const  [inline], [override], [virtual]
```

summary>Overriden output operator, calls print method on object

param name="os">std::ostream object to write to

param name="dataValue">DataValueBase object to write

returns>std::ostream object

Implements DataValueBase.

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

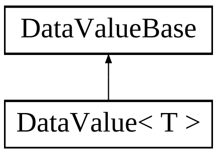
- Source/VehicleTestbed/Public/DataRecording/DataValue.h

## 4.14 DataValueBase Class Reference

Abstract base DataValue. Used to obtain pointers to concrete children without knowing their type

`#include <DataValue.h>`

Inheritance diagram for DataValueBase:



### Public Member Functions

- virtual std::unique_ptr< DataValueBase > Clone () const =0

  *Virtual Clone Method, creates a deep copy of the current object para>PURE VIRTUAL METHOD: Must be overriden in derived classes*
- virtual void Print (std::ostream &os) const =0

### Friends

- std::ostream & **operator<<** (std::ostream &os, const DataValueBase &dataValue)

### 4.14.1 Detailed Description

Abstract base DataValue. Used to obtain pointers to concrete children without knowing their type

### 4.14.2 Member Function Documentation

#### 4.14.2.1 Clone()

```
virtual std::unique_ptr<DataValueBase> DataValueBase::Clone ( ) const  [pure virtual]
```

Virtual Clone Method, creates a deep copy of the current object para>PURE VIRTUAL METHOD: Must be overriden in derived classes

returns>New DataValueBase

summary>Virtual Print Method, outputs to an std::ostream para>PURE VIRTUAL METHOD: Must be overriden in derived classes

param name="os">std::ostream object to write to

Implemented in DataValue< T >.

#### 4.14.2.2 Print()

```
virtual void DataValueBase::Print (
            std::ostream & os ) const  [pure virtual]
```

summary>Overriden output operator, calls print method on object

param name="os">std::ostream object to write to

param name="dataValue">DataValueBase object to write

returns>std::ostream object

Implemented in DataValue< T >.

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

- Source/VehicleTestbed/Public/DataRecording/DataValue.h

## 4.15 FCommConfigStruct Struct Reference

### Public Attributes

- FString **File**
- UCommConfig ∗ **Object**

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

- Source/VehicleTestbed/Public/Configurator/ScenarioConfig.h

## 4.16 rapidxml::file< Ch > Class Template Reference

Represents data loaded from a file.

```
#include <rapidxml_utils.hpp>
```

### Public Member Functions

- file (const char ∗filename)
- file (std::basic_istream< Ch > &stream)
- Ch ∗ data ()
- const Ch ∗ data () const
- std::size_t size () const

### 4.16.1 Detailed Description

**template<class Ch = char>**
**class rapidxml::file< Ch >**

Represents data loaded from a file.

### 4.16.2 Constructor & Destructor Documentation

#### 4.16.2.1 file() [1/2]

```
template<class Ch = char>
rapidxml::file< Ch >::file (
             const char * filename )  [inline]
```

Loads file into the memory. Data will be automatically destroyed by the destructor.

**Parameters**

| | |
|---|---|
| *filename* | Filename to load. |

#### 4.16.2.2 file() [2/2]

```
template<class Ch = char>
rapidxml::file< Ch >::file (
             std::basic_istream< Ch > & stream )  [inline]
```

Loads file into the memory. Data will be automatically destroyed by the destructor

**Parameters**

|  |  |
|---|---|
| *stream* | Stream to load from |

## 4.16.3 Member Function Documentation

### 4.16.3.1 data() [1/2]

```
template<class Ch = char>
Ch* rapidxml::file< Ch >::data ( )  [inline]
```

Gets file data.

**Returns**

Pointer to data of file.

### 4.16.3.2 data() [2/2]

```
template<class Ch = char>
const Ch* rapidxml::file< Ch >::data ( ) const  [inline]
```

Gets file data.

**Returns**

Pointer to data of file.

### 4.16.3.3 size()

```
template<class Ch = char>
std::size_t rapidxml::file< Ch >::size ( ) const  [inline]
```

Gets file data size.

**Returns**

Size of file data, in characters.

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

- Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml_utils.hpp

## 4.17 UEventRecorder::FRecordableEvent Class Reference

A simple class to hold information about an event trigger

```
#include <RecordableEvent.h>
```

### Public Types

- typedef TPair< FString, FString > **Pair**

### Public Member Functions

- FRecordableEvent (const FString EventName, const UObject ∗Caller, const TMap< FString, FString > Details=TMap< FString, FString >())

  *summary> Virtual destructor*
- virtual ∼FRecordableEvent ()
- **FRecordableEvent** (const FRecordableEvent &Other)
- FRecordableEvent & operator= (const FRecordableEvent &Other)=delete
- virtual const TArray< FString > **GetXMLFormattedOutput** () const

### Public Attributes

- const FString **Timestamp**
- const FString **GameTimestamp**
- const FString **Name**
- const FString **Caller**
- const TMap< FString, FString > Details

### 4.17.1 Detailed Description

A simple class to hold information about an event trigger

### 4.17.2 Constructor & Destructor Documentation

#### 4.17.2.1 FRecordableEvent()

```
UEventRecorder::FRecordableEvent::FRecordableEvent (
            const FString EventName,
            const UObject ∗ Caller,
            const TMap< FString, FString > Details = TMap<FString, FString>() )
```

summary>Virtual destructor

#### 4.17.2.2 ∼FRecordableEvent()

```
UEventRecorder::FRecordableEvent::∼FRecordableEvent ( )  [virtual]
```

summary>Copy Constructor

param name="Other">Another FRecordableEvent object

### 4.17.3 Member Function Documentation

#### 4.17.3.1 operator=()

```
FRecordableEvent& UEventRecorder::FRecordableEvent::operator= (
             const FRecordableEvent & Other )  [delete]
```

summary>Gets the info in an XML format as an array of lines

returns>An array of FStrings where each element of the array is a new line

### 4.17.4 Member Data Documentation

#### 4.17.4.1 Details

```
const TMap<FString, FString> UEventRecorder::FRecordableEvent::Details
```

summary>Constructor which requires a name and handler

param name="EventName">Name of the event which was triggered

param name="Caller">Pointer to the object which created this event

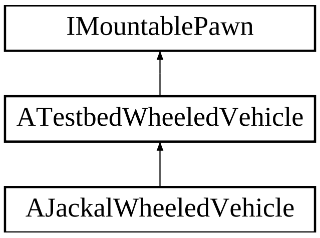
param name="Details">A TMap of key-value pairs to log

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

- Source/VehicleTestbed/Public/EventRecorder/RecordableEvent.h
- Source/VehicleTestbed/Private/EventRecorder/RecordableEvent.cpp

## 4.18 IMountablePawn Class Reference

Inheritance diagram for IMountablePawn:



### Public Member Functions

- virtual TArray< UGadgetMountingNode * > GetMountingNodes ()=0
  *Get a TArray of pointers to the UGadgetMouningNodes on the vehicle*
- virtual UGadgetMountingNode * GetMountingNodeBySocketName (FName SocketName)=0
- virtual void **MountGadget** (TSubclassOf< AGadget > GadgetClass, USkeletalMeshSocket *Socket)=0

### 4.18.1 Member Function Documentation

#### 4.18.1.1 GetMountingNodeBySocketName()

```
virtual UGadgetMountingNode* IMountablePawn::GetMountingNodeBySocketName (
            FName SocketName )  [pure virtual]
```

summary> GadgetMountingNodes related to the socket denoted by FName toAddTo

params name = 'GadgetClass'>A static class of the gadget that will be attached to the pawn</params> params name = 'Socket'>The socket on the skeletal mesh of the pawn</params>

Implemented in ATestbedWheeledVehicle.

#### 4.18.1.2 GetMountingNodes()

```
virtual TArray<UGadgetMountingNode*> IMountablePawn::GetMountingNodes ( )  [pure virtual]
```

Get a TArray of pointers to the UGadgetMouningNodes on the vehicle

returns>A TArray of pointers to the UGadgetMountingNodes

summary>Search for and returns a GadgetMountingNode that is associated with a socket name. Can return a nullptr if nothing is found

params name = 'SocketName'>The socket name that is associated with the GadgetMountingNode being searched for</params> returns>The GadgetMountingNode that is associated with the SocketName

Implemented in ATestbedWheeledVehicle.

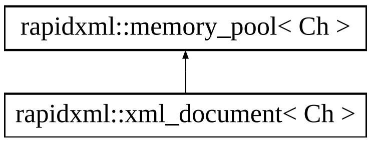
The documentation for this class was generated from the following file:

- Source/VehicleTestbed/Public/MountablePawns/MountablePawn.h

## 4.19 rapidxml::memory_pool< Ch > Class Template Reference

```
#include <rapidxml.hpp>
```

Inheritance diagram for rapidxml::memory_pool< Ch >:



### Public Member Functions

- memory_pool ()

  *Constructs empty pool with default allocator functions.*
- ∼memory_pool ()
- xml_node< Ch > ∗ allocate_node (node_type type, const Ch ∗name=0, const Ch ∗value=0, std::size_↩ t name_size=0, std::size_t value_size=0)
- xml_attribute< Ch > ∗ allocate_attribute (const Ch ∗name=0, const Ch ∗value=0, std::size_t name_size=0, std::size_t value_size=0)
- Ch ∗ allocate_string (const Ch ∗source=0, std::size_t size=0)
- xml_node< Ch > ∗ clone_node (const xml_node< Ch > ∗source, xml_node< Ch > ∗result=0)
- void clear ()
- void set_allocator (alloc_func ∗af, free_func ∗ff)

### 4.19.1 Detailed Description

**template<class Ch = char>**
**class rapidxml::memory_pool< Ch >**

This class is used by the parser to create new nodes and attributes, without overheads of dynamic memory allocation. In most cases, you will not need to use this class directly. However, if you need to create nodes manually or modify names/values of nodes, you are encouraged to use memory_pool of relevant xml_document to allocate the memory. Not only is this faster than allocating them by using `new` operator, but also their lifetime will be tied to the lifetime of document, possibly simplyfing memory management.

Call allocate_node() or allocate_attribute() functions to obtain new nodes or attributes from the pool. You can also call allocate_string() function to allocate strings. Such strings can then be used as names or values of nodes without worrying about their lifetime. Note that there is no `free()` function – all allocations are freed at once when clear() function is called, or when the pool is destroyed.

It is also possible to create a standalone memory_pool, and use it to allocate nodes, whose lifetime will not be tied to any document.

Pool maintains `RAPIDXML_STATIC_POOL_SIZE` bytes of statically allocated memory. Until static memory is exhausted, no dynamic memory allocations are done. When static memory is exhausted, pool allocates additional blocks of memory of size `RAPIDXML_DYNAMIC_POOL_SIZE` each, by using global `new[]` and `delete[]` operators. This behaviour can be changed by setting custom allocation routines. Use set_allocator() function to set them.

Allocations for nodes, attributes and strings are aligned at `RAPIDXML_ALIGNMENT` bytes. This value defaults to the size of pointer on target architecture.

To obtain absolutely top performance from the parser, it is important that all nodes are allocated from a single, contiguous block of memory. Otherwise, cache misses when jumping between two (or more) disjoint blocks of memory can slow down parsing quite considerably. If required, you can tweak `RAPIDXML_STATIC_POOL↩ _SIZE`, `RAPIDXML_DYNAMIC_POOL_SIZE` and `RAPIDXML_ALIGNMENT` to obtain best wasted memory to performance compromise. To do it, define their values before rapidxml.hpp file is included.

**Parameters**

| | |
|---|---|
| *Ch* | Character type of created nodes. |

### 4.19.2 Constructor & Destructor Documentation

#### 4.19.2.1 ∼memory_pool()

```
template<class Ch = char>
rapidxml::memory_pool< Ch >::∼memory_pool ( )  [inline]
```

Destroys pool and frees all the memory. This causes memory occupied by nodes allocated by the pool to be freed. Nodes allocated from the pool are no longer valid.

### 4.19.3 Member Function Documentation

#### 4.19.3.1 allocate_attribute()

```
template<class Ch = char>
xml_attribute<Ch>* rapidxml::memory_pool< Ch >::allocate_attribute (
             const Ch * name = 0,
             const Ch * value = 0,
             std::size_t name_size = 0,
             std::size_t value_size = 0 )  [inline]
```

Allocates a new attribute from the pool, and optionally assigns name and value to it. If the allocation request cannot be accomodated, this function will throw `std::bad_alloc`. If exceptions are disabled by defining RAPIDXML↩ _NO_EXCEPTIONS, this function will call rapidxml::parse_error_handler() function.

**Parameters**

| | |
|---|---|
| *name* | Name to assign to the attribute, or 0 to assign no name. |
| *value* | Value to assign to the attribute, or 0 to assign no value. |
| *name_size* | Size of name to assign, or 0 to automatically calculate size from name string. |
| *value_size* | Size of value to assign, or 0 to automatically calculate size from value string. |

**Returns**

Pointer to allocated attribute. This pointer will never be NULL.

#### 4.19.3.2 allocate_node()

```
template<class Ch = char>
xml_node<Ch>* rapidxml::memory_pool< Ch >::allocate_node (
            node_type type,
            const Ch * name = 0,
            const Ch * value = 0,
            std::size_t name_size = 0,
            std::size_t value_size = 0 )  [inline]
```

Allocates a new node from the pool, and optionally assigns name and value to it. If the allocation request cannot be accomodated, this function will throw `std::bad_alloc`. If exceptions are disabled by defining RAPIDXML_N↩O_EXCEPTIONS, this function will call rapidxml::parse_error_handler() function.

**Parameters**

| | |
|---|---|
| *type* | Type of node to create. |
| *name* | Name to assign to the node, or 0 to assign no name. |
| *value* | Value to assign to the node, or 0 to assign no value. |
| *name_size* | Size of name to assign, or 0 to automatically calculate size from name string. |
| *value_size* | Size of value to assign, or 0 to automatically calculate size from value string. |

**Returns**

Pointer to allocated node. This pointer will never be NULL.

#### 4.19.3.3 allocate_string()

```
template<class Ch = char>
Ch* rapidxml::memory_pool< Ch >::allocate_string (
            const Ch * source = 0,
            std::size_t size = 0 )  [inline]
```

Allocates a char array of given size from the pool, and optionally copies a given string to it. If the allocation request cannot be accomodated, this function will throw `std::bad_alloc`. If exceptions are disabled by defining RA↩PIDXML_NO_EXCEPTIONS, this function will call rapidxml::parse_error_handler() function.

**Parameters**

| | |
|---|---|
| *source* | String to initialize the allocated memory with, or 0 to not initialize it. |
| *size* | Number of characters to allocate, or zero to calculate it automatically from source string length; if size is 0, source string must be specified and null terminated. |

**Returns**

Pointer to allocated char array. This pointer will never be NULL.

#### 4.19.3.4 clear()

```
template<class Ch = char>
void rapidxml::memory_pool< Ch >::clear ( )  [inline]
```

Clears the pool. This causes memory occupied by nodes allocated by the pool to be freed. Any nodes or strings allocated from the pool will no longer be valid.

#### 4.19.3.5 clone_node()

```
template<class Ch = char>
xml_node<Ch>* rapidxml::memory_pool< Ch >::clone_node (
             const xml_node< Ch > * source,
             xml_node< Ch > * result = 0 )  [inline]
```

Clones an xml_node and its hierarchy of child nodes and attributes. Nodes and attributes are allocated from this memory pool. Names and values are not cloned, they are shared between the clone and the source. Result node can be optionally specified as a second parameter, in which case its contents will be replaced with cloned source node. This is useful when you want to clone entire document.

**Parameters**

| | |
|---|---|
| *source* | Node to clone. |
| *result* | Node to put results in, or 0 to automatically allocate result node |

**Returns**

Pointer to cloned node. This pointer will never be NULL.

#### 4.19.3.6 set_allocator()

```
template<class Ch = char>
void rapidxml::memory_pool< Ch >::set_allocator (
             alloc_func * af,
             free_func * ff )  [inline]
```

Sets or resets the user-defined memory allocation functions for the pool. This can only be called when no memory is allocated from the pool yet, otherwise results are undefined. Allocation function must not return invalid pointer on failure. It should either throw, stop the program, or use `longjmp()` function to pass control to other place of program. If it returns invalid pointer, results are undefined.

User defined allocation functions must have the following forms:

```
void *allocate(std::size_t size);
void free(void *pointer);
```

**Parameters**

| | |
|---|---|
| *af* | Allocation function, or 0 to restore default function |
| *ff* | Free function, or 0 to restore default function |

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

- Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml.hpp

## 4.20 rapidxml::node_iterator< Ch > Class Template Reference

Iterator of child nodes of xml_node.

```
#include <rapidxml_iterators.hpp>
```

### Public Types

- typedef xml_node< Ch > **value_type**
- typedef xml_node< Ch > & **reference**
- typedef xml_node< Ch > ∗ **pointer**
- typedef std::ptrdiff_t **difference_type**
- typedef std::bidirectional_iterator_tag **iterator_category**

### Public Member Functions

- **node_iterator** (xml_node< Ch > ∗node)
- reference **operator∗** () const
- pointer **operator->** () const
- node_iterator & **operator++** ()
- node_iterator **operator++** (int)
- node_iterator & **operator--** ()
- node_iterator **operator--** (int)
- bool **operator==** (const node_iterator< Ch > &rhs)
- bool **operator!=** (const node_iterator< Ch > &rhs)

### 4.20.1 Detailed Description

**template<class Ch>**
**class rapidxml::node_iterator< Ch >**

Iterator of child nodes of xml_node.

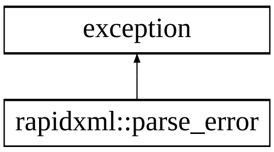
The documentation for this class was generated from the following file:

- Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml_iterators.hpp

## 4.21 rapidxml::parse_error Class Reference

`#include <rapidxml.hpp>`

Inheritance diagram for rapidxml::parse_error:



### Public Member Functions

- parse_error (const char ∗what, void ∗where)

  *Constructs parse error.*
- virtual const char ∗ what () const throw ()
- template<class Ch >
  Ch ∗ where () const

### 4.21.1 Detailed Description

Parse error exception. This exception is thrown by the parser when an error occurs. Use what() function to get human-readable error message. Use where() function to get a pointer to position within source text where error was detected.

If throwing exceptions by the parser is undesirable, it can be disabled by defining RAPIDXML_NO_EXCEPT↩
IONS macro before rapidxml.hpp is included. This will cause the parser to call rapidxml::parse_error_handler() function instead of throwing an exception. This function must be defined by the user.

This class derives from `std::exception` class.

### 4.21.2 Member Function Documentation

#### 4.21.2.1 what()

```
virtual const char* rapidxml::parse_error::what ( ) const throw ( )   [inline], [virtual]
```

Gets human readable description of error.

**Returns**

Pointer to null terminated description of the error.

#### 4.21.2.2 where()

```
template<class Ch >
Ch* rapidxml::parse_error::where ( ) const  [inline]
```

Gets pointer to character data where error happened. Ch should be the same as char type of xml_document that produced the error.

**Returns**

Pointer to location within the parsed string where error occured.

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

- Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml.hpp

## 4.22 SpawnPoint Class Reference

SpawnPoint class for SpawnList, holds location name, location coordinates, Rotation and Tags to identify the spawn-point

```
#include <SpawnPoint.h>
```

### Public Member Functions

- SpawnPoint (FName aName, FVector aLocation, FRotator aRotation, FString aTags)

  *Constructor with name and location provided*
- SpawnPoint ()

  *summary> Destructor*
- ∼SpawnPoint ()
- void SetName (FName newName)
- void SetLocation (FVector newLocation)
- void SetRotation (FRotator newRotation)
- void SetTags (FString newTags)
- void SetSpawnPoint (FVector newLocation, FRotator newRotation)
- FName GetName () const
- FVector GetLocation () const
- FRotator GetRotation () const
- FString **GetTags** () const

### 4.22.1 Detailed Description

SpawnPoint class for SpawnList, holds location name, location coordinates, Rotation and Tags to identify the spawn-point

### 4.22.2 Constructor & Destructor Documentation

#### 4.22.2.1 SpawnPoint() [1/2]

```
SpawnPoint::SpawnPoint (
             FName aName,
             FVector aLocation,
             FRotator aRotation,
             FString aTags )
```

Constructor with name and location provided

summary>Default Constructor

#### 4.22.2.2 SpawnPoint() [2/2]

```
SpawnPoint::SpawnPoint ( )
```

summary>Destructor

#### 4.22.2.3 ∼SpawnPoint()

```
SpawnPoint::∼SpawnPoint ( )
```

summary>Update the name of the SpawnPoint

param name="newName">Name to be set

### 4.22.3 Member Function Documentation

#### 4.22.3.1 GetLocation()

```
FVector SpawnPoint::GetLocation ( ) const
```

summary>Returns the Rotation of the Spawn Point

returns>FRotator Rotation of SpawnPoint

#### 4.22.3.2 GetName()

```
FName SpawnPoint::GetName ( ) const
```

summary>Returns the Location of the Spawn Point

returns>FVector Location of SpawnPoint

#### 4.22.3.3 GetRotation()

```
FRotator SpawnPoint::GetRotation ( ) const
```

summary>Returns the Tags of the Spawn Point

returns>FString Tags of SpawnPoint

#### 4.22.3.4 SetLocation()

```
void SpawnPoint::SetLocation (
             FVector newLocation )
```

summary>Update the Location of the SpawnPoint

param name="newRotation">Location to be set

#### 4.22.3.5 SetName()

```
void SpawnPoint::SetName (
             FName newName )
```

summary>Update the Location of the SpawnPoint

param name="newLocation">Location to be set

#### 4.22.3.6 SetRotation()

```
void SpawnPoint::SetRotation (
             FRotator newRotation )
```

summary>Update the Tags of the SpawnPoint

param name="newTags">Tags to be set

#### 4.22.3.7 SetSpawnPoint()

```
void SpawnPoint::SetSpawnPoint (
             FVector newLocation,
             FRotator newRotation )
```

summary>Returns the name of the Spawn Point

returns>FName Name of SpawnPoint

#### 4.22.3.8 SetTags()

```
void SpawnPoint::SetTags (
             FString newTags )
```

summary>Update the Location of the SpawnPoint

param name="newLocation">Location to be set

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

- Source/VehicleTestbed/Public/SpawnPoints/SpawnPoint.h
- Source/VehicleTestbed/Private/SpawnPoints/SpawnPoint.cpp

## 4.23 SpawnPointList Class Reference

SpawnPointList class for storing and modifying a TArray of SpawnPoints

`#include <SpawnPointList.h>`

### Public Member Functions

- SpawnPointList ()

  *Default Constructor*
- ∼SpawnPointList ()

  *summary>Populate SpawnPoint List*
- void PopulateList ()
- bool AddSpawnPoint (FName Name, FVector Location, FRotator Rotation, FString Tags)
- SpawnPoint GetSpawnPointbyPos (int position) const
- SpawnPoint GetSpawnPointbyName (FName SpawnPointName) const
- TArray< FName > GetSpawnPointRefs () const
- bool **CheckSpawnPointInList** (FName Name) const

### 4.23.1 Detailed Description

SpawnPointList class for storing and modifying a TArray of SpawnPoints

### 4.23.2 Constructor & Destructor Documentation

#### 4.23.2.1 SpawnPointList()

```
SpawnPointList::SpawnPointList ( )
```

Default Constructor

summary>Destructor

#### 4.23.2.2 ∼SpawnPointList()

```
SpawnPointList::∼SpawnPointList ( )
```

summary>Populate SpawnPoint List

### 4.23.3 Member Function Documentation

#### 4.23.3.1 AddSpawnPoint()

```
bool SpawnPointList::AddSpawnPoint (
            FName Name,
            FVector Location,
            FRotator Rotation,
            FString Tags )
```

summary>Returns a Spawn Ppoint based on position in Array

param name="position">position in TArray of SpawnPoint

returns>SpawnPoint SpawnPoint requested or if no Spawnpoint at postion provided returns default SpawnPoint

#### 4.23.3.2 GetSpawnPointbyName()

```
SpawnPoint SpawnPointList::GetSpawnPointbyName (
            FName SpawnPointName ) const
```

summary>Returns an Array of FNames representing all SpawnPoints

returns>Returns a TArray of FNames

#### 4.23.3.3 GetSpawnPointbyPos()

```
SpawnPoint SpawnPointList::GetSpawnPointbyPos (
            int position ) const
```

summary>Returns a SpawnPoint Based on Name of SpawnPoint

param name="SpawnPointName">Name of SpawnPoint to be returned

returns>SpawnPoint SpawnPoint requestedor if no Spawnpoint with name provided returns default SpawnPoint

#### 4.23.3.4 GetSpawnPointRefs()

```
TArray< FName > SpawnPointList::GetSpawnPointRefs ( ) const
```

summary>Returns True if Spawnpoint found in list

returns>Returns True if Spawnpoint is found in list. False if not found

#### 4.23.3.5 PopulateList()

```
void SpawnPointList::PopulateList ( )
```

summary>Add a new SpawnPoint to the SpawnPointList

param name="Name">Name of SpawnPoint

param name="Location">Location of SpawnPoint

param name="Rotation">Rotation of SpawnPoint

param name="Tags">Tags to find SpawnPoint

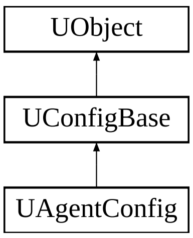
returns>bool True if Spawn Point is created successfully

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

- Source/VehicleTestbed/Public/SpawnPoints/SpawnPointList.h
- Source/VehicleTestbed/Private/SpawnPoints/SpawnPointList.cpp

## 4.24 UAgentConfig Class Reference

Inheritance diagram for UAgentConfig:



### Public Member Functions

- **UAgentConfig** (rapidxml::xml_node<> ∗Node)
- virtual void **AppendDocument** (rapidxml::xml_document<> &OutDocument) const override

### Protected Attributes

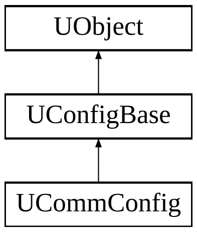
- UClass ∗ **AgentClass**
- FString **AgentName**
- TSubclassOf< ATestbedPlayerController > **Controller**
- TMap< FString, FString > **Properties**

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

- Source/VehicleTestbed/Public/Configurator/AgentConfig.h
- Source/VehicleTestbed/Private/Configurator/AgentConfig.cpp

## 4.25 UCommConfig Class Reference

Inheritance diagram for UCommConfig:



### Public Member Functions

- virtual void **AppendDocument** (rapidxml::xml_document<> &OutDocument) const override
- virtual bool **InitializeFromXML** (rapidxml::xml_document<> &doc) override
- virtual bool **Instantiate** (UObject ∗ContextObject) override

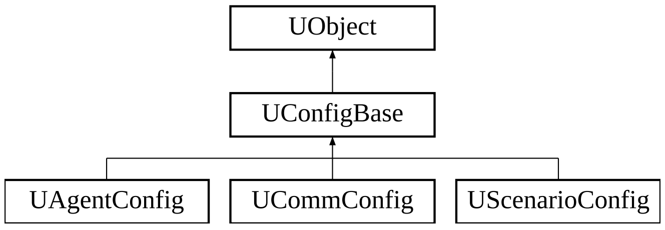
### Additional Inherited Members

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

- Source/VehicleTestbed/Public/Configurator/CommConfig.h
- Source/VehicleTestbed/Private/Configurator/CommConfig.cpp

## 4.26 UConfigBase Class Reference

Inheritance diagram for UConfigBase:



### Public Member Functions

- virtual void **AppendDocument** (rapidxml::xml_document<> &OutDocument) const
- virtual bool **InitializeFromXML** (rapidxml::xml_document<> &doc)
- virtual bool **Instantiate** (UObject ∗ContextObject)
- FString **GetFileLocation** () const
- void **SetFileLocation** (const FString &NewLocation)

### Protected Attributes

- FString **FileLocation**

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

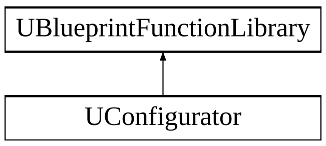
- Source/VehicleTestbed/Public/Configurator/ConfigBase.h
- Source/VehicleTestbed/Private/Configurator/ConfigBase.cpp

## 4.27 UConfigurator Class Reference

Static function library to read and write configurations

```
#include <Configurator.h>
```

Inheritance diagram for UConfigurator:



### Static Public Member Functions

- static void StartScenario (UObject ∗ContextObject)

  *Runs the saved scenario, setting things up and spawning things into the current map*
- static UConfigBase ∗ LoadConfig (FString Filename)
- static void SaveConfig (UConfigBase ∗Config)
- static void **ReloadConfig** (UConfigBase ∗Config)
- static FString **GetScenario** ()
- static void **SetScenario** (FString ScenarioFile)

### 4.27.1 Detailed Description

Static function library to read and write configurations

### 4.27.2 Member Function Documentation

#### 4.27.2.1 LoadConfig()

```
UConfigBase ∗ UConfigurator::LoadConfig (
            FString Filename ) [static]
```

summary>Saves the config object at the specified file location

param name="Config">The config object to save

#### 4.27.2.2 SaveConfig()

```
void UConfigurator::SaveConfig (
            UConfigBase * Config )  [static]
```

summary>Reloads a config object from the file it is based on

param name="Config">The config object to reload

#### 4.27.2.3 StartScenario()

```
void UConfigurator::StartScenario (
            UObject * ContextObject )  [static]
```

Runs the saved scenario, setting things up and spawning things into the current map

param name="ContextObject">The context object used to load the map, the widget calling this function will suffice

summary>Loads a config object from the specified file

param name="Filename">The full path of the file to load from

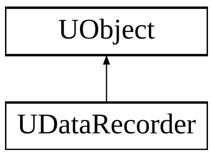
returns>The config object which was loaded, or nullptr if it wasn't

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

- Source/VehicleTestbed/Public/Configurator/Configurator.h
- Source/VehicleTestbed/Private/Configurator/Configurator.cpp

## 4.28 UDataRecorder Class Reference

Inheritance diagram for UDataRecorder:



### Public Member Functions

- UDataRecorder ()

  *Default constructor, sets clock rate to 100ms and filename to 'data.csv'*
- UDataRecorder (int clockRateMS)
- UDataRecorder (std::string filename)
- UDataRecorder (int clockRateMS, std::string filename)

  *summary>Destructor, clean up collectors, ensure queue is fulled destroyed*
- virtual ∼UDataRecorder ()

  *summary>Override base begin destroy to ensure all threads are exited cleanly*
- virtual void BeginDestroy () override
- std::thread StartReader ()
- std::thread StartWriter ()
- void SetClockRate (int clockRateMS)
- int GetClockRate ()
- void SetFilePath (const std::string &NewPath)
- std::string GetFilePath () const
- void AddCollector (DataCollectorBase ∗collector)
- void Start ()

  *summary>Joins running threads and ends execution*
- void Stop ()

  *summary>Pauses the data collection thread until Resume() is called*
- void Pause ()

  *summary>Resumes the data collection thread*
- void **Resume** ()

## 4.28.1 Constructor & Destructor Documentation

#### 4.28.1.1 UDataRecorder() [1/4]

```
UDataRecorder::UDataRecorder ( )
```

Default constructor, sets clock rate to 100ms and filename to 'data.csv'

summary>Constructor with clock rate, sets filename to 'data.csv'

param name="clockRateMS">Clock rate to use

#### 4.28.1.2 UDataRecorder() [2/4]

```
UDataRecorder::UDataRecorder (
              int clockRateMS )
```

summary>Constructor with filename, sets clock rate to 100ms

param name="filename">Output filename

#### 4.28.1.3 UDataRecorder() [3/4]

```
UDataRecorder::UDataRecorder (
             std::string filename )
```

summary>Constructor with clock rate

param name="clockRateMS">Clock rate to use

param name="filename">Output filename

#### 4.28.1.4 UDataRecorder() [4/4]

```
UDataRecorder::UDataRecorder (
             int clockRateMS,
             std::string filename )
```

summary>Destructor, clean up collectors, ensure queue is fulled destroyed

#### 4.28.1.5 ∼UDataRecorder()

```
UDataRecorder::∼UDataRecorder ( ) [virtual]
```

summary>Override base begin destroy to ensure all threads are exited cleanly

### 4.28.2 Member Function Documentation

#### 4.28.2.1 AddCollector()

```
void UDataRecorder::AddCollector (
             DataCollectorBase * collector )
```

summary>Calls StartReader() and StartWriter() and stores the thread references

#### 4.28.2.2 BeginDestroy()

```
void UDataRecorder::BeginDestroy ( ) [override], [virtual]
```

summary>Spawns a thread calling ReadFromCollectors()

returns>std::thread reference for spawned thread

#### 4.28.2.3 GetClockRate()

```
int UDataRecorder::GetClockRate ( )
```

summary>Sets the path for the output file

param name="NewPath">New folder path

#### 4.28.2.4 GetFilePath()

```
std::string UDataRecorder::GetFilePath ( ) const
```

summary>Add a new collector

param name="collector">Const Pointer to the data to collect

param name="type">of the pointer

#### 4.28.2.5 Pause()

```
void UDataRecorder::Pause ( )
```

summary>Resumes the data collection thread

#### 4.28.2.6 SetClockRate()

```
void UDataRecorder::SetClockRate (
             int clockRateMS )
```

summary>Gets the current clock rate

returns>Current clock rate in ms

#### 4.28.2.7 SetFilePath()

```
void UDataRecorder::SetFilePath (
             const std::string & NewPath )
```

summary>Gets the folder path for the output file

returns>The folder path

#### 4.28.2.8 Start()

```
void UDataRecorder::Start ( )
```

summary>Joins running threads and ends execution

#### 4.28.2.9 StartReader()

```
std::thread UDataRecorder::StartReader ( )
```

summary>Spawns a thread calling WriteToFile()

returns>std::thread reference for spawned thread

#### 4.28.2.10 StartWriter()

```
std::thread UDataRecorder::StartWriter ( )
```

summary>Sets Clock Rate

param name="clockRateMS">Clock rate in ms

#### 4.28.2.11 Stop()

```
void UDataRecorder::Stop ( )
```

summary>Pauses the data collection thread until Resume() is called

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

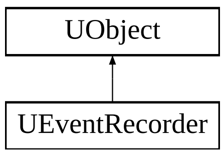
- Source/VehicleTestbed/Public/DataRecording/DataRecorder.h
- Source/VehicleTestbed/Private/DataRecording/DataRecorder.cpp

## 4.29 UEventRecorder Class Reference

Collects details of events and writes them to a file

```
#include <EventRecorder.h>
```

Inheritance diagram for UEventRecorder:



### Classes

- class FRecordableEvent
  *A simple class to hold information about an event trigger*

## Public Types

- typedef TSharedRef< const FRecordableEvent > **EventRef**

## Static Public Member Functions

- static void RecordEvent (EventRef REvent)

  *Collects data about an event and adds it to the queue to be written*
- static void RecordEvent (const FString EventName, const UObject ∗Caller)
- static void RecordEventWithDetails (const FString EventName, const UObject ∗Caller, const TMap< FString, FString > Details)
- static const bool Start ()

  *summary>Sets the stop variable to true, causing the WriterThread to complete gracefully when it next checks the boolean's value*
- static void Stop ()
- static FString GetFolderOutput ()
- static void **SetFolderOutput** (FString NewFolderLocation)

### 4.29.1 Detailed Description

Collects details of events and writes them to a file

### 4.29.2 Member Function Documentation

#### 4.29.2.1 GetFolderOutput()

```
FString UEventRecorder::GetFolderOutput ( )  [static]
```

summary>Sets the output file location

param="NewFolderLocation">The string representation of the new folder location

#### 4.29.2.2 RecordEvent() [1/2]

```
void UEventRecorder::RecordEvent (
             EventRef REvent )  [static]
```

Collects data about an event and adds it to the queue to be written

param name ="REvent">An FRecordableEvent object to be added to the queue

summary>Collects data about an event and adds it to the queue to be written

param name="EventName">Name of the event which was triggered

param name="Caller">Pointer to the UObject which called this function

#### 4.29.2.3 RecordEvent() [2/2]

```
void UEventRecorder::RecordEvent (
            const FString EventName,
            const UObject * Caller )  [static]
```

summary>Collects data about an event and adds it to the queue to be written

param name="EventName">Name of the event which was triggered

param name="Caller">Pointer to the UObject which called this funtion

param name="Details">A TMap of key-value pairs to log

#### 4.29.2.4 RecordEventWithDetails()

```
void UEventRecorder::RecordEventWithDetails (
            const FString EventName,
            const UObject * Caller,
            const TMap< FString, FString > Details )  [static]
```

summary>Creates a thread running Write()

returns>True if the thread was started, false if it was already running.

#### 4.29.2.5 Start()

```
const bool UEventRecorder::Start ( )  [static]
```

summary>Sets the stop variable to true, causing the WriterThread to complete gracefully when it next checks the boolean's value

#### 4.29.2.6 Stop()

```
void UEventRecorder::Stop ( )  [static]
```

summary>Returns the output file location

returns>The string representation of the output folder location

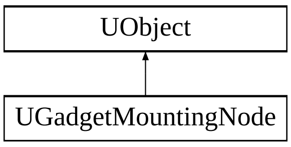
The documentation for this class was generated from the following files:

- Source/VehicleTestbed/Public/EventRecorder/EventRecorder.h
- Source/VehicleTestbed/Private/EventRecorder/EventRecorder.cpp

## 4.30 UGadgetMountingNode Class Reference

```
#include <GadgetMountingNode.h>
```

Inheritance diagram for UGadgetMountingNode:



### Public Member Functions

- UGadgetMountingNode ()

  *Default Constructor*
- ∼UGadgetMountingNode ()
- AGadget ∗ GetMountedGadget ()
- void SetMountedGadget (AGadget ∗Gadget)

  *summary>Calls the gadget's Activate method if there is a gadget attached*
- void ActivateGadget ()
- USkeletalMeshSocket ∗ GetMeshSocket ()
- void **SetMeshSocket** (USkeletalMeshSocket ∗Socket)

### 4.30.1 Detailed Description

This class represents the socket on the skeletal mesh of the vehicle that will have a gadget attached

### 4.30.2 Constructor & Destructor Documentation

#### 4.30.2.1 UGadgetMountingNode()

```
UGadgetMountingNode::UGadgetMountingNode ( )
```

Default Constructor

summary>Default Deconstructor

#### 4.30.2.2 ∼UGadgetMountingNode()

```
UGadgetMountingNode::∼UGadgetMountingNode ( )
```

summary>Returns the pointer to the mounted gadget, can be nullptr

returns>Pointer to mounted gadget

### 4.30.3 Member Function Documentation

#### 4.30.3.1 ActivateGadget()

```
void UGadgetMountingNode::ActivateGadget ( )
```

summary>Returns the name of the socket on the MountablePawn related to this node

returns>FName of related socket

#### 4.30.3.2 GetMeshSocket()

```
USkeletalMeshSocket * UGadgetMountingNode::GetMeshSocket ( )
```

summary>Sets the SkeletalMeshSocket for the GadgetMountingNode

params name='Socket'>Pointer to the SkeletalMeshSocket that will be set</params>

#### 4.30.3.3 GetMountedGadget()

```
AGadget * UGadgetMountingNode::GetMountedGadget ( )
```

summary>Sets the internal refrence for the mounted gadget

params name='toSetTo'>Pointer to replace current refrence to mounted gadget</params>

#### 4.30.3.4 SetMountedGadget()

```
void UGadgetMountingNode::SetMountedGadget (
             AGadget * Gadget )
```

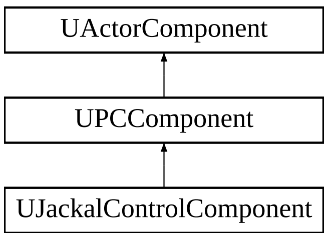
summary>Calls the gadget's Activate method if there is a gadget attached

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

- Source/VehicleTestbed/Public/MountablePawns/GadgetMountingNode.h
- Source/VehicleTestbed/Private/MountablePawns/GadgetMountingNode.cpp

## 4.31 UJackalControlComponent Class Reference

Inheritance diagram for UJackalControlComponent:



## Public Member Functions

- void SetupPlayerInputComponent (UInputComponent ∗InputComponent) override

  *Binds the actions of driving and changing cameras of the Jackal*
- void BeginPlay () override
- void SetThrottleInput (float Value)
- void SetSteeringInput (float Value)
- void SetBrakeInput (float Value)

  *summary>Gets overhead view for the pawn*
- void SwitchToOverheadCamera ()

  *summary>Gets first person view camera for the pawn*
- void SwitchToInternalCamera ()

  *summary>Gets third person view camera for the pawn*
- void **SwitchToChaseCamera** ()

## Additional Inherited Members

## 4.31.1 Member Function Documentation

### 4.31.1.1 BeginPlay()

```
void UJackalControlComponent::BeginPlay ( )  [override], [virtual]
```

summary>Sets the current forward movement applied to the pawn by the player

param name='Value'>Value of the movement applied

Reimplemented from UPCComponent.

### 4.31.1.2 SetBrakeInput()

```
void UJackalControlComponent::SetBrakeInput (
             float Value )
```

summary>Gets overhead view for the pawn

### 4.31.1.3 SetSteeringInput()

```
void UJackalControlComponent::SetSteeringInput (
             float Value )
```

summary>Sets the current braking applied to the vehicle by the player

param name='Value'>Value of the brakes applied

#### 4.31.1.4 SetThrottleInput()

```
void UJackalControlComponent::SetThrottleInput (
            float Value )
```

summary>Sets the steering/strafing direction and magnitude of it to the pawn

param name='Value'>Value of the steering/strafing applied, positive and negative give steering direction (+ is right)

#### 4.31.1.5 SetupPlayerInputComponent()

```
void UJackalControlComponent::SetupPlayerInputComponent (
            UInputComponent * InputComponent ) [override], [virtual]
```

Binds the actions of driving and changing cameras of the Jackal

param name="inputComponent">The input component of the player controller this component is part of

summary>Called automatically by Unreal

Reimplemented from UPCComponent.

#### 4.31.1.6 SwitchToInternalCamera()

```
void UJackalControlComponent::SwitchToInternalCamera ( )
```

summary>Gets third person view camera for the pawn

#### 4.31.1.7 SwitchToOverheadCamera()

```
void UJackalControlComponent::SwitchToOverheadCamera ( )
```

summary>Gets first person view camera for the pawn

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

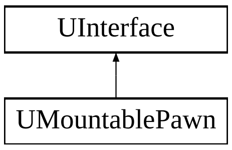
- Source/VehicleTestbed/Public/PlayerController/PlayerControllerComponent/JackalControlComponent.h
- Source/VehicleTestbed/Private/PlayerController/PlayerControllerComponent/JackalControlComponent.cpp

## 4.32 UMountablePawn Class Reference

This is an interface that allows any class that uses it to have gadgets attached to their sockets on their skeletal mesh

```
#include <MountablePawn.h>
```

Inheritance diagram for UMountablePawn:



### 4.32.1 Detailed Description

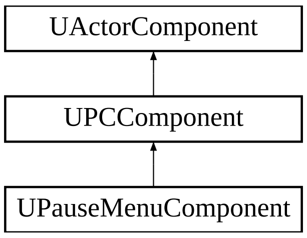
This is an interface that allows any class that uses it to have gadgets attached to their sockets on their skeletal mesh

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

- Source/VehicleTestbed/Public/MountablePawns/MountablePawn.h

## 4.33 UPauseMenuComponent Class Reference

Inheritance diagram for UPauseMenuComponent:



### Public Member Functions

- UPauseMenuComponent ()

  *Default constructor. Gets a reference to the class of the pause widget so it can be instantiated later*
- void SetupPlayerInputComponent (UInputComponent ∗InputComponent) override

  *summary> Called automatically by Unreal. Instantiates an object of the pause menu.*
- void **BeginPlay** () override

### Additional Inherited Members

### 4.33.1 Constructor & Destructor Documentation

#### 4.33.1.1 UPauseMenuComponent()

```
UPauseMenuComponent::UPauseMenuComponent ( )
```

Default constructor. Gets a reference to the class of the pause widget so it can be instantiated later

summary>Needs to be overriden in sub classes. Put input bindings in here and call this function in the SetupPlayerInputComponent() function of the TestbedPlayerController

param name="inputComponent">The input component of the TestbedPlayerController

### 4.33.2 Member Function Documentation

#### 4.33.2.1 SetupPlayerInputComponent()

```
void UPauseMenuComponent::SetupPlayerInputComponent (
            UInputComponent * InputComponent )  [override], [virtual]
```

summary>Called automatically by Unreal. Instantiates an object of the pause menu.

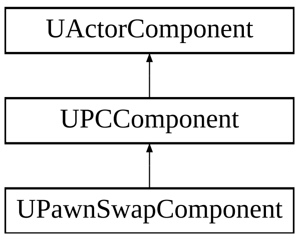
Reimplemented from UPCComponent.

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

- Source/VehicleTestbed/Public/PlayerController/PlayerControllerComponent/PauseMenuComponent.h
- Source/VehicleTestbed/Private/PlayerController/PlayerControllerComponent/PauseMenuComponent.cpp

## 4.34 UPawnSwapComponent Class Reference

Inheritance diagram for UPawnSwapComponent:



### Public Member Functions

- void SetupPlayerInputComponent (UInputComponent ∗InputComponent) override

  *Binds the actions of cycling forward and backward between pawns*
- void BeginPlay () override

  *summary>Switches to the next available vehicle in the array*
- void CycleCharacterForward ()

  *summary>Switches to the previous available vehicle in the array*
- void **CycleCharacterBackward** ()

### Additional Inherited Members

### 4.34.1 Member Function Documentation

#### 4.34.1.1 BeginPlay()

```
void UPawnSwapComponent::BeginPlay ( )  [override], [virtual]
```

summary>Switches to the next available vehicle in the array

Reimplemented from UPCComponent.

#### 4.34.1.2 CycleCharacterForward()

```
void UPawnSwapComponent::CycleCharacterForward ( )
```

summary>Switches to the previous available vehicle in the array

#### 4.34.1.3 SetupPlayerInputComponent()

```
void UPawnSwapComponent::SetupPlayerInputComponent (
             UInputComponent * InputComponent )  [override], [virtual]
```

Binds the actions of cycling forward and backward between pawns

param name="inputComponent">The input component of the player controller this component is part of

summary>Called automatically by Unreal. Sets up a list of all the pawns in the game world.

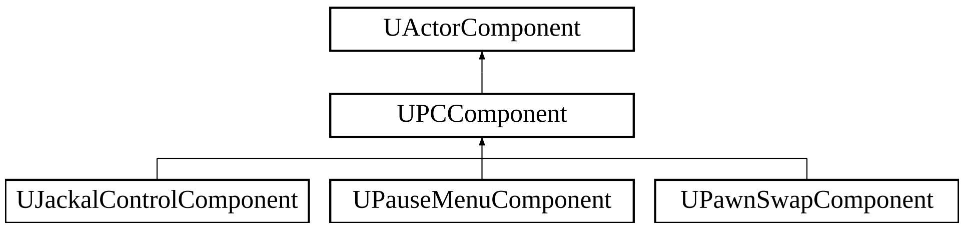
Reimplemented from UPCComponent.

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

- Source/VehicleTestbed/Public/PlayerController/PlayerControllerComponent/PawnSwapComponent.h
- Source/VehicleTestbed/Private/PlayerController/PlayerControllerComponent/PawnSwapComponent.cpp

## 4.35 UPCComponent Class Reference

Inheritance diagram for UPCComponent:



### Public Member Functions

- UPCComponent ()

    *Default constructor. Gets a reference to the TestbedPlayerController it is a component of.*
- virtual void SetupPlayerInputComponent (UInputComponent ∗InputComponent)

    *summary>Called automatically by Unreal, override this if you need to.*
- virtual void **BeginPlay** () override

### Protected Attributes

- APlayerController ∗ **Controller**

## 4.35.1 Constructor & Destructor Documentation

#### 4.35.1.1 UPCComponent()

```
UPCComponent::UPCComponent ( )  [inline]
```

Default constructor. Gets a reference to the TestbedPlayerController it is a component of.

summary>Needs to be overriden in sub classes. Put input bindings in here and call this function in the SetupPlayerInputComponent() function of the TestbedPlayerController

param name="inputComponent">The input component of the TestbedPlayerController

## 4.35.2 Member Function Documentation

#### 4.35.2.1 SetupPlayerInputComponent()

```
virtual void UPCComponent::SetupPlayerInputComponent (
            UInputComponent * InputComponent )  [inline], [virtual]
```

summary>Called automatically by Unreal, override this if you need to.

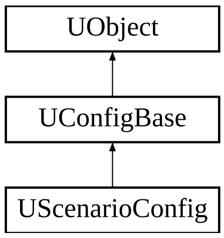
Reimplemented in UPauseMenuComponent, UPawnSwapComponent, and UJackalControlComponent.

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

- Source/VehicleTestbed/Public/PlayerController/PlayerControllerComponent/PCComponent.h

## 4.36 UScenarioConfig Class Reference

Inheritance diagram for UScenarioConfig:



### Public Member Functions

- UScenarioConfig ()

  *Default constructor*
- void AppendDocument (rapidxml::xml_document< > &OutDocument) const override
- bool InitializeFromXML (rapidxml::xml_document< > &doc) override
- bool Instantiate (UObject ∗ContextObject) override
- FName GetMapName () const
- void SetMapName (FName NewMapName)
- TArray< FString > GetAgentFileLocations () const
- TArray< UAgentConfig ∗ > GetAgents ()
- UAgentConfig ∗ GetAgent (const FString AgentFile)
- void AddAgent (const FString AgentFile, UAgentConfig ∗AgentConfig=nullptr)
- void RemoveAgentByFile (const FString AgentFile)
- void RemoveAgentByObject (const UAgentConfig ∗AgentConfig)
- TArray< FName > GetSpawnPoints (const FString AgentFile=TEXT("")) const
- UAgentConfig ∗ GetAgentBySpawn (const FName SpawnName)
- bool AddSpawn (const FName SpawnName, const FString AgentFile)
- bool ChangeAgentAtSpawn (const FName SpawnName, const FString NewAgentFile)
- bool RemoveSpawn (const FName SpawnName)
- int32 RemoveAllSpawnsOfAgent (const FString AgentFile)
- FString GetDataRecordingOutputFolder () const
- void SetDataRecordingOutputFolder (const FString &NewOutputLocation)
- FString GetEventRecordingOutputFolder () const
- void SetEventRecordingOuptutFolder (const FString &NewOutputLocation)
- FString GetCommConfig () const
- UCommConfig ∗ GetCommConfigObject ()
- void **SetCommConfig** (const FString &NewCommConfig)

### Additional Inherited Members

### 4.36.1 Constructor & Destructor Documentation

#### 4.36.1.1 UScenarioConfig()

```
UScenarioConfig::UScenarioConfig ( )
```

Default constructor

summary>Generates a rapidxml node structure from the information in this object

param name="OutDocument">The document to append the xml node to

### 4.36.2 Member Function Documentation

#### 4.36.2.1 AddAgent()

```
void UScenarioConfig::AddAgent (
             const FString AgentFile,
             UAgentConfig * AgentConfig = nullptr )
```

summary>Removes the selected agent config from this scenario

param="AgentFile">The string representation of the file location of the Agent Config to remove

#### 4.36.2.2 AddSpawn()

```
bool UScenarioConfig::AddSpawn (
             const FName SpawnName,
             const FString AgentFile )
```

summary>Changes the agent spawned at a spawn point

param="SpawnName">The name of the spawn point to modify

param="NewAgentConfig">The config of the new agent to spawn at that point

returns>Whether the spawn point config was changed or not

#### 4.36.2.3 AppendDocument()

```
void UScenarioConfig::AppendDocument (
             rapidxml::xml_document<> & OutDocument ) const  [override], [virtual]
```

summary>Initializes this config object from a rapidxml document

param="doc">The rapidxml document to parse

Reimplemented from UConfigBase.

#### 4.36.2.4 ChangeAgentAtSpawn()

```
bool UScenarioConfig::ChangeAgentAtSpawn (
            const FName SpawnName,
            const FString NewAgentFile )
```

summary>Removes a spawn point from this scenario

param="SpawnName">The name of the spawn point to remove

returns>Whether the spawn point was removed or not

#### 4.36.2.5 GetAgent()

```
UAgentConfig * UScenarioConfig::GetAgent (
            const FString AgentFile )
```

summary>Adds an agent to the scenario param="AgentFile">The string representation of the file location used for this Agent Config

param="AgentConfig">An optional Agent Config object if it is already loaded in memory

#### 4.36.2.6 GetAgentBySpawn()

```
UAgentConfig * UScenarioConfig::GetAgentBySpawn (
            const FName SpawnName )
```

summary>Adds a new spawn point configuration to this scenario

param="SpawnName">The name of the spawn point

param="AgentConfig">The agent config object to be used

returns>Whether a spawn point was successfully added or not

#### 4.36.2.7 GetAgentFileLocations()

```
TArray< FString > UScenarioConfig::GetAgentFileLocations ( ) const
```

summary>Returns an array of Agent Config objects used by this scenario, creating them from file if they don't exist in memory

returns>An array of Agent Config objects used by this scenario

#### 4.36.2.8 GetAgents()

```
TArray< UAgentConfig * > UScenarioConfig::GetAgents ( )
```

summary>Returns a pointer to the AgentConfig object loaded from the passed in config file location

param="AgentFile">The file location of the agent config object to find

returns>A pointer to the agent config object, or nullptr if it could not be found

#### 4.36.2.9 GetCommConfig()

```
FString UScenarioConfig::GetCommConfig ( ) const
```

summary>Gets the communications config object used in this scenario

returns>The communications config object

#### 4.36.2.10 GetCommConfigObject()

```
UCommConfig * UScenarioConfig::GetCommConfigObject ( )
```

summary>Sets the file location of the communications config to use

param name="NewCommConfig">The file location

#### 4.36.2.11 GetDataRecordingOutputFolder()

```
FString UScenarioConfig::GetDataRecordingOutputFolder ( ) const
```

summary>Sets the output file location of the data recording system

param="NewOutputLocation">The string representation of a new output location

#### 4.36.2.12 GetEventRecordingOutputFolder()

```
FString UScenarioConfig::GetEventRecordingOutputFolder ( ) const
```

summary>Sets the output file location of the event recording system

param="NewOutputLocation">The string representation of a new output location

#### 4.36.2.13 GetMapName()

```
FName UScenarioConfig::GetMapName ( ) const
```

summary>Sets the name of the map used for this scenario

param="NewMapName">The name of the new map used for this scenario

#### 4.36.2.14 GetSpawnPoints()

```
TArray< FName > UScenarioConfig::GetSpawnPoints (
             const FString AgentFile = TEXT("") ) const
```

summary>Returns the agent config object used at this spawn point

param="SpawnName">The name of the spawn point to search with

returns>The agent config object used at this spawn point or nullptr if it could not be found or loaded

#### 4.36.2.15 InitializeFromXML()

```
bool UScenarioConfig::InitializeFromXML (
             rapidxml::xml_document<> & doc )  [override], [virtual]
```

summary>Instantiates the scenario this config object depicts in the current world (assumes the map has already been loaded)

param name="ContextObject">The context object used to load the map

Reimplemented from UConfigBase.

#### 4.36.2.16 Instantiate()

```
bool UScenarioConfig::Instantiate (
             UObject * ContextObject )  [override], [virtual]
```

summary>Returns the name of the map used in this scenario</summmary> returns>The name of the map uesd in this scenario

Reimplemented from UConfigBase.

#### 4.36.2.17 RemoveAgentByFile()

```
void UScenarioConfig::RemoveAgentByFile (
             const FString AgentFile )
```

summary>Removes the selected agent config from this scenario

param="AgentConfig">The config object of the agent to remove. Uses the FileLocation of the config object.

#### 4.36.2.18 RemoveAgentByObject()

```
void UScenarioConfig::RemoveAgentByObject (
             const UAgentConfig * AgentConfig )
```

summary>Returns an array of names of the spawn points used with an agent config

param="AgentConfig">The agent config to search with

returns>An array of names of the spawn points used with the agent config, or all if the parameter string is empty

#### 4.36.2.19 RemoveAllSpawnsOfAgent()

```
int32 UScenarioConfig::RemoveAllSpawnsOfAgent (
             const FString AgentFile )
```

summary>Returns the output file location of the data recording system

returns>The string representation of the output location

#### 4.36.2.20 RemoveSpawn()

```
bool UScenarioConfig::RemoveSpawn (
             const FName SpawnName )
```

summary>Removes all spawn points used by an agent config

param="AgentFile">The agent config file to search with

returns>The number of spawn points removed

#### 4.36.2.21 SetDataRecordingOutputFolder()

```
void UScenarioConfig::SetDataRecordingOutputFolder (
             const FString & NewOutputLocation )
```

summary>Returns the output file location of the event recording system

returns>The string representation of the output location

#### 4.36.2.22 SetEventRecordingOuptutFolder()

```
void UScenarioConfig::SetEventRecordingOuptutFolder (
             const FString & NewOutputLocation )
```

summary>Gets the file location of the communications config to use

returns>The file location

#### 4.36.2.23 SetMapName()

```
void UScenarioConfig::SetMapName (
             FName NewMapName )
```

summary>Returns an array of the file locations of the agent configs used in this scenario

returns>An array of file locations of the agents configs used in this scenario

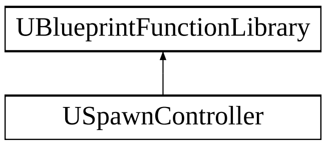
The documentation for this class was generated from the following files:

- Source/VehicleTestbed/Public/Configurator/ScenarioConfig.h
- Source/VehicleTestbed/Private/Configurator/ScenarioConfig.cpp

## 4.37 USpawnController Class Reference

```
#include <SpawnController.h>
```

Inheritance diagram for USpawnController:



### Public Member Functions

- TArray< FName > GetSpawnPointRefs () const
  *Returns an Array of FNames representing all SpawnPoints*

### 4.37.1 Detailed Description

CONTROLLER CLASS FOR UPDATING SPAWN LOCATION - MAY NOT BE NEEDED

### 4.37.2 Member Function Documentation

#### 4.37.2.1 GetSpawnPointRefs()

```
TArray< FName > USpawnController::GetSpawnPointRefs ( ) const
```

Returns an Array of FNames representing all SpawnPoints

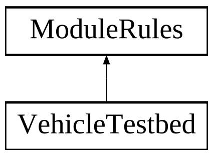
returns>Returns a TArray of FNames

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

- Source/VehicleTestbed/Public/SpawnPoints/SpawnController.h
- Source/VehicleTestbed/Private/SpawnPoints/SpawnController.cpp

## 4.38 VehicleTestbed Class Reference

Inheritance diagram for VehicleTestbed:



### Public Member Functions

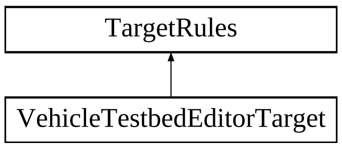
- **VehicleTestbed** (ReadOnlyTargetRules Target)

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

- Source/VehicleTestbed/VehicleTestbed.Build.cs

## 4.39 VehicleTestbedEditorTarget Class Reference

Inheritance diagram for VehicleTestbedEditorTarget:



### Public Member Functions

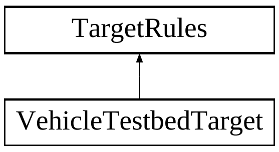
- **VehicleTestbedEditorTarget** (TargetInfo Target)

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

- Source/VehicleTestbedEditor.Target.cs

## 4.40 VehicleTestbedTarget Class Reference

Inheritance diagram for VehicleTestbedTarget:



### Public Member Functions

- **VehicleTestbedTarget** (TargetInfo Target)

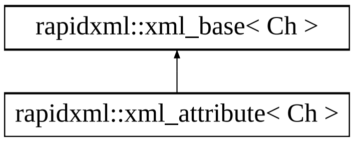
The documentation for this class was generated from the following file:

- Source/VehicleTestbed.Target.cs

## 4.41 rapidxml::xml_attribute< Ch > Class Template Reference

```
#include <rapidxml.hpp>
```

Inheritance diagram for rapidxml::xml_attribute< Ch >:



## Public Member Functions

- xml_attribute ()
- xml_document< Ch > ∗ document () const
- xml_attribute< Ch > ∗ previous_attribute (const Ch ∗name=0, std::size_t name_size=0, bool case_↩ sensitive=true) const
- xml_attribute< Ch > ∗ next_attribute (const Ch ∗name=0, std::size_t name_size=0, bool case_↩ sensitive=true) const

## Friends

- class **xml_node< Ch >**

## Additional Inherited Members

### 4.41.1 Detailed Description

**template<class Ch = char>**
**class rapidxml::xml_attribute< Ch >**

Class representing attribute node of XML document. Each attribute has name and value strings, which are available through name() and value() functions (inherited from xml_base). Note that after parse, both name and value of attribute will point to interior of source text used for parsing. Thus, this text must persist in memory for the lifetime of attribute.

**Parameters**

| | |
|---|---|
| *Ch* | Character type to use. |

### 4.41.2 Constructor & Destructor Documentation

#### 4.41.2.1 xml_attribute()

```
template<class Ch = char>
rapidxml::xml_attribute< Ch >::xml_attribute ( )  [inline]
```

Constructs an empty attribute with the specified type. Consider using memory_pool of appropriate xml_document if allocating attributes manually.

### 4.41.3 Member Function Documentation

#### 4.41.3.1 document()

```
template<class Ch = char>
xml_document<Ch>* rapidxml::xml_attribute< Ch >::document ( ) const  [inline]
```

Gets document of which attribute is a child.

**Returns**

Pointer to document that contains this attribute, or 0 if there is no parent document.

#### 4.41.3.2 next_attribute()

```
template<class Ch = char>
xml_attribute<Ch>* rapidxml::xml_attribute< Ch >::next_attribute (
             const Ch * name = 0,
             std::size_t name_size = 0,
             bool case_sensitive = true ) const  [inline]
```

Gets next attribute, optionally matching attribute name.

**Parameters**

| | |
|---|---|
| *name* | Name of attribute to find, or 0 to return next attribute regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero |
| *name_size* | Size of name, in characters, or 0 to have size calculated automatically from string |
| *case_sensitive* | Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters |

**Returns**

Pointer to found attribute, or 0 if not found.

#### 4.41.3.3 previous_attribute()

```
template<class Ch = char>
xml_attribute<Ch>* rapidxml::xml_attribute< Ch >::previous_attribute (
             const Ch * name = 0,
             std::size_t name_size = 0,
             bool case_sensitive = true ) const  [inline]
```

Gets previous attribute, optionally matching attribute name.

**Parameters**

| | |
|---|---|
| *name* | Name of attribute to find, or 0 to return previous attribute regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero |
| *name_size* | Size of name, in characters, or 0 to have size calculated automatically from string |
| *case_sensitive* | Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters |

**Returns**

Pointer to found attribute, or 0 if not found.

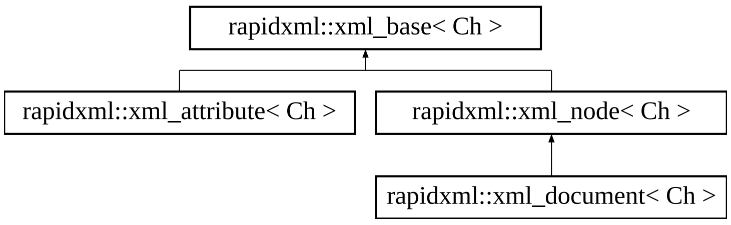
The documentation for this class was generated from the following file:

- Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml.hpp

## 4.42 rapidxml::xml_base< Ch > Class Template Reference

```
#include <rapidxml.hpp>
```

Inheritance diagram for rapidxml::xml_base< Ch >:



### Public Member Functions

- Ch * name () const
- std::size_t name_size () const
- Ch * value () const
- std::size_t value_size () const
- void name (const Ch *name, std::size_t size)
- void name (const Ch *name)
- void value (const Ch *value, std::size_t size)
- void value (const Ch *value)
- xml_node< Ch > * parent () const

### Static Protected Member Functions

- static Ch * **nullstr** ()

### Protected Attributes

- Ch * **m_name**
- Ch * **m_value**
- std::size_t **m_name_size**
- std::size_t **m_value_size**
- xml_node< Ch > * **m_parent**

### 4.42.1 Detailed Description

**template<class Ch = char>**
**class rapidxml::xml_base< Ch >**

Base class for xml_node and xml_attribute implementing common functions: name(), name_size(), value(), value_size() and parent().

**Parameters**

| | |
|---|---|
| *Ch* | Character type to use |

## 4.42.2 Member Function Documentation

### 4.42.2.1 name() [1/3]

```
template<class Ch = char>
Ch* rapidxml::xml_base< Ch >::name ( ) const  [inline]
```

Gets name of the node. Interpretation of name depends on type of node. Note that name will not be zero-terminated if rapidxml::parse_no_string_terminators option was selected during parse.

Use name_size() function to determine length of the name.

**Returns**

Name of node, or empty string if node has no name.

### 4.42.2.2 name() [2/3]

```
template<class Ch = char>
void rapidxml::xml_base< Ch >::name (
             const Ch * name,
             std::size_t size )  [inline]
```

Sets name of node to a non zero-terminated string. See ownership_of_strings.

Note that node does not own its name or value, it only stores a pointer to it. It will not delete or otherwise free the pointer on destruction. It is reponsibility of the user to properly manage lifetime of the string. The easiest way to achieve it is to use memory_pool of the document to allocate the string - on destruction of the document the string will be automatically freed.

Size of name must be specified separately, because name does not have to be zero terminated. Use name(const Ch ∗) function to have the length automatically calculated (string must be zero terminated).

**Parameters**

| | |
|---|---|
| *name* | Name of node to set. Does not have to be zero terminated. |
| *size* | Size of name, in characters. This does not include zero terminator, if one is present. |

#### 4.42.2.3 name() [3/3]

```
template<class Ch = char>
void rapidxml::xml_base< Ch >::name (
             const Ch * name )  [inline]
```

Sets name of node to a zero-terminated string. See also ownership_of_strings and xml_node::name(const Ch *, std::size_t).

**Parameters**

| | |
|---|---|
| *name* | Name of node to set. Must be zero terminated. |

#### 4.42.2.4 name_size()

```
template<class Ch = char>
std::size_t rapidxml::xml_base< Ch >::name_size ( ) const  [inline]
```

Gets size of node name, not including terminator character. This function works correctly irrespective of whether name is or is not zero terminated.

**Returns**

Size of node name, in characters.

#### 4.42.2.5 parent()

```
template<class Ch = char>
xml_node<Ch>* rapidxml::xml_base< Ch >::parent ( ) const  [inline]
```

Gets node parent.

**Returns**

Pointer to parent node, or 0 if there is no parent.

#### 4.42.2.6 value() [1/3]

```
template<class Ch = char>
Ch* rapidxml::xml_base< Ch >::value ( ) const  [inline]
```

Gets value of node. Interpretation of value depends on type of node. Note that value will not be zero-terminated if rapidxml::parse_no_string_terminators option was selected during parse.

Use value_size() function to determine length of the value.

**Returns**

Value of node, or empty string if node has no value.

#### 4.42.2.7 value() [2/3]

```
template<class Ch = char>
void rapidxml::xml_base< Ch >::value (
             const Ch * value,
             std::size_t size )  [inline]
```

Sets value of node to a non zero-terminated string. See ownership_of_strings.

Note that node does not own its name or value, it only stores a pointer to it. It will not delete or otherwise free the pointer on destruction. It is reponsibility of the user to properly manage lifetime of the string. The easiest way to achieve it is to use memory_pool of the document to allocate the string - on destruction of the document the string will be automatically freed.

Size of value must be specified separately, because it does not have to be zero terminated. Use value(const Ch *) function to have the length automatically calculated (string must be zero terminated).

If an element has a child node of type node_data, it will take precedence over element value when printing. If you want to manipulate data of elements using values, use parser flag rapidxml::parse_no_data_nodes to prevent creation of data nodes by the parser.

**Parameters**

| | |
|---|---|
| *value* | value of node to set. Does not have to be zero terminated. |
| *size* | Size of value, in characters. This does not include zero terminator, if one is present. |

#### 4.42.2.8 value() [3/3]

```
template<class Ch = char>
void rapidxml::xml_base< Ch >::value (
             const Ch * value )  [inline]
```

Sets value of node to a zero-terminated string. See also ownership_of_strings and xml_node::value(const Ch *, std::size_t).

**Parameters**

| | |
|---|---|
| *value* | Vame of node to set. Must be zero terminated. |

#### 4.42.2.9 value_size()

```
template<class Ch = char>
std::size_t rapidxml::xml_base< Ch >::value_size ( ) const  [inline]
```

Gets size of node value, not including terminator character. This function works correctly irrespective of whether value is or is not zero terminated.

**Returns**

Size of node value, in characters.

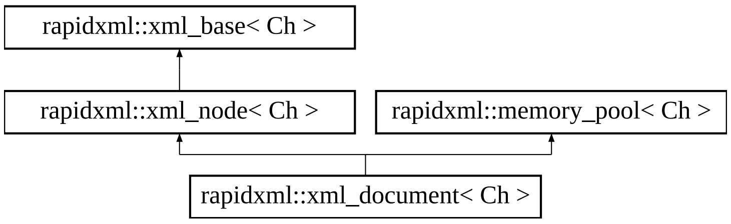
The documentation for this class was generated from the following file:

- Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml.hpp

## 4.43 rapidxml::xml_document< Ch > Class Template Reference

```
#include <rapidxml.hpp>
```

Inheritance diagram for rapidxml::xml_document< Ch >:



### Public Member Functions

- xml_document ()

  *Constructs empty XML document.*
- template<int Flags>
  void parse (Ch ∗text)
- void clear ()

### Additional Inherited Members

### 4.43.1 Detailed Description

**template<class Ch = char>**
**class rapidxml::xml_document< Ch >**

This class represents root of the DOM hierarchy. It is also an xml_node and a memory_pool through public inheritance. Use parse() function to build a DOM tree from a zero-terminated XML text string. parse() function allocates memory for nodes and attributes by using functions of xml_document, which are inherited from memory_pool. To access root node of the document, use the document itself, as if it was an xml_node.

**Parameters**

| | |
|---|---|
| *Ch* | Character type to use. |

### 4.43.2 Member Function Documentation

#### 4.43.2.1 clear()

```
template<class Ch = char>
void rapidxml::xml_document< Ch >::clear ( )  [inline]
```

Clears the document by deleting all nodes and clearing the memory pool. All nodes owned by document pool are destroyed.

#### 4.43.2.2 parse()

```
template<class Ch = char>
template<int Flags>
void rapidxml::xml_document< Ch >::parse (
             Ch * text )  [inline]
```

Parses zero-terminated XML string according to given flags. Passed string will be modified by the parser, unless rapidxml::parse_non_destructive flag is used. The string must persist for the lifetime of the document. In case of error, rapidxml::parse_error exception will be thrown.

If you want to parse contents of a file, you must first load the file into the memory, and pass pointer to its beginning. Make sure that data is zero-terminated.

Document can be parsed into multiple times. Each new call to parse removes previous nodes and attributes (if any), but does not clear memory pool.

**Parameters**

| | |
|---|---|
| *text* | XML data to parse; pointer is non-const to denote fact that this data may be modified by the parser. |

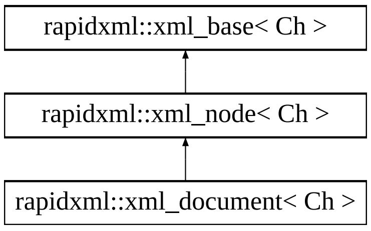
The documentation for this class was generated from the following file:

- Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml.hpp

## 4.44 rapidxml::xml_node< Ch > Class Template Reference

```
#include <rapidxml.hpp>
```

Inheritance diagram for rapidxml::xml_node< Ch >:



## Public Member Functions

- xml_node (node_type type)
- node_type type () const
- xml_document< Ch > ∗ document () const
- xml_node< Ch > ∗ first_node (const Ch ∗name=0, std::size_t name_size=0, bool case_sensitive=true) const
- xml_node< Ch > ∗ last_node (const Ch ∗name=0, std::size_t name_size=0, bool case_sensitive=true) const
- xml_node< Ch > ∗ previous_sibling (const Ch ∗name=0, std::size_t name_size=0, bool case_sensitive=true) const
- xml_node< Ch > ∗ next_sibling (const Ch ∗name=0, std::size_t name_size=0, bool case_sensitive=true) const
- xml_attribute< Ch > ∗ first_attribute (const Ch ∗name=0, std::size_t name_size=0, bool case_sensitive=true) const
- xml_attribute< Ch > ∗ last_attribute (const Ch ∗name=0, std::size_t name_size=0, bool case_sensitive=true) const
- void type (node_type type)
- void prepend_node (xml_node< Ch > ∗child)
- void append_node (xml_node< Ch > ∗child)
- void insert_node (xml_node< Ch > ∗where, xml_node< Ch > ∗child)
- void remove_first_node ()
- void remove_last_node ()
- void remove_node (xml_node< Ch > ∗where)

  *Removes specified child from the node.*
- void remove_all_nodes ()

  *Removes all child nodes (but not attributes).*
- void prepend_attribute (xml_attribute< Ch > ∗attribute)
- void append_attribute (xml_attribute< Ch > ∗attribute)
- void insert_attribute (xml_attribute< Ch > ∗where, xml_attribute< Ch > ∗attribute)
- void remove_first_attribute ()
- void remove_last_attribute ()
- void remove_attribute (xml_attribute< Ch > ∗where)
- void remove_all_attributes ()

  *Removes all attributes of node.*

## Additional Inherited Members

### 4.44.1 Detailed Description

**template<class Ch = char>**
**class rapidxml::xml_node< Ch >**

Class representing a node of XML document. Each node may have associated name and value strings, which are available through name() and value() functions. Interpretation of name and value depends on type of the node. Type of node can be determined by using type() function.

Note that after parse, both name and value of node, if any, will point interior of source text used for parsing. Thus, this text must persist in the memory for the lifetime of node.

**Parameters**

| | |
|---|---|
| *Ch* | Character type to use. |

## 4.44.2 Constructor & Destructor Documentation

### 4.44.2.1 xml_node()

```
template<class Ch = char>
rapidxml::xml_node< Ch >::xml_node (
             node_type type )  [inline]
```

Constructs an empty node with the specified type. Consider using memory_pool of appropriate document to allocate nodes manually.

**Parameters**

| | |
|---|---|
| *type* | Type of node to construct. |

## 4.44.3 Member Function Documentation

### 4.44.3.1 append_attribute()

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::append_attribute (
             xml_attribute< Ch > * attribute )  [inline]
```

Appends a new attribute to the node.

**Parameters**

| | |
|---|---|
| *attribute* | Attribute to append. |

### 4.44.3.2 append_node()

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::append_node (
             xml_node< Ch > * child )  [inline]
```

Appends a new child node. The appended child becomes the last child.

**Parameters**

| | |
|---|---|
| *child* | Node to append. |

### 4.44.3.3 document()

```
template<class Ch = char>
xml_document<Ch>* rapidxml::xml_node< Ch >::document ( ) const  [inline]
```

Gets document of which node is a child.

**Returns**

Pointer to document that contains this node, or 0 if there is no parent document.

### 4.44.3.4 first_attribute()

```
template<class Ch = char>
xml_attribute<Ch>* rapidxml::xml_node< Ch >::first_attribute (
             const Ch * name = 0,
             std::size_t name_size = 0,
             bool case_sensitive = true ) const  [inline]
```

Gets first attribute of node, optionally matching attribute name.

**Parameters**

| | |
|---|---|
| *name* | Name of attribute to find, or 0 to return first attribute regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero |
| *name_size* | Size of name, in characters, or 0 to have size calculated automatically from string |
| *case_sensitive* | Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters |

**Returns**

Pointer to found attribute, or 0 if not found.

### 4.44.3.5 first_node()

```
template<class Ch = char>
xml_node<Ch>* rapidxml::xml_node< Ch >::first_node (
             const Ch * name = 0,
             std::size_t name_size = 0,
             bool case_sensitive = true ) const  [inline]
```

Gets first child node, optionally matching node name.

**Parameters**

| | |
|---|---|
| *name* | Name of child to find, or 0 to return first child regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero |
| *name_size* | Size of name, in characters, or 0 to have size calculated automatically from string |
| *case_sensitive* | Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters |

**Returns**

Pointer to found child, or 0 if not found.

#### 4.44.3.6 insert_attribute()

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::insert_attribute (
            xml_attribute< Ch > * where,
            xml_attribute< Ch > * attribute )  [inline]
```

Inserts a new attribute at specified place inside the node. All attributes after and including the specified attribute are moved one position back.

**Parameters**

| | |
|---|---|
| *where* | Place where to insert the attribute, or 0 to insert at the back. |
| *attribute* | Attribute to insert. |

#### 4.44.3.7 insert_node()

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::insert_node (
            xml_node< Ch > * where,
            xml_node< Ch > * child )  [inline]
```

Inserts a new child node at specified place inside the node. All children after and including the specified node are moved one position back.

**Parameters**

| | |
|---|---|
| *where* | Place where to insert the child, or 0 to insert at the back. |
| *child* | Node to insert. |

#### 4.44.3.8 last_attribute()

```
template<class Ch = char>
xml_attribute<Ch>* rapidxml::xml_node< Ch >::last_attribute (
             const Ch * name = 0,
             std::size_t name_size = 0,
             bool case_sensitive = true ) const  [inline]
```

Gets last attribute of node, optionally matching attribute name.

**Parameters**

| | |
|---|---|
| *name* | Name of attribute to find, or 0 to return last attribute regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero |
| *name_size* | Size of name, in characters, or 0 to have size calculated automatically from string |
| *case_sensitive* | Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters |

**Returns**

Pointer to found attribute, or 0 if not found.

#### 4.44.3.9 last_node()

```
template<class Ch = char>
xml_node<Ch>* rapidxml::xml_node< Ch >::last_node (
             const Ch * name = 0,
             std::size_t name_size = 0,
             bool case_sensitive = true ) const  [inline]
```

Gets last child node, optionally matching node name. Behaviour is undefined if node has no children. Use first_node() to test if node has children.

**Parameters**

| | |
|---|---|
| *name* | Name of child to find, or 0 to return last child regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero |
| *name_size* | Size of name, in characters, or 0 to have size calculated automatically from string |
| *case_sensitive* | Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters |

**Returns**

Pointer to found child, or 0 if not found.

#### 4.44.3.10 next_sibling()

```
template<class Ch = char>
xml_node<Ch>* rapidxml::xml_node< Ch >::next_sibling (
             const Ch * name = 0,
             std::size_t name_size = 0,
             bool case_sensitive = true ) const  [inline]
```

Gets next sibling node, optionally matching node name. Behaviour is undefined if node has no parent. Use parent() to test if node has a parent.

**Parameters**

| | |
|---|---|
| *name* | Name of sibling to find, or 0 to return next sibling regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero |
| *name_size* | Size of name, in characters, or 0 to have size calculated automatically from string |
| *case_sensitive* | Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters |

**Returns**

Pointer to found sibling, or 0 if not found.

#### 4.44.3.11 prepend_attribute()

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::prepend_attribute (
             xml_attribute< Ch > * attribute )  [inline]
```

Prepends a new attribute to the node.

**Parameters**

| | |
|---|---|
| *attribute* | Attribute to prepend. |

#### 4.44.3.12 prepend_node()

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::prepend_node (
             xml_node< Ch > * child )  [inline]
```

Prepends a new child node. The prepended child becomes the first child, and all existing children are moved one position back.

**Parameters**

| | |
|---|---|
| *child* | Node to prepend. |

#### 4.44.3.13 previous_sibling()

```
template<class Ch = char>
xml_node<Ch>* rapidxml::xml_node< Ch >::previous_sibling (
             const Ch * name = 0,
             std::size_t name_size = 0,
             bool case_sensitive = true ) const  [inline]
```

Gets previous sibling node, optionally matching node name. Behaviour is undefined if node has no parent. Use parent() to test if node has a parent.

**Parameters**

| | |
|---|---|
| *name* | Name of sibling to find, or 0 to return previous sibling regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero |
| *name_size* | Size of name, in characters, or 0 to have size calculated automatically from string |
| *case_sensitive* | Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters |

**Returns**

Pointer to found sibling, or 0 if not found.

#### 4.44.3.14 remove_attribute()

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::remove_attribute (
             xml_attribute< Ch > * where )  [inline]
```

Removes specified attribute from node.

**Parameters**

| | |
|---|---|
| *where* | Pointer to attribute to be removed. |

#### 4.44.3.15 remove_first_attribute()

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::remove_first_attribute ( )  [inline]
```

Removes first attribute of the node. If node has no attributes, behaviour is undefined. Use first_attribute() to test if node has attributes.

#### 4.44.3.16 remove_first_node()

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::remove_first_node ( )  [inline]
```

Removes first child node. If node has no children, behaviour is undefined. Use first_node() to test if node has children.

#### 4.44.3.17 remove_last_attribute()

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::remove_last_attribute ( )  [inline]
```

Removes last attribute of the node. If node has no attributes, behaviour is undefined. Use first_attribute() to test if node has attributes.

#### 4.44.3.18 remove_last_node()

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::remove_last_node ( )  [inline]
```

Removes last child of the node. If node has no children, behaviour is undefined. Use first_node() to test if node has children.

#### 4.44.3.19 type() [1/2]

```
template<class Ch = char>
node_type rapidxml::xml_node< Ch >::type ( ) const  [inline]
```

Gets type of node.

**Returns**

Type of node.

#### 4.44.3.20 type() [2/2]

```
template<class Ch = char>
void rapidxml::xml_node< Ch >::type (
             node_type type )  [inline]
```

Sets type of node.

**Parameters**

| | |
|---|---|
| *type* | Type of node to set. |

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

- Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml.hpp

# Chapter 5

# File Documentation

## 5.1 Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml.hpp File Reference

This file contains rapidxml parser and DOM implementation.

```
#include <cstdlib>
#include <cassert>
#include <new>
#include <exception>
```

### Classes

- class rapidxml::parse_error
- class rapidxml::xml_node< Ch >
- class rapidxml::xml_attribute< Ch >
- class rapidxml::xml_document< Ch >
- class rapidxml::memory_pool< Ch >
- class rapidxml::xml_base< Ch >
- class rapidxml::xml_attribute< Ch >
- class rapidxml::xml_node< Ch >
- class rapidxml::xml_document< Ch >

### Macros

- #define **RAPIDXML_PARSE_ERROR**(what, where) throw parse_error(what, where)
- #define **RAPIDXML_STATIC_POOL_SIZE** (64 ∗ 1024)
- #define **RAPIDXML_DYNAMIC_POOL_SIZE** (64 ∗ 1024)
- #define **RAPIDXML_ALIGNMENT** sizeof(void ∗)

### Enumerations

- enum rapidxml::node_type {
  rapidxml::node_document, rapidxml::node_element, rapidxml::node_data, rapidxml::node_cdata,
  rapidxml::node_comment, rapidxml::node_declaration, rapidxml::node_doctype, rapidxml::node_pi }

## Variables

- const int rapidxml::parse_no_data_nodes = 0x1
- const int rapidxml::parse_no_element_values = 0x2
- const int rapidxml::parse_no_string_terminators = 0x4
- const int rapidxml::parse_no_entity_translation = 0x8
- const int rapidxml::parse_no_utf8 = 0x10
- const int rapidxml::parse_declaration_node = 0x20
- const int rapidxml::parse_comment_nodes = 0x40
- const int rapidxml::parse_doctype_node = 0x80
- const int rapidxml::parse_pi_nodes = 0x100
- const int rapidxml::parse_validate_closing_tags = 0x200
- const int rapidxml::parse_trim_whitespace = 0x400
- const int rapidxml::parse_normalize_whitespace = 0x800
- const int rapidxml::parse_default = 0
- const int rapidxml::parse_non_destructive = parse_no_string_terminators | parse_no_entity_translation
- const int rapidxml::parse_fastest = parse_non_destructive | parse_no_data_nodes
- const int rapidxml::parse_full = parse_declaration_node | parse_comment_nodes | parse_doctype_node | parse_pi_nodes | parse_validate_closing_tags

### 5.1.1 Detailed Description

This file contains rapidxml parser and DOM implementation.

### 5.1.2 Enumeration Type Documentation

#### 5.1.2.1 node_type

```
enum rapidxml::node_type
```

Enumeration listing all node types produced by the parser. Use xml_node::type() function to query node type.

**Enumerator**

| | |
|---|---|
| node_document | A document node. Name and value are empty. |
| node_element | An element node. Name contains element name. Value contains text of first data node. |
| node_data | A data node. Name is empty. Value contains data text. |
| node_cdata | A CDATA node. Name is empty. Value contains data text. |
| node_comment | A comment node. Name is empty. Value contains comment text. |
| node_declaration | A declaration node. Name and value are empty. Declaration parameters (version, encoding and standalone) are in node attributes. |
| node_doctype | A DOCTYPE node. Name is empty. Value contains DOCTYPE text. |
| node_pi | A PI node. Name contains target. Value contains instructions. |

### 5.1.3 Variable Documentation

#### 5.1.3.1 parse_comment_nodes

```
const int rapidxml::parse_comment_nodes = 0x40
```

Parse flag instructing the parser to create comments nodes. By default, comment nodes are not created. Can be combined with other flags by use of | operator.

See xml_document::parse() function.

#### 5.1.3.2 parse_declaration_node

```
const int rapidxml::parse_declaration_node = 0x20
```

Parse flag instructing the parser to create XML declaration node. By default, declaration node is not created. Can be combined with other flags by use of | operator.

See xml_document::parse() function.

#### 5.1.3.3 parse_default

```
const int rapidxml::parse_default = 0
```

Parse flags which represent default behaviour of the parser. This is always equal to 0, so that all other flags can be simply ored together. Normally there is no need to inconveniently disable flags by anding with their negated (∼) values. This also means that meaning of each flag is a *negation* of the default setting. For example, if flag name is rapidxml::parse_no_utf8, it means that utf-8 is *enabled* by default, and using the flag will disable it.

See xml_document::parse() function.

#### 5.1.3.4 parse_doctype_node

```
const int rapidxml::parse_doctype_node = 0x80
```

Parse flag instructing the parser to create DOCTYPE node. By default, doctype node is not created. Although W3C specification allows at most one DOCTYPE node, RapidXml will silently accept documents with more than one. Can be combined with other flags by use of | operator.

See xml_document::parse() function.

#### 5.1.3.5 parse_fastest

```
const int rapidxml::parse_fastest = parse_non_destructive | parse_no_data_nodes
```

A combination of parse flags resulting in fastest possible parsing, without sacrificing important data.

See xml_document::parse() function.

#### 5.1.3.6 parse_full

```
const int rapidxml::parse_full = parse_declaration_node | parse_comment_nodes | parse_doctype↩
_node | parse_pi_nodes | parse_validate_closing_tags
```

A combination of parse flags resulting in largest amount of data being extracted. This usually results in slowest parsing.

See xml_document::parse() function.

#### 5.1.3.7 parse_no_data_nodes

```
const int rapidxml::parse_no_data_nodes = 0x1
```

Parse flag instructing the parser to not create data nodes. Text of first data node will still be placed in value of parent element, unless rapidxml::parse_no_element_values flag is also specified. Can be combined with other flags by use of | operator.

See xml_document::parse() function.

#### 5.1.3.8 parse_no_element_values

```
const int rapidxml::parse_no_element_values = 0x2
```

Parse flag instructing the parser to not use text of first data node as a value of parent element. Can be combined with other flags by use of | operator. Note that child data nodes of element node take precendence over its value when printing. That is, if element has one or more child data nodes *and* a value, the value will be ignored. Use rapidxml::parse_no_data_nodes flag to prevent creation of data nodes if you want to manipulate data using values of elements.

See xml_document::parse() function.

#### 5.1.3.9 parse_no_entity_translation

```
const int rapidxml::parse_no_entity_translation = 0x8
```

Parse flag instructing the parser to not translate entities in the source text. By default entities are translated, modifying source text. Can be combined with other flags by use of | operator.

See xml_document::parse() function.

#### 5.1.3.10 parse_no_string_terminators

```
const int rapidxml::parse_no_string_terminators = 0x4
```

Parse flag instructing the parser to not place zero terminators after strings in the source text. By default zero terminators are placed, modifying source text. Can be combined with other flags by use of | operator.

See xml_document::parse() function.

#### 5.1.3.11 parse_no_utf8

```
const int rapidxml::parse_no_utf8 = 0x10
```

Parse flag instructing the parser to disable UTF-8 handling and assume plain 8 bit characters. By default, UTF-8 handling is enabled. Can be combined with other flags by use of | operator.

See xml_document::parse() function.

#### 5.1.3.12 parse_non_destructive

```
const int rapidxml::parse_non_destructive = parse_no_string_terminators | parse_no_entity_↩
translation
```

A combination of parse flags that forbids any modifications of the source text. This also results in faster parsing. However, note that the following will occur:

- names and values of nodes will not be zero terminated, you have to use xml_base::name_size() and xml_↩ base::value_size() functions to determine where name and value ends
- entities will not be translated
- whitespace will not be normalized

See xml_document::parse() function.

#### 5.1.3.13 parse_normalize_whitespace

```
const int rapidxml::parse_normalize_whitespace = 0x800
```

Parse flag instructing the parser to condense all whitespace runs of data nodes to a single space character. Trimming of leading and trailing whitespace of data is controlled by rapidxml::parse_trim_whitespace flag. By default, whitespace is not normalized. If this flag is specified, source text will be modified. Can be combined with other flags by use of | operator.

See xml_document::parse() function.

#### 5.1.3.14 parse_pi_nodes

```
const int rapidxml::parse_pi_nodes = 0x100
```

Parse flag instructing the parser to create PI nodes. By default, PI nodes are not created. Can be combined with other flags by use of | operator.

See xml_document::parse() function.

#### 5.1.3.15 parse_trim_whitespace

```
const int rapidxml::parse_trim_whitespace = 0x400
```

Parse flag instructing the parser to trim all leading and trailing whitespace of data nodes. By default, whitespace is not trimmed. This flag does not cause the parser to modify source text. Can be combined with other flags by use of | operator.

See xml_document::parse() function.

#### 5.1.3.16 parse_validate_closing_tags

```
const int rapidxml::parse_validate_closing_tags = 0x200
```

Parse flag instructing the parser to validate closing tag names. If not set, name inside closing tag is irrelevant to the parser. By default, closing tags are not validated. Can be combined with other flags by use of | operator.

See xml_document::parse() function.

## 5.2 Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml_iterators.hpp File Reference

This file contains rapidxml iterators.

```
#include "rapidxml.hpp"
```

### Classes

- class rapidxml::node_iterator< Ch >

    *Iterator of child nodes of xml_node.*
- class rapidxml::attribute_iterator< Ch >

    *Iterator of child attributes of xml_node.*

### 5.2.1 Detailed Description

This file contains rapidxml iterators.

## 5.3 Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml_print.hpp File Reference

This file contains rapidxml printer implementation.

```
#include "rapidxml.hpp"
#include <ostream>
#include <iterator>
```

### Functions

- template<class OutIt , class Ch >
  OutIt rapidxml::print (OutIt out, const xml_node< Ch > &node, int flags=0)
- template<class Ch >
  std::basic_ostream< Ch > & rapidxml::print (std::basic_ostream< Ch > &out, const xml_node< Ch > &node, int flags=0)
- template<class Ch >
  std::basic_ostream< Ch > & rapidxml::operator<< (std::basic_ostream< Ch > &out, const xml_node< Ch > &node)

### Variables

- const int rapidxml::print_no_indenting = 0x1

  *Printer flag instructing the printer to suppress indenting of XML. See print() function.*

## 5.3.1 Detailed Description

This file contains rapidxml printer implementation.

## 5.3.2 Function Documentation

### 5.3.2.1 operator<<()

```
template<class Ch >
std::basic_ostream<Ch>& rapidxml::operator<< (
            std::basic_ostream< Ch > & out,
            const xml_node< Ch > & node )  [inline]
```

Prints formatted XML to given output stream. Uses default printing flags. Use print() function to customize printing process.

**Parameters**

| | |
|---|---|
| *out* | Output stream to print to. |
| *node* | Node to be printed. |

**Returns**

Output stream.

### 5.3.2.2 print() [1/2]

```
template<class OutIt , class Ch >
OutIt rapidxml::print (
            OutIt out,
            const xml_node< Ch > & node,
            int flags = 0 )  [inline]
```

Prints XML to given output iterator.

**Parameters**

| | |
|---|---|
| *out* | Output iterator to print to. |
| *node* | Node to be printed. Pass xml_document to print entire document. |
| *flags* | Flags controlling how XML is printed. |

**Returns**

Output iterator pointing to position immediately after last character of printed text.

#### 5.3.2.3 print() [2/2]

```
template<class Ch >
std::basic_ostream<Ch>& rapidxml::print (
             std::basic_ostream< Ch > & out,
             const xml_node< Ch > & node,
             int flags = 0 )  [inline]
```

Prints XML to given output stream.

**Parameters**

| | |
|---|---|
| *out* | Output stream to print to. |
| *node* | Node to be printed. Pass xml_document to print entire document. |
| *flags* | Flags controlling how XML is printed. |

**Returns**

Output stream.

## 5.4 Source/VehicleTestbed/Public/Configurator/RapidXML/rapidxml_utils.hpp File Reference

```
#include "rapidxml.hpp"
#include <vector>
#include <string>
#include <fstream>
#include <stdexcept>
```

### Classes

- class rapidxml::file< Ch >

  *Represents data loaded from a file.*

### Functions

- template<class Ch >
  std::size_t rapidxml::count_children (xml_node< Ch > *node)
- template<class Ch >
  std::size_t rapidxml::count_attributes (xml_node< Ch > *node)

### 5.4.1 Detailed Description

This file contains high-level rapidxml utilities that can be useful in certain simple scenarios. They should probably not be used if maximizing performance is the main objective.

### 5.4.2 Function Documentation

#### 5.4.2.1 count_attributes()

```
template<class Ch >
std::size_t rapidxml::count_attributes (
            xml_node< Ch > * node )  [inline]
```

Counts attributes of node. Time complexity is O(n).

**Returns**

Number of attributes of node

#### 5.4.2.2 count_children()

```
template<class Ch >
std::size_t rapidxml::count_children (
            xml_node< Ch > * node )  [inline]
```

Counts children of node. Time complexity is O(n).

**Returns**

Number of children of node

# Index