

UACS

1.0.0

Generated by Doxygen 1.12.0

| | |
|---|----------|
| 1 Hierarchical Index | 1 |
| 1.1 Class Hierarchy | 1 |
| 2 Class Index | 3 |
| 2.1 Class List | 3 |
| 3 File Index | 5 |
| 3.1 File List | 5 |
| 4 Class Documentation | 7 |
| 4.1 UACS::ActionInfo Struct Reference | 7 |
| 4.2 UACS::AirlockInfo Struct Reference | 7 |
| 4.2.1 Member Data Documentation | 8 |
| 4.2.1.1 dir | 8 |
| 4.2.1.2 rot | 8 |
| 4.3 UACS::AstrInfo Struct Reference | 8 |
| 4.3.1 Member Data Documentation | 9 |
| 4.3.1.1 className | 9 |
| 4.4 UACS::Astronaut Class Reference | 9 |
| 4.4.1 Detailed Description | 10 |
| 4.4.2 Member Function Documentation | 10 |
| 4.4.2.1 clbkGetAstrInfo() | 10 |
| 4.4.2.2 clbkSetAstrInfo() | 10 |
| 4.4.2.3 GetAstrInfoByHandle() | 10 |
| 4.4.2.4 GetAstrInfoByIndex() | 11 |
| 4.4.2.5 GetNearestAction() | 11 |
| 4.4.2.6 GetNearestAirlock() | 11 |
| 4.4.2.7 GetNearestBreathable() | 12 |
| 4.4.2.8 GetScnAstrCount() | 12 |
| 4.4.2.9 GetUACSVersion() | 12 |
| 4.4.2.10 GetVslAstrInfo() | 12 |
| 4.4.2.11 InBreathable() | 13 |
| 4.4.2.12 Ingress() | 13 |
| 4.4.2.13 TriggerAction() | 13 |
| 4.5 UACS::Cargo Class Reference | 14 |
| 4.5.1 Detailed Description | 15 |
| 4.5.2 Member Function Documentation | 15 |
| 4.5.2.1 clbkDrainResource() | 15 |
| 4.5.2.2 clbkGetCargoInfo() | 15 |
| 4.5.2.3 clbkPackCargo() | 15 |
| 4.5.2.4 clbkUnpackCargo() | 15 |
| 4.5.2.5 GetUACSVersion() | 16 |
| 4.6 UACS::Cargo::CargoInfo Struct Reference | 16 |

| | |
|--|----|
| 4.6.1 Detailed Description | 16 |
| 4.6.2 Member Data Documentation | 17 |
| 4.6.2.1 breathable | 17 |
| 4.6.2.2 frontPos | 17 |
| 4.6.2.3 resource | 17 |
| 4.6.2.4 unpacked | 17 |
| 4.7 UACS::CargoInfo Struct Reference | 18 |
| 4.7.1 Member Data Documentation | 18 |
| 4.7.1.1 breathable | 18 |
| 4.8 UACS::GroundInfo Struct Reference | 18 |
| 4.8.1 Member Data Documentation | 19 |
| 4.8.1.1 colDir | 19 |
| 4.8.1.2 pos | 19 |
| 4.8.1.3 rowDir | 19 |
| 4.9 UACS::Module Class Reference | 19 |
| 4.9.1 Detailed Description | 21 |
| 4.9.2 Constructor & Destructor Documentation | 21 |
| 4.9.2.1 Module() | 21 |
| 4.9.3 Member Function Documentation | 22 |
| 4.9.3.1 AddAstronaut() | 22 |
| 4.9.3.2 AddCargo() | 22 |
| 4.9.3.3 clbkSaveState() | 22 |
| 4.9.3.4 DeleteCargo() | 23 |
| 4.9.3.5 DrainGrappledResource() | 23 |
| 4.9.3.6 DrainScenarioResource() | 23 |
| 4.9.3.7 DrainStationResource() | 24 |
| 4.9.3.8 DrawAstrInfo() | 24 |
| 4.9.3.9 DrawCargoInfo() | 25 |
| 4.9.3.10 EgressAstronaut() | 25 |
| 4.9.3.11 GetAstrInfoByHandle() | 25 |
| 4.9.3.12 GetAstrInfoByIndex() | 26 |
| 4.9.3.13 GetAvailAstrCount() | 27 |
| 4.9.3.14 GetAvailAstrName() | 27 |
| 4.9.3.15 GetAvailCargoCount() | 27 |
| 4.9.3.16 GetAvailCargoName() | 27 |
| 4.9.3.17 GetCargoInfoByHandle() | 28 |
| 4.9.3.18 GetCargoInfoByIndex() | 28 |
| 4.9.3.19 GetScnAstrCount() | 28 |
| 4.9.3.20 GetScnCargoCount() | 29 |
| 4.9.3.21 GetStationResources() | 29 |
| 4.9.3.22 GetTotalAstrMass() | 29 |
| 4.9.3.23 GetTotalCargoMass() | 29 |

| | |
|--|-----------|
| 4.9.3.24 GetUACSVersion() | 30 |
| 4.9.3.25 GetVslAstrInfo() | 30 |
| 4.9.3.26 GrappleCargo() | 31 |
| 4.9.3.27 PackCargo() | 31 |
| 4.9.3.28 ParseScenarioLine() | 31 |
| 4.9.3.29 ReleaseCargo() | 32 |
| 4.9.3.30 SetAstrInfoByHandle() | 32 |
| 4.9.3.31 SetAstrInfoByIndex() | 32 |
| 4.9.3.32 TransferAstronaut() | 33 |
| 4.9.3.33 UnpackCargo() | 33 |
| 4.10 UACS::NearestAction Struct Reference | 34 |
| 4.11 UACS::NearestAirlock Struct Reference | 34 |
| 4.12 UACS::SlotInfo Struct Reference | 34 |
| 4.12.1 Member Data Documentation | 35 |
| 4.12.1.1 holdDir | 35 |
| 4.13 UACS::StationInfo Struct Reference | 35 |
| 4.14 UACS::VslAstrInfo Struct Reference | 35 |
| 4.15 UACS::VslCargoInfo Struct Reference | 36 |
| 4.15.1 Member Data Documentation | 36 |
| 4.15.1.1 astrMode | 36 |
| 5 File Documentation | 37 |
| 5.1 API/Astronaut.h File Reference | 37 |
| 5.1.1 Detailed Description | 37 |
| 5.2 Astronaut.h | 37 |
| 5.3 API/Cargo.h File Reference | 38 |
| 5.3.1 Detailed Description | 38 |
| 5.4 Cargo.h | 39 |
| 5.5 Common.h | 39 |
| 5.6 API/Module.h File Reference | 41 |
| 5.6.1 Detailed Description | 42 |
| 5.6.2 Enumeration Type Documentation | 42 |
| 5.6.2.1 DrainResult | 42 |
| 5.6.2.2 EgressResult | 42 |
| 5.6.2.3 GrappleResult | 42 |
| 5.6.2.4 PackResult | 42 |
| 5.6.2.5 ReleaseResult | 43 |
| 5.6.2.6 TransferResult | 43 |
| 5.7 Module.h | 43 |
| Index | 47 |

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

| | |
|----------------------------------|----|
| UACS::ActionInfo | 7 |
| UACS::AirlockInfo | 7 |
| UACS::AstrInfo | 8 |
| UACS::Cargo::CargoInfo | 16 |
| UACS::CargoInfo | 18 |
| UACS::GroundInfo | 18 |
| UACS::Module | 19 |
| UACS::NearestAction | 34 |
| UACS::NearestAirlock | 34 |
| UACS::SlotInfo | 34 |
| UACS::StationInfo | 35 |
| VESSEL4 | |
| UACS::Astronaut | 9 |
| UACS::Cargo | 14 |
| UACS::VslAstrInfo | 35 |
| UACS::VslCargoInfo | 36 |

Chapter 2

Class Index

2.1 Class List

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

| | |
|---|----|
| UACS::ActionInfo | 7 |
| UACS::AirlockInfo | 7 |
| UACS::AstrInfo | 8 |
| UACS::Astronaut | |
| UACS astronaut API | 9 |
| UACS::Cargo | |
| UACS cargo API | 14 |
| UACS::Cargo::CargoInfo | |
| The cargo information struct. This is the information the cargo must pass to UACS. On the other hand, API::CargoInfo is used to pass cargo information from UACS to vessels | 16 |
| UACS::CargoInfo | 18 |
| UACS::GroundInfo | 18 |
| UACS::Module | |
| UACS module API | 19 |
| UACS::NearestAction | 34 |
| UACS::NearestAirlock | 34 |
| UACS::SlotInfo | 34 |
| UACS::StationInfo | 35 |
| UACS::VslAstrInfo | 35 |
| UACS::VslCargoInfo | 36 |

Chapter 3

File Index

3.1 File List

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

| | |
|-------------------------------------|----|
| API/ Astronaut.h | |
| UACS astronaut API header | 37 |
| API/ Cargo.h | |
| UACS cargo API header | 38 |
| API/ Common.h | |
| UACS module API header | 39 |
| API/ Module.h | |
| UACS module API header | 41 |

Chapter 4

Class Documentation

4.1 UACS::ActionInfo Struct Reference

Public Attributes

- std::string **name**
- VECTOR3 **pos**
The action area position in vessel-relative coordinates.
- double **range** { 5 }
The action area trigger range in meters.
- bool **enabled** { true }

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

- API/Common.h

4.2 UACS::AirlockInfo Struct Reference

Public Attributes

- std::string **name**
- VECTOR3 **pos**
The airlock position in vessel-relative coordinates. It's used to position astronauts when egressed in space.
- VECTOR3 **dir**
The airlock direction in vessel-relative coordinates. It's used to orientate astronauts when egressed in space.
- VECTOR3 **rot**
The airlock longitudinal alignment vector in vessel-relative coordinates. It's used to orientate astronauts when egressed in space.
- double **range** { 5 }
The airlock ingress range in meters. The range is calculated from pos in space, and from gndInfo.pos on ground.
- bool **open** { true }
- double **relVel** { 0 }
The astronaut egress velocity (if egressed in space) in meters per second.
- **GroundInfo** **gndInfo** {}
The airlock ground egress information.
- DOCKHANDLE **hDock** {}
The dock handle associated with the airlock, which is used to transfer astronaut to a docked vessel.

4.2.1 Member Data Documentation

4.2.1.1 dir

```
VECTOR3 UACS::AirlockInfo::dir
```

The airlock direction in vessel-relative coordinates. It's used to orientate astronauts when egressed in space.

Note

It must be perpendicular to rot and normalised.

4.2.1.2 rot

```
VECTOR3 UACS::AirlockInfo::rot
```

The airlock longitudinal alignment vector in vessel-relative coordinates. It's used to orientate astronauts when egressed in space.

Note

It must be perpendicular to dir and normalised.

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

- API/Common.h

4.3 UACS::AstrInfo Struct Reference

Public Attributes

- std::string **name**
This is the astronaut person name, NOT the astronaut vessel name in the scenario.
- std::string **role**
Use standard astronaut roles (see UACS developer manual).
- double **mass**
The astronaut body mass in kilograms. It does NOT include suit, fuel, or oxygen mass.
- double **height**
The astronaut height with the suit on in meters. This is used to set the astronaut height when egressed on ground.
- double **fuelLvl** { 1 }
The astronaut fuel level, from 0 to 1.
- double **oxyLvl** { 1 }
The astronaut oxygen level, from 0 to 1.
- bool **alive** { true }
The astronaut life flag.
- std::string **customData** {}
The astronaut custom data, which is passed to the astronaut when egressed from a vessel.
- std::string **className** {}
The astronaut class name, which is used to spawn the astronaut when egressed from a vessel.

4.3.1 Member Data Documentation

4.3.1.1 className

```
std::string UACS::AstrInfo::className {}
```

The astronaut class name, which is used to spawn the astronaut when egressed from a vessel.

This is the astronaut vessel config file path from 'Config\Vessels' folder without '.cfg'.

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

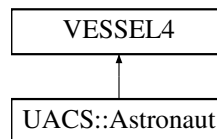
- API/Common.h

4.4 UACS::Astronaut Class Reference

UACS astronaut API.

```
#include <Astronaut.h>
```

Inheritance diagram for UACS::Astronaut:



Public Member Functions

- **Astronaut** (OBJHANDLE hVessel, int fModel=1)
- virtual bool [clbkSetAstrInfo](#) (const [AstrInfo](#) &astrInfo)=0
The astronaut information set callback. It must be implemented by astronauts to change the astronaut status accordingly.
- virtual const [AstrInfo](#) * [clbkGetAstrInfo](#) ()=0
The astronaut information callback. It must be implemented by astronauts.
- std::string_view [GetUACSVersion](#) ()
Gets UACS version.
- size_t [GetScnAstrCount](#) ()
Gets the astronaut count in the scenario.
- std::pair< OBJHANDLE, const [AstrInfo](#) * > [GetAstrInfoByIndex](#) (size_t astrIdx)
Gets an astronaut information by the astronaut index.
- const [AstrInfo](#) * [GetAstrInfoByHandle](#) (OBJHANDLE hAstr)
Gets an astronaut information by the astronaut handle.
- const [VslAstrInfo](#) * [GetVslAstrInfo](#) (OBJHANDLE hVessel)
Gets a vessel astronaut information by the vessel handle.
- std::optional< [NearestAirlock](#) > [GetNearestAirlock](#) (double range, bool airlockOpen=true, bool station↔Empty=true)
Gets the nearest airlock within the passed range.
- std::pair< OBJHANDLE, VECTOR3 > [GetNearestBreathable](#) (double range)

- Gets the nearest breathable vessel (either a cargo or station) within the passed range.*
- std::optional< [NearestAction](#) > [GetNearestAction](#) (double range, bool areaEnabled=true)
- Gets the nearest enabled action area in the passed range.*
- bool [InBreathable](#) (bool checkAtm=true)
- Determines whether the vessel is in a breathable vessel or atmosphere.*
- IngressResult [Ingress](#) (OBJHANDLE hVessel=nullptr, std::optional< size_t > airlockIdx={}, std::optional< size_t > stationIdx={})
- Ingresses the astronaut to the passed station via the passed airlock in the passed vessel.*
- IngressResult [TriggerAction](#) (OBJHANDLE hVessel=nullptr, std::optional< size_t > actionIdx={})
- Triggers the passed action area in the passed vessel.*

4.4.1 Detailed Description

UACS astronaut API.

4.4.2 Member Function Documentation

4.4.2.1 [clbkGetAstrInfo\(\)](#)

```
virtual const AstrInfo * UACS::Astronaut::clbkGetAstrInfo () [pure virtual]
```

The astronaut information callback. It must be implemented by astronauts.

Returns

A const pointer to the [AstrInfo](#) struct, which must live until the astronaut is destroyed. Don't pass the address of a temporary variable.

4.4.2.2 [clbkSetAstrInfo\(\)](#)

```
virtual bool UACS::Astronaut::clbkSetAstrInfo (
    const AstrInfo & astrInfo) [pure virtual]
```

The astronaut information set callback. It must be implemented by astronauts to change the astronaut status accordingly.

Parameters

| | |
|-----------------|----------------------------|
| <i>astrInfo</i> | The astronaut information. |
|-----------------|----------------------------|

Returns

True if the astronaut information is set, false if not.

4.4.2.3 [GetAstrInfoByHandle\(\)](#)

```
const AstrInfo * UACS::Astronaut::GetAstrInfoByHandle (
    OBJHANDLE hAstr)
```

Gets an astronaut information by the astronaut handle.

Parameters

| | |
|--------------|------------------------------|
| <i>hAstr</i> | The astronaut vessel handle. |
|--------------|------------------------------|

Returns

A pointer to the astronaut [AstrInfo](#) struct, or nullptr if hAstr is invalid or not an astronaut.

4.4.2.4 GetAstrInfoByIndex()

```
std::pair< OBJHANDLE, const AstrInfo * > UACS::Astronaut::GetAstrInfoByIndex (
    size_t astrIdx)
```

Gets an astronaut information by the astronaut index.

Parameters

| | |
|----------------|--|
| <i>astrIdx</i> | The astronaut index. It must be less than GetScnAstrCount. |
|----------------|--|

Returns

A pair of the astronaut vessel handle and a pointer to the astronaut [AstrInfo](#) struct.

4.4.2.5 GetNearestAction()

```
std::optional< NearestAction > UACS::Astronaut::GetNearestAction (
    double range,
    bool areaEnabled = true)
```

Gets the nearest enabled action area in the passed range.

Parameters

| | |
|----------------------|--|
| <i>range</i> | The search range in meters. |
| <i>actionEnabled</i> | Set true if the action area must be enabled, false if not. |

Returns

The nearest action.

4.4.2.6 GetNearestAirlock()

```
std::optional< NearestAirlock > UACS::Astronaut::GetNearestAirlock (
    double range,
    bool airlockOpen = true,
    bool stationEmpty = true)
```

Gets the nearest airlock within the passed range.

Parameters

| | |
|---------------------|--|
| <i>range</i> | The search range in meters. |
| <i>airlockOpen</i> | Set true if the airlock must be open, false if not. |
| <i>stationEmpty</i> | Set true if the vessel must have at least one empty station, false if not. |

Returns

The nearest airlock.

4.4.2.7 GetNearestBreathable()

```
std::pair< OBJHANDLE, VECTOR3 > UACS::Astronaut::GetNearestBreathable (
    double range)
```

Gets the nearest breathable vessel (either a cargo or station) within the passed range.

Returns

A pair of the nearest breathable vessel handle and its relative position. If no vessel was found, a nullptr and empty position is returned.

4.4.2.8 GetScnAstrCount()

```
size_t UACS::Astronaut::GetScnAstrCount ()
```

Gets the astronaut count in the scenario.

Returns

The astronaut count in the scenario.

4.4.2.9 GetUACSVersion()

```
std::string_view UACS::Astronaut::GetUACSVersion ()
```

Gets UACS version.

Returns

UACS version.

4.4.2.10 GetVslAstrInfo()

```
const VslAstrInfo * UACS::Astronaut::GetVslAstrInfo (
    OBJHANDLE hVessel)
```

Gets a vessel astronaut information by the vessel handle.

Parameters

| | |
|----------------|--------------------|
| <i>hVessel</i> | The vessel handle. |
|----------------|--------------------|

Returns

A pointer to the vessel [VslAstrInfo](#) struct, or nullptr if hVessel is invalid or not an astronaut.

4.4.2.11 InBreathable()

```
bool UACS::Astronaut::InBreathable (
    bool checkAtm = true)
```

Determines whether the vessel is in a breathable vessel or atmosphere.

The surrounding atmosphere is considered breathable if its temperature is between 223 and 373 kelvin, and pressure between 36 and 250 kPa. The vessel is considered in a breathable vessel if the distance between the vessel and the nearest breathable vessel is less than the breathable vessel radius.

Parameters

| | |
|-----------------|--|
| <i>checkAtm</i> | Set true to check if the vessel is in a breathable atmosphere or vessel, false to check for breathable vessels only. |
|-----------------|--|

Returns

True if the vessel is in a breathable area, false if not.

4.4.2.12 Ingress()

```
IngressResult UACS::Astronaut::Ingress (
    OBJHANDLE hVessel = nullptr,
    std::optional< size_t > airlockIdx = {},
    std::optional< size_t > stationIdx = {})
```

Ingresses the astronaut to the passed station via the passed airlock in the passed vessel.

Parameters

| | |
|-------------------|--|
| <i>hVessel</i> | The vessel handle. If nullptr is passed, the nearest airlock in a 10 km range is used. |
| <i>airlockIdx</i> | The airlock index. If nullopt is passed, the first open airlock is used. |
| <i>stationIdx</i> | The station index. If nullopt is passed, the first empty station is used. |

Returns

The ingress result.

4.4.2.13 TriggerAction()

```
IngressResult UACS::Astronaut::TriggerAction (
    OBJHANDLE hVessel = nullptr,
    std::optional< size_t > actionIdx = {})
```

Triggers the passed action area in the passed vessel.

Parameters

| | |
|------------------|--|
| <i>hVessel</i> | The vessel handle. If nullptr is passed, the nearest action area in a 10 km range is used. |
| <i>actionIdx</i> | The action area index. If nullopt is passed, the first enabled action area is used. |

Returns

The trigger result as the IngressResult enum.

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

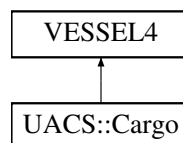
- API/[Astronaut.h](#)
- API/Astronaut.cpp

4.5 UACS::Cargo Class Reference

UACS cargo API.

```
#include <Cargo.h>
```

Inheritance diagram for UACS::Cargo:



Classes

- struct [CargoInfo](#)

The cargo information struct. This is the information the cargo must pass to UACS. On the other hand, API::CargoInfo is used to pass cargo information from UACS to vessels.

Public Member Functions

- **Cargo** (OBJHANDLE hVessel, int fModel=1)
- virtual const [CargoInfo](#) * **clbkGetCargoInfo** ()=0
The cargo information callback. It must be implemented by cargoes.
- std::string_view **GetUACSVersion** ()
Gets UACS version.
- virtual void **clbkCargoGrappled** ()
Optional callback: Called when the cargo is grappled.
- virtual void **clbkCargoReleased** ()
Optional callback: Called when the cargo is released.
- virtual bool **clbkPackCargo** ()
For packable cargoes only: Called when the cargo is packed. Implement the packing logic here.
- virtual bool **clbkUnpackCargo** ()
For unpackable cargoes only: Called when the cargo is unpacked. Implement the unpacking logic here.
- virtual double **clbkDrainResource** (double mass)
For resource cargoes only: Called when the cargo resource is drained. Implement the drainage logic here.

4.5.1 Detailed Description

UACS cargo API.

4.5.2 Member Function Documentation

4.5.2.1 `clbkDrainResource()`

```
double UACS::Cargo::clbkDrainResource (
    double mass) [virtual]
```

For resource cargoes only: Called when the cargo resource is drained. Implement the drainage logic here.

Returns

The drained mass, or 0 if the drainage failed.

4.5.2.2 `clbkGetCargoInfo()`

```
virtual const CargoInfo * UACS::Cargo::clbkGetCargoInfo () [pure virtual]
```

The cargo information callback. It must be implemented by cargoes.

Returns

A const pointer to the [CargoInfo](#) struct. The struct must live until the cargo is destroyed. Don't pass the address of a temporary variable.

4.5.2.3 `clbkPackCargo()`

```
bool UACS::Cargo::clbkPackCargo () [virtual]
```

For packable cargoes only: Called when the cargo is packed. Implement the packing logic here.

Returns

True if the packing is successful, false if not.

4.5.2.4 `clbkUnpackCargo()`

```
bool UACS::Cargo::clbkUnpackCargo () [virtual]
```

For unpackable cargoes only: Called when the cargo is unpacked. Implement the unpacking logic here.

Returns

True if the unpacking is successful, false if not.

4.5.2.5 GetUACSVersion()

```
std::string_view UACS::Cargo::GetUACSVersion ()
```

Gets UACS version.

Returns

UACS version.

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

- [API/Cargo.h](#)
- [API/Cargo.cpp](#)

4.6 UACS::Cargo::CargoInfo Struct Reference

The cargo information struct. This is the information the cargo must pass to UACS. On the other hand, [API::CargoInfo](#) is used to pass cargo information from UACS to vessels.

```
#include <Cargo.h>
```

Public Attributes

- **ATTACHMENTHANDLE hAttach**
The attachment handle used to grapple the cargo.
- **CargoType type**
- **bool unpackOnly** { false }
If the cargo is unpackable only, and cannot be packed again. Applicable only if type is UNPACKABLE.
- **bool unpacked** { false }
The cargo unpacking flag.
- **bool breathable** { false }
The cargo breathability flag.
- **VECTOR3 frontPos** {}
The cargo front end position in vessel-relative coordinates.
- **std::optional< VECTOR3 > rightPos** {}
- **std::optional< VECTOR3 > leftPos** {}
- **std::optional< std::string > resource** {}
The cargo resource. If the cargo isn't a resource, it should be a nullopt.

4.6.1 Detailed Description

The cargo information struct. This is the information the cargo must pass to UACS. On the other hand, [API::CargoInfo](#) is used to pass cargo information from UACS to vessels.

4.6.2 Member Data Documentation

4.6.2.1 breathable

```
bool UACS::Cargo::CargoInfo::breathable { false }
```

The cargo breathability flag.

Note

If the cargo is breathable, this flag should be true even if the cargo is packed. UACS automatically considers the cargo unbreathable if it's packed.

4.6.2.2 frontPos

```
VECTOR3 UACS::Cargo::CargoInfo::frontPos { }
```

The cargo front end position in vessel-relative coordinates.

If rightPos and leftPos are passed, all three values are used to properly orientate the cargo on sloped ground. Otherwise, at least frontPos Y value must be set to the cargo height (see UACS developer manual).

4.6.2.3 resource

```
std::optional<std::string> UACS::Cargo::CargoInfo::resource { }
```

The cargo resource. If the cargo isn't a resource, it should be a nullopt.

Note

Use the standard resource names (see UACS developer manual).

4.6.2.4 unpacked

```
bool UACS::Cargo::CargoInfo::unpacked { false }
```

The cargo unpacking flag.

Note

The cargo is responsible for setting the flag correctly (e.g. loading/saving it to scenario, or when `clbkPackCargo` or `clbkUnpackCargo` is called, etc.).

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

- [API/Cargo.h](#)

4.7 UACS::CargoInfo Struct Reference

Public Attributes

- OBJHANDLE **handle**
- VECTOR3 **attachPos**
The attachment point position in cargo-relative coordinates.
- bool **attached**
If the cargo is attached to another vessel.
- CargoType **type**
- bool **unpackOnly**
If the cargo is unpackable only and can't be packed again. Applicable only if type is UNPACKABLE.
- bool **unpacked**
- bool **breathable**
While the cargo is breathable only if it's unpacked, this flag is true even if the cargo is packed.
- std::optional< std::string > **resource**

4.7.1 Member Data Documentation

4.7.1.1 breathable

```
bool UACS::CargoInfo::breathable
```

While the cargo is breathable only if it's unpacked, this flag is true even if the cargo is packed.

The cargo is considered breathable in its current state if both unpacked and breathable flags are true.

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

- API/[Module.h](#)

4.8 UACS::GroundInfo Struct Reference

Public Attributes

- std::optional< VECTOR3 > **pos**
The ground initial release position in vessel-relative coordinates. If nullopt is passed, it's set to airlock/slot position.
- std::optional< VECTOR3 > **colDir**
The ground column direction in vessel-relative coordinates. If nullopt is passed, it is calculated based on pos (see UACS developer manual).
- std::optional< VECTOR3 > **rowDir**
The ground row direction in vessel-relative coordinates. If nullopt is passed, it is calculated based on pos (see UACS developer manual).
- size_t **colCount** { 3 }
- size_t **rowCount** { 3 }
- double **colSpace** { 1.5 }
The space between each column in meters.
- double **rowSpace** { 1.5 }
The space between each row in meters.

4.8.1 Member Data Documentation

4.8.1.1 colDir

```
std::optional<VECTOR3> UACS::GroundInfo::colDir
```

The ground column direction in vessel-relative coordinates. If nullopt is passed, it is calculated based on pos (see UACS developer manual).

Note

It must be perpendicular to rowDir and normalised. Y value is ignored, as it's determined by the surface elevation.

4.8.1.2 pos

```
std::optional<VECTOR3> UACS::GroundInfo::pos
```

The ground initial release position in vessel-relative coordinates. If nullopt is passed, it's set to airlock/slot position.

Note

Y value is ignored, as it's determined by the surface elevation.

4.8.1.3 rowDir

```
std::optional<VECTOR3> UACS::GroundInfo::rowDir
```

The ground row direction in vessel-relative coordinates. If nullopt is passed, it is calculated based on pos (see UACS developer manual).

Note

It must be perpendicular to colDir and normalised. Y value is ignored, as it's determined by the surface elevation.

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

- API/Common.h

4.9 UACS::Module Class Reference

UACS module API.

```
#include <Module.h>
```

Public Member Functions

- [Module](#) (VESSEL *pVessel, [VslAstrInfo](#) *pVslAstrInfo, [VslCargoInfo](#) *pVslCargoInfo)
Creates an instance from the module API. It can be called anywhere.
- `std::string_view` [GetUACSVersion](#) ()
Gets UACS version. It can be used to find out if UACS is installed.
- `bool` [ParseScenarioLine](#) (char *line)
Parses the scenario file to load UACS information. It must be called in the vessel `clbkLoadStateEx` method.
- `void` [clbkPostCreation](#) ()
Finishes UACS initialization. It must be called once from the vessel `clbkPostCreation` method, after defining all airlocks, stations, and slots.
- `void` [clbkSaveState](#) (FILEHANDLE scn)
Saves UACS information to the scenario file. It must be called in the vessel `clbkSaveState` method.
- `size_t` [GetScnAstrCount](#) ()
Gets the astronaut count in the scenario.
- `std::pair< OBJHANDLE, const AstrInfo * >` [GetAstrInfoByIndex](#) (size_t astrIdx)
Gets an astronaut information by the astronaut index.
- `const AstrInfo *` [GetAstrInfoByHandle](#) (OBJHANDLE hAstr)
Gets an astronaut information by the astronaut handle.
- `const VslAstrInfo *` [GetVslAstrInfo](#) (OBJHANDLE hVessel)
Gets a vessel astronaut information by the vessel handle.
- `bool` [SetAstrInfoByIndex](#) (size_t astrIdx, const [AstrInfo](#) &astrInfo)
Sets an astronaut information by the astronaut index.
- `bool` [SetAstrInfoByHandle](#) (OBJHANDLE hAstr, const [AstrInfo](#) &astrInfo)
Sets an astronaut information by the astronaut handle.
- `void` [DrawAstrInfo](#) (const [AstrInfo](#) &astrInfo, oapi::Sketchpad *skp, int x, int &y, int lineSpacing)
Draws the passed astronaut name, role, mass, fuel and oxygen level, and life flag, each in a line.
- `size_t` [GetAvailAstrCount](#) ()
Gets the available astronaut count.
- `std::string_view` [GetAvailAstrName](#) (size_t availIdx)
Gets the name of an available astronaut, which is the filename of the astronaut config file.
- `double` [GetTotalAstrMass](#) ()
Gets the mass of onboard astronauts.
- `IngressResult` [AddAstronaut](#) (size_t availIdx, std::optional< size_t > stationIdx={}, std::optional< [AstrInfo](#) > astrInfo={})
Adds an astronaut with the passed information to the passed station.
- `TransferResult` [TransferAstronaut](#) (std::optional< size_t > stationIdx={}, std::optional< size_t > airlockIdx={}, std::optional< size_t > tgtStationIdx={})
Transfers the astronaut in the passed station to the vessel docked to the docking port associated with the passed airlock.
- `EgressResult` [EgressAstronaut](#) (std::optional< size_t > stationIdx={}, std::optional< size_t > airlockIdx={})
Egresses the astronaut in the passed station via the passed airlock.
- `size_t` [GetScnCargoCount](#) ()
Gets cargo count in the scenario.
- `CargoInfo` [GetCargoInfoByIndex](#) (size_t cargIdx)
Gets a cargo information by the cargo index.
- `std::optional< CargoInfo >` [GetCargoInfoByHandle](#) (OBJHANDLE hCargo)
Gets a cargo information by the cargo handle.
- `std::optional< std::vector< std::string > >` [GetStationResources](#) (OBJHANDLE hStation)
Gets a station resources by the station handle. The method is expensive, so the result should be cached.
- `void` [DrawCargoInfo](#) ([CargoInfo](#) cargoInfo, oapi::Sketchpad *skp, int x, int &y, int lineSpacing)
Draws the passed cargo name, mass, type, breathability, and resource if applicable, each in a line.

- `size_t GetAvailCargoCount ()`
Gets the available cargo count.
- `std::string_view GetAvailCargoName (size_t availIdx)`
Gets the name of an available cargo, which is the filename of the cargo config file.
- `double GetTotalCargoMass ()`
Gets the mass of all grappled cargoes.
- `GrappleResult AddCargo (size_t availIdx, std::optional< size_t > slotIdx={})`
Adds the passed available cargo to the passed slot.
- `ReleaseResult DeleteCargo (std::optional< size_t > slotIdx={})`
Deletes the cargo in the passed slot.
- `GrappleResult GrappleCargo (OBJHANDLE hCargo=nullptr, std::optional< size_t > slotIdx={})`
Grapples the passed cargo into the passed slot.
- `ReleaseResult ReleaseCargo (std::optional< size_t > slotIdx={})`
Releases the cargo in the passed slot.
- `PackResult PackCargo (OBJHANDLE hCargo=nullptr)`
Packs the passed cargo.
- `PackResult UnpackCargo (OBJHANDLE hCargo=nullptr)`
Unpacks the passed cargo.
- `std::pair< DrainResult, double > DrainGrappledResource (std::string_view resource, double mass, std::optional< size_t > slotIdx={})`
Drains the passed resource from the cargo in the passed slot.
- `std::pair< DrainResult, double > DrainScenarioResource (std::string_view resource, double mass, OBJHANDLE hCargo=nullptr)`
Drains the passed resource from the passed cargo.
- `std::pair< DrainResult, double > DrainStationResource (std::string_view resource, double mass, OBJHANDLE hStation=nullptr)`
Drains the passed resource from the passed station.

4.9.1 Detailed Description

UACS module API.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Module()

```
UACS::Module::Module (
    VESSEL * pVessel,
    VslAstrInfo * pVslAstrInfo,
    VslCargoInfo * pVslCargoInfo)
```

Creates an instance from the module API. It can be called anywhere.

Parameters

| | |
|----------------------|---|
| <i>pVessel</i> | A pointer to the vessel class. If nullptr is passed, astronaut station and airlock and cargo slot specific methods can't be called. pVslAstrInfo and pVslCargoInfo must be set nullptr as well. |
| <i>pVslAstrInfo</i> | A pointer to the vessel astronaut information. If nullptr is passed, astronaut station and airlock specific methods can't be called. |
| <i>pVslCargoInfo</i> | A pointer to vessel cargo information. If nullptr is passed, cargo slot specific methods can't be called. |

Note

All pointers must live until the module API instance is destroyed. Don't pass the address of temporary variables.

Returns

An instance of the module API even if UACS isn't installed. To find out whether UACS is installed or not, use `GetUACSVersion`.

4.9.3 Member Function Documentation**4.9.3.1 AddAstronaut()**

```
IngressResult UACS::Module::AddAstronaut (
    size_t availIdx,
    std::optional< size_t > stationIdx = {},
    std::optional< AstrInfo > astrInfo = {})
```

Adds an astronaut with the passed information to the passed station.

Parameters

| | |
|-------------------|---|
| <i>availIdx</i> | The available astronaut index. It must be less than <code>GetAvailAstrCount</code> . |
| <i>stationIdx</i> | The station index. If <code>nullopt</code> is passed, the first empty station is used. |
| <i>astrInfo</i> | The astronaut information. The class name is set by UACS. If <code>nullopt</code> is passed, the astronaut default information is used. |

Returns

The addition result as the `IngressResult` enum.

4.9.3.2 AddCargo()

```
GrappleResult UACS::Module::AddCargo (
    size_t availIdx,
    std::optional< size_t > slotIdx = {})
```

Adds the passed available cargo to the passed slot.

Parameters

| | |
|-----------------|---|
| <i>availIdx</i> | The available cargo index. It must be less than <code>GetAvailCargoCount</code> . |
| <i>slotIdx</i> | The slot index. If <code>nullopt</code> is passed, the first empty slot is used. |

Returns

The addition result as the `GrappleResult` enum.

4.9.3.3 clbkSaveState()

```
void UACS::Module::clbkSaveState (
    FILEHANDLE scn)
```

Saves UACS information to the scenario file. It must be called in the vessel `clbkSaveState` method.

Parameters

| | |
|------------|--------------------|
| <i>scn</i> | The scenario file. |
|------------|--------------------|

4.9.3.4 DeleteCargo()

```
ReleaseResult UACS::Module::DeleteCargo (
    std::optional< size_t > slotIdx = {})
```

Deletes the cargo in the passed slot.

Parameters

| | |
|----------------|--|
| <i>slotIdx</i> | The slot index. If nullopt is passed, the first occupied slot is used. |
|----------------|--|

Returns

The delete result as the ReleaseResult enum.

4.9.3.5 DrainGrappledResource()

```
std::pair< DrainResult, double > UACS::Module::DrainGrappledResource (
    std::string_view resource,
    double mass,
    std::optional< size_t > slotIdx = {})
```

Drains the passed resource from the cargo in the passed slot.

Parameters

| | |
|-----------------|---|
| <i>resource</i> | The resource name. Use standard resource names (see UACS developer manual). |
| <i>mass</i> | The requested resource mass in kilograms. |
| <i>slotIdx</i> | The slot index. If nullopt is passed, the first suitable cargo is used. |

Returns

A pair of the drainage result and drained mass. The drained mass is 0 if no resource was drained.

4.9.3.6 DrainScenarioResource()

```
std::pair< DrainResult, double > UACS::Module::DrainScenarioResource (
    std::string_view resource,
    double mass,
    OBJHANDLE hCargo = nullptr)
```

Drains the passed resource from the passed cargo.

Parameters

| | |
|-----------------|--|
| <i>resource</i> | The resource name. Use standard resource names (see UACS developer manual). |
| <i>mass</i> | The requested resource mass in kilograms. |
| <i>hCargo</i> | The cargo vessel handle. If nullptr is passed, the nearest suitable cargo in the drainage range is used. |

Returns

A pair of the drainage result and drained mass. The drained mass is 0 if no resource was drained.

4.9.3.7 DrainStationResource()

```
std::pair< DrainResult, double > UACS::Module::DrainStationResource (
    std::string_view resource,
    double mass,
    OBJHANDLE hStation = nullptr)
```

Drains the passed resource from the passed station.

Parameters

| | |
|-----------------|--|
| <i>resource</i> | The resource name. Use standard resource names (see UACS developer manual). |
| <i>mass</i> | The requested resource mass in kilograms. |
| <i>hStation</i> | The station vessel handle. If nullptr is passed, the nearest suitable station in the drainage range is used. |

Returns

A pair of the drainage result and drained mass. The drained mass is 0 if no resource was drained.

4.9.3.8 DrawAstrInfo()

```
void UACS::Module::DrawAstrInfo (
    const AstrInfo & astrInfo,
    oapi::Sketchpad * skp,
    int x,
    int & y,
    int lineSpacing)
```

Draws the passed astronaut name, role, mass, fuel and oxygen level, and life flag, each in a line.

Parameters

| | |
|--------------------|---|
| <i>astrInfo</i> | The astronaut information. |
| <i>skp</i> | The HUD sketchpad. |
| <i>x</i> | The text X coordinate. |
| <i>y</i> | A reference to the text initial Y coordinate. After the information is drawn, it will be the Y coordinate of the last line of text. |
| <i>lineSpacing</i> | The vertical space between each line. |

4.9.3.9 DrawCargoInfo()

```
void UACS::Module::DrawCargoInfo (
    CargoInfo cargoInfo,
    oapi::Sketchpad * skp,
    int x,
    int & y,
    int lineSpacing)
```

Draws the passed cargo name, mass, type, breathability, and resource if applicable, each in a line.

Parameters

| | |
|--------------------|---|
| <i>cargoInfo</i> | The cargo information. |
| <i>skp</i> | The HUD sketchpad. |
| <i>x</i> | The text X coordinate. |
| <i>y</i> | A reference to the text initial Y coordinate. After the information is drawn, it will be the Y coordinate of the last line of text. |
| <i>lineSpacing</i> | The vertical space between each line. |

4.9.3.10 EgressAstronaut()

```
EgressResult UACS::Module::EgressAstronaut (
    std::optional< size_t > stationIdx = {},
    std::optional< size_t > airlockIdx = {})
```

Egresses the astronaut in the passed station via the passed airlock.

Parameters

| | |
|-------------------|------------------------------|
| <i>stationIdx</i> | The astronaut station index. |
| <i>airlockIdx</i> | The airlock index. |

Returns

The egress result.

4.9.3.11 GetAstrInfoByHandle()

```
const AstrInfo * UACS::Module::GetAstrInfoByHandle (
    OBJHANDLE hAstr)
```

Gets an astronaut information by the astronaut handle.

Parameters

| | |
|--------------|------------------------------|
| <i>hAstr</i> | The astronaut vessel handle. |
|--------------|------------------------------|

Returns

A pointer to the astronaut [AstrInfo](#) struct, or nullptr if hAstr is invalid or not an astronaut.

4.9.3.12 GetAstrInfoByIndex()

```
std::pair< OBJHANDLE, const AstrInfo * > UACS::Module::GetAstrInfoByIndex (
    size_t astrIdx)
```

Gets an astronaut information by the astronaut index.

Parameters

| | |
|----------------|--|
| <i>astrIdx</i> | The astronaut index. It must be less than GetScnAstrCount. |
|----------------|--|

Returns

A pair of the astronaut vessel handle and a pointer to the astronaut [AstrInfo](#) struct.

4.9.3.13 GetAvailAstrCount()

```
size_t UACS::Module::GetAvailAstrCount ()
```

Gets the available astronaut count.

It's the count of astronauts that can be added to the scenario, which is the config file count in 'Config\Vessels\↵UACS\Astronauts'.

Returns

The available cargo count.

4.9.3.14 GetAvailAstrName()

```
std::string_view UACS::Module::GetAvailAstrName (
    size_t availIdx)
```

Gets the name of an available astronaut, which is the filename of the astronaut config file.

Parameters

| | |
|-----------------|--|
| <i>availIdx</i> | The available astronaut index. It must be less than GetAvailAstrCount. |
|-----------------|--|

Returns

The available astronaut name.

4.9.3.15 GetAvailCargoCount()

```
size_t UACS::Module::GetAvailCargoCount ()
```

Gets the available cargo count.

It's the count of cargoes that can be added to the scenario, which is the config file count in 'Config\Vessels\UACS\↵Cargoes'.

Returns

The available cargo count.

4.9.3.16 GetAvailCargoName()

```
std::string_view UACS::Module::GetAvailCargoName (
    size_t availIdx)
```

Gets the name of an available cargo, which is the filename of the cargo config file.

Parameters

| | |
|-----------------|---|
| <i>availIdx</i> | The available cargo index. It must be less than GetAvailCargoCount. |
|-----------------|---|

Returns

The available cargo name.

4.9.3.17 GetCargoInfoByHandle()

```
std::optional< CargoInfo > UACS::Module::GetCargoInfoByHandle (
    OBJHANDLE hCargo)
```

Gets a cargo information by the cargo handle.

Parameters

| | |
|---------------|--------------------------|
| <i>hCargo</i> | The cargo vessel handle. |
|---------------|--------------------------|

Returns

The cargo information, or nullopt is hCargo is invalid or not a cargo.

4.9.3.18 GetCargoInfoByIndex()

```
CargoInfo UACS::Module::GetCargoInfoByIndex (
    size_t cargoIdx)
```

Gets a cargo information by the cargo index.

Parameters

| | |
|-----------------|---|
| <i>cargoIdx</i> | The cargo index. It must be less than GetScnCargoCount. |
|-----------------|---|

Returns

The cargo information.

4.9.3.19 GetScnAstrCount()

```
size_t UACS::Module::GetScnAstrCount ()
```

Gets the astronaut count in the scenario.

Returns

The astronaut count in the scenario.

4.9.3.20 GetScnCargoCount()

```
size_t UACS::Module::GetScnCargoCount ()
```

Gets cargo count in the scenario.

Returns

The cargo count in the scenario.

4.9.3.21 GetStationResources()

```
std::optional< std::vector< std::string > > UACS::Module::GetStationResources (
    OBJHANDLE hStation)
```

Gets a station resources by the station handle. The method is expensive, so the result should be cached.

Parameters

| | |
|-----------------|----------------------------|
| <i>hStation</i> | The station vessel handle. |
|-----------------|----------------------------|

Returns

The station resources, or nullopt is hCargo is invalid or not a cargo. The vector is empty if the station supports all resources.

4.9.3.22 GetTotalAstrMass()

```
double UACS::Module::GetTotalAstrMass ()
```

Gets the mass of onboard astronauts.

Returns

The mass of onboard astronauts.

4.9.3.23 GetTotalCargoMass()

```
double UACS::Module::GetTotalCargoMass ()
```

Gets the mass of all grappled cargoes.

Returns

The mass of all grappled cargoes.

4.9.3.24 GetUACSVersion()

```
std::string_view UACS::Module::GetUACSVersion ()
```

Gets UACS version. It can be used to find out if UACS is installed.

Returns

UACS version if UACS is installed, or an empty string view if not.

4.9.3.25 GetVslAstrInfo()

```
const VslAstrInfo * UACS::Module::GetVslAstrInfo (  
    OBJHANDLE hVessel)
```

Gets a vessel astronaut information by the vessel handle.

Parameters

| | |
|----------------|--------------------|
| <i>hVessel</i> | The vessel handle. |
|----------------|--------------------|

Returns

A pointer to the vessel [VslAstrInfo](#) struct, or nullptr if hVessel is invalid or not an astronaut.

4.9.3.26 GrappleCargo()

```
GrappleResult UACS::Module::GrappleCargo (  
    OBJHANDLE hCargo = nullptr,  
    std::optional< size_t > slotIdx = {})
```

Grapples the passed cargo into the passed slot.

Parameters

| | |
|----------------|--|
| <i>hCargo</i> | The cargo vessel handle. If nullptr is passed, the nearest cargo in the grapple range is used. |
| <i>slotIdx</i> | The slot index. If nullopt is passed, the first empty slot is used. |

Returns

The grapple result.

4.9.3.27 PackCargo()

```
PackResult UACS::Module::PackCargo (  
    OBJHANDLE hCargo = nullptr)
```

Packs the passed cargo.

Parameters

| | |
|---------------|---|
| <i>hCargo</i> | The cargo vessel handle. If nullptr is passed, the nearest packable cargo in the packing range is used. |
|---------------|---|

Returns

The packing result.

4.9.3.28 ParseScenarioLine()

```
bool UACS::Module::ParseScenarioLine (  
    char * line)
```

Parses the scenario file to load UACS information. It must be called in the vessel `clbkLoadStateEx` method.

Parameters

| | |
|-------------|---|
| <i>line</i> | The scenario line from the vessel clbkLoadStateEx method. |
|-------------|---|

Returns

True if UACS information was loaded, false if not.

4.9.3.29 ReleaseCargo()

```
ReleaseResult UACS::Module::ReleaseCargo (
    std::optional< size_t > slotIdx = {})
```

Releases the cargo in the passed slot.

Parameters

| | |
|----------------|--|
| <i>slotIdx</i> | The slot index. If nullopt is passed, the first occupied slot is used. |
|----------------|--|

Returns

The release result.

4.9.3.30 SetAstrInfoByHandle()

```
bool UACS::Module::SetAstrInfoByHandle (
    OBJHANDLE hAstr,
    const AstrInfo & astrInfo)
```

Sets an astronaut information by the astronaut handle.

Parameters

| | |
|-----------------|---|
| <i>astrIdx</i> | The astronaut index. It must be less than GetScnAstrCount. |
| <i>astrInfo</i> | The astronaut information. Avoid making an instance of the AstrInfo struct. Rather, use GetAstrInfoByHandle and change the information as required. |

Returns

True if the astronaut information is set, false if not.

4.9.3.31 SetAstrInfoByIndex()

```
bool UACS::Module::SetAstrInfoByIndex (
    size_t astrIdx,
    const AstrInfo & astrInfo)
```

Sets an astronaut information by the astronaut index.

Parameters

| | |
|-----------------|--|
| <i>astrIdx</i> | The astronaut index. It must be less than GetScnAstrCount. |
| <i>astrInfo</i> | The astronaut information. Avoid making an instance of the AstrInfo struct. Rather, use GetAstrInfoByIndex and change the information as required. |

Returns

True if the astronaut information is set, false if not.

4.9.3.32 TransferAstronaut()

```
TransferResult UACS::Module::TransferAstronaut (
    std::optional< size_t > stationIdx = {},
    std::optional< size_t > airlockIdx = {},
    std::optional< size_t > tgtStationIdx = {})
```

Transfers the astronaut in the passed station to the vessel docked to the docking port associated with the passed airlock.

Parameters

| | |
|----------------------|--|
| <i>stationIdx</i> | The astronaut station index. |
| <i>airlockIdx</i> | The airlock index. A vessel with UACS astronaut implementation must be docked to the docking port associated with the airlock. |
| <i>tgtStationIdx</i> | The target vessel station index to transfer the astronaut into it. If nullopt is passed, the first empty station is used. |

Returns

The transfer result.

4.9.3.33 UnpackCargo()

```
PackResult UACS::Module::UnpackCargo (
    OBJHANDLE hCargo = nullptr)
```

Unpacks the passed cargo.

Parameters

| | |
|---------------|---|
| <i>hCargo</i> | The cargo vessel handle. If nullptr is passed, the nearest unpackable cargo in the packing range is used. |
|---------------|---|

Returns

The unpacking result as the PackResult enum.

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

- [API/Module.h](#)
- [API/Module.cpp](#)

4.10 UACS::NearestAction Struct Reference

Public Attributes

- OBJHANDLE **hVessel**
- size_t **actionIdx**
- [ActionInfo](#) **actionInfo**

The action area position is converted from vessel-relative coordinates to astronaut-relative coordinates.

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

- API/[Astronaut.h](#)

4.11 UACS::NearestAirlock Struct Reference

Public Attributes

- OBJHANDLE **hVessel**
- size_t **airlockIdx**
- [AirlockInfo](#) **airlockInfo**

The airlock position is converted from vessel-relative coordinates to astronaut-relative coordinates.

- size_t **stationIdx**
The first empty station index.

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

- API/[Astronaut.h](#)

4.12 UACS::SlotInfo Struct Reference

Public Attributes

- ATTACHMENTHANDLE **hAttach**
- VECTOR3 [holdDir](#)
The slot cargo holding direction. It's used to position the grappled cargo properly (see UACS developer manual).
- bool **open** { true }
- double **relVel** { 0 }
The cargo release velocity (if released in space) in meters per second.
- [GroundInfo](#) **gndInfo** {}
The slot ground release information. If astronaut mode is on, it has no effect.
- std::optional< [CargoInfo](#) > **cargoInfo** {}

4.12.1 Member Data Documentation

4.12.1.1 holdDir

VECTOR3 UACS::SlotInfo::holdDir

The slot cargo holding direction. It's used to position the grappled cargo properly (see UACS developer manual).

If the cargo is held from below, the Y value is -1. From above, Y value is 1. From right, X value is 1. From left, X value is -1. From front, Z value is 1. From rear, Z value is -1. All other values is set to 0.

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

- [API/Module.h](#)

4.13 UACS::StationInfo Struct Reference

Public Attributes

- std::string **name**
- std::optional< [AstrInfo](#) > **astrInfo**

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

- [API/Common.h](#)

4.14 UACS::VslAstrInfo Struct Reference

Public Attributes

- std::vector< [StationInfo](#) > **stations**
- std::vector< [AirlockInfo](#) > **airlocks**
- std::vector< [ActionInfo](#) > **actionAreas**

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

- [API/Common.h](#)

4.15 UACS::VslCargoInfo Struct Reference

Public Attributes

- `std::vector< SlotInfo > slots`
- `bool astrMode { false }`
The astronaut mode flag.
- `double grappleRange { 50 }`
The cargo grapple range in meters. It's the max distance between the slot attachment point position and the cargo attachment point position.
- `double packRange { 50 }`
The search range for packable and unpackable cargoes in meters. It's the max distance between the vessel radius (size) and the cargo radius.
- `double drainRange { 100 }`
The search range for draining resources in meters. It's the max distance between the vessel radius (size) and the cargo/station radius.
- `std::optional< double > maxCargoMass`
The maximum single cargo mass in kilograms.
- `std::optional< double > maxTotalCargoMass`
The maximum total cargo mass in kilograms.

4.15.1 Member Data Documentation

4.15.1.1 [astrMode](#)

```
bool UACS::VslCargoInfo::astrMode { false }
```

The astronaut mode flag.

If on:

1. Unpacked cargoes can be grappled.
2. On ground, cargoes are released in a single position (slot attachment point position or ground information position), not in a table. All of the slot ground information except position is ignored. If a cargo exists in that position, the grappled cargo can't be released.
3. On ground, cargo heading once released is the same as its heading when grappled. If astronaut mode is off, the heading is set to the vessel heading.

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

- [API/Module.h](#)

Chapter 5

File Documentation

5.1 API/Astronaut.h File Reference

UACS astronaut API header.

```
#include "Common.h"
#include <VesselAPI.h>
```

Classes

- struct [UACS::NearestAirlock](#)
- struct [UACS::NearestAction](#)
- class [UACS::Astronaut](#)

UACS astronaut API.

5.1.1 Detailed Description

UACS astronaut API header.

5.2 Astronaut.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "Common.h"
00003 #include <VesselAPI.h>
00004
00010 namespace UACS
00011 {
00012     namespace Core { class Astronaut; }
00013
00014     struct NearestAirlock
00015     {
00016         OBJHANDLE hVessel;
00017         size_t airlockIdx;
00018
00020         AirlockInfo airlockInfo;
00021
00023         size_t stationIdx;
00024     };
```

```

00025
00026     struct NearestAction
00027     {
00028         OBJHANDLE hVessel;
00029         size_t actionIdx;
00030
00031         ActionInfo actionInfo;
00032     };
00033
00034     class Astronaut : public VESSEL4
00035     {
00036     public:
00037         Astronaut(OBJHANDLE hVessel, int fModel = 1);
00038         ~Astronaut();
00039
00040         virtual bool clbkSetAstrInfo(const AstrInfo& astrInfo) = 0;
00041
00042         virtual const AstrInfo* clbkGetAstrInfo() = 0;
00043
00044         std::string_view GetUACSVersion();
00045
00046         size_t GetScnAstrCount();
00047
00048         std::pair<OBJHANDLE, const AstrInfo*> GetAstrInfoByIndex(size_t astrIdx);
00049
00050         const AstrInfo* GetAstrInfoByHandle(OBJHANDLE hAstr);
00051
00052         const VslAstrInfo* GetVslAstrInfo(OBJHANDLE hVessel);
00053
00054         std::optional<NearestAirlock> GetNearestAirlock(double range, bool airlockOpen = true, bool
stationEmpty = true);
00055
00056         std::pair<OBJHANDLE, VECTOR3> GetNearestBreathable(double range);
00057
00058         std::optional<NearestAction> GetNearestAction(double range, bool areaEnabled = true);
00059
00060         bool InBreathable(bool checkAtm = true);
00061
00062         IngressResult Ingress(OBJHANDLE hVessel = nullptr, std::optional<size_t> airlockIdx = {},
std::optional<size_t> stationIdx = {});
00063
00064         IngressResult TriggerAction(OBJHANDLE hVessel = nullptr, std::optional<size_t> actionIdx =
{});
00065
00066     private:
00067         HINSTANCE coreDLL;
00068         Core::Astronaut* pCoreAstr{};
00069     };
00070 }

```

5.3 API/Cargo.h File Reference

UACS cargo API header.

```

#include "Common.h"
#include <VesselAPI.h>

```

Classes

- class [UACS::Cargo](#)
UACS cargo API.
- struct [UACS::Cargo::CargoInfo](#)
The cargo information struct. This is the information the cargo must pass to UACS. On the other hand, API::CargoInfo is used to pass cargo information from UACS to vessels.

5.3.1 Detailed Description

UACS cargo API header.

5.4 Cargo.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002 #include "Common.h"
00003 #include <VesselAPI.h>
00004
00010 namespace UACS
00011 {
00012     namespace Core { class Cargo; }
00013
00015     class Cargo : public VESSEL4
00016     {
00017     public:
00022         struct CargoInfo
00023         {
00025             ATTACHMENTHANDLE hAttach;
00026
00027             CargoType type;
00028
00030             bool unpackOnly{ false };
00031
00036             bool unpacked{ false };
00037
00043             bool breathable{ false };
00044
00051             VECTOR3 frontPos{};
00052
00053             std::optional<VECTOR3> rightPos{};
00054
00055             std::optional<VECTOR3> leftPos{};
00056
00061             std::optional<std::string> resource{};
00062         };
00063
00064         Cargo(OBJHANDLE hVessel, int fModel = 1);
00065         virtual ~Cargo();
00066
00071         virtual const CargoInfo* clbkGetCargoInfo() = 0;
00072
00077         std::string_view GetUACSVersion();
00078
00080         virtual void clbkCargoGrappled();
00081
00083         virtual void clbkCargoReleased();
00084
00089         virtual bool clbkPackCargo();
00090
00095         virtual bool clbkUnpackCargo();
00096
00101         virtual double clbkDrainResource(double mass);
00102
00103     private:
00104         HINSTANCE coreDLL;
00105         Core::Cargo* pCoreCargo{};
00106     };
00107 }

```

5.5 Common.h

```

00001 #pragma once
00002 #include <OrbiterAPI.h>
00003 #include <vector>
00004 #include <optional>
00005
00006 namespace UACS
00007 {
00008     constexpr int MSG = 0x55414353;
00009
00010     enum Message
00011     {
00012         ASTR_INGRS,
00013         ASTR_EGRS,
00014         ACTN_TRIG
00015     };
00016
00017     struct GroundInfo
00018     {
00023         std::optional<VECTOR3> pos;
00024
00029         std::optional<VECTOR3> colDir;

```

```

00030
00035     std::optional<VECTOR3> rowDir;
00036
00037     size_t colCount{ 3 };
00038
00039     size_t rowCount{ 3 };
00040
00042     double colSpace{ 1.5 };
00043
00045     double rowSpace{ 1.5 };
00046 };
00047
00048 struct AstrInfo
00049 {
00051     std::string name;
00052
00054     std::string role;
00055
00057     double mass;
00058
00060     double height;
00061
00063     double fuelLvl{ 1 };
00064
00066     double oxyLvl{ 1 };
00067
00069     bool alive{ true };
00070
00072     std::string customData{};
00073
00079     std::string className{};
00080 };
00081
00082 struct StationInfo
00083 {
00084     std::string name;
00085     std::optional<AstrInfo> astrInfo;
00086 };
00087
00088 struct AirlockInfo
00089 {
00090     std::string name;
00091
00096     VECTOR3 pos;
00097
00103     VECTOR3 dir;
00104
00110     VECTOR3 rot;
00111
00116     double range{ 5 };
00117
00118     bool open{ true };
00119
00121     double relVel{ 0 };
00122
00124     GroundInfo gndInfo{};
00125
00127     DOCKHANDLE hDock{};
00128 };
00129
00130 struct ActionInfo
00131 {
00132     std::string name;
00133
00135     VECTOR3 pos;
00136
00138     double range{ 5 };
00139
00140     bool enabled{ true };
00141 };
00142
00143 struct VslAstrInfo
00144 {
00145     std::vector<StationInfo> stations;
00146     std::vector<AirlockInfo> airlocks;
00147     std::vector<ActionInfo> actionAreas;
00148 };
00149
00150 enum IngressResult
00151 {
00152     INGRS_SUCCED,
00153
00159     INGRS_NOT_IN_RNG,
00160
00162     INGRS_ARLCK_UNDEF,
00163
00165     INGRS_ARLCK_CLSD,

```

```

00166
00168         INGRS_ARLCK_DCKD,
00169
00171         INGRS_STN_UNDEF,
00172
00174         INGRS_STN_OCCP,
00175
00177         INGRS_VSL_REJC,
00178
00179         INGRS_FAIL
00180     };
00181
00182     enum CargoType { STATIC, UNPACKABLE };
00183 }

```

5.6 API/Module.h File Reference

UACS module API header.

```
#include "Common.h"
```

Classes

- struct [UACS::CargoInfo](#)
- struct [UACS::SlotInfo](#)
- struct [UACS::VslCargoInfo](#)
- class [UACS::Module](#)

UACS module API.

Enumerations

- enum [UACS::TransferResult](#) {
TRNS_SUCCEEDED, **TRNS_STN_EMPTY**, **TRNS_ARLCK_CLSD**, [UACS::TRNS_DOCK_UNDEF](#),
[UACS::TRNS_DOCK_EMPTY](#), [UACS::TRNS_TGT_ARLCK_UNDEF](#), [UACS::TRNS_TGT_ARLCK_CLSD](#),
[UACS::TRNS_TGT_STN_UNDEF](#),
[UACS::TRNS_TGT_STN_OCCP](#), [UACS::TRNS_VSL_REJC](#), **TRNS_FAIL** }
- enum [UACS::EgressResult](#) {
EGRS_SUCCEEDED, **EGRS_STN_EMPTY**, **EGRS_ARLCK_CLSD**, [UACS::EGRS_ARLCK_DCKD](#),
[UACS::EGRS_NO_EMPTY_POS](#), [UACS::EGRS_INFO_NOT_SET](#), **EGRS_FAIL** }
- enum [UACS::GrappleResult](#) {
GRPL_SUCCEEDED, [UACS::GRPL_SLT_CLSD](#), [UACS::GRPL_SLT_OCCP](#), [UACS::GRPL_NOT_IN_RNG](#),
[UACS::GRPL_CRG_UNPCKD](#), [UACS::GRPL_CRG_ATCHD](#), **GRPL_MASS_EXCD**, [GRPL_TTL_MASS_EXCD](#),
GRPL_FAIL }
- enum [UACS::ReleaseResult](#) {
RLES_SUCCEEDED, [UACS::RLES_SLT_CLSD](#), [UACS::RLES_SLT_EMPTY](#), [UACS::RLES_NO_EMPTY_POS](#),
RLES_FAIL }
- enum [UACS::PackResult](#) {
PACK_SUCCEEDED, [UACS::PACK_NOT_IN_RNG](#), [UACS::PACK_CRG_PCKD](#), [UACS::PACK_CRG_UNPCKD](#),
[UACS::PACK_CRG_NOT_PCKABL](#), [UACS::PACK_CRG_ATCHD](#), **PACK_FAIL** }
- enum [UACS::DrainResult](#) {
DRIN_SUCCEEDED, [UACS::DRIN_SLT_EMPTY](#), [UACS::DRIN_NOT_IN_RNG](#), [UACS::DRIN_VSL_NOT_RES](#),
[UACS::DRIN_RES_NOT_FND](#), [UACS::DRIN_RES_ATCHD](#), **DRIN_FAIL** }

5.6.1 Detailed Description

UACS module API header.

5.6.2 Enumeration Type Documentation

5.6.2.1 DrainResult

enum `UACS::DrainResult`

Enumerator

| | |
|------------------|---|
| DRIN_SLT_EMPTY | The passed slot (or all slots if slotIdx is nullopt) is empty. Returned only by DrainGrappledResource method. |
| DRIN_NOT_IN_RNG | The passed vessel (or all resource cargoes or stations if hCargo or hStation is nullptr) is outside the drainage range. Not returned by DrainGrappledResource method. |
| DRIN_VSL_NOT_RES | The passed cargo or station (or all resource cargoes or stations if hCargo or hStation is nullptr) isn't a resource. |
| DRIN_RES_NOT_FND | The passed cargo or station doesn't contain the passed resource. |
| DRIN_RES_ATCHD | The passed cargo is attached to another vessel. Returned only by DrainScenarioResource. |

5.6.2.2 EgressResult

enum `UACS::EgressResult`

Enumerator

| | |
|-------------------|---|
| EGRS_ARLCK_DCKD | A vessel is docked to the passed airlock (or all airlocks if airlockIdx is nullopt) docking port. |
| EGRS_NO_EMPTY_POS | No empty position to egress the astronaut on the ground. |
| EGRS_INFO_NOT_SET | The astronaut was egressed but its information couldn't be set. |

5.6.2.3 GrappleResult

enum `UACS::GrappleResult`

Enumerator

| | |
|-----------------|---|
| GRPL_SLT_CLSD | The passed slot (or all slots if slotIdx is nullopt) is closed. |
| GRPL_SLT_OCCP | The passed slot (or all slots if slotIdx is nullopt) is occupied. |
| GRPL_NOT_IN_RNG | The passed cargo (or all grappable cargoes if hCargo is nullptr) is outside the grapple range. |
| GRPL_CRG_UNPCKD | The passed cargo can't be grappled because it is unpacked. Not applicable if grappleUnpacked is true. |
| GRPL_CRG_ATCHD | The passed cargo is attached to another vessel. |

5.6.2.4 PackResult

enum `UACS::PackResult`

Enumerator

| | |
|---------------------|--|
| PACK_NOT_IN_RNG | The passed cargo (or all packable cargoes if hCargo is nullptr) is outside the packing range. |
| PACK_CRG_PCKD | The passed cargo is already packed. Returned only by PackCargo method. |
| PACK_CRG_UNPCKD | The passed cargo is already unpacked. Returned only by UnpackCargo method. |
| PACK_CRG_NOT_PCKABL | The passed cargo is not packable if PackCargo method is called, or not unpackable if UnpackCargo method is called. |
| PACK_CRG_ATCHD | The passed cargo is attached to another vessel. |

5.6.2.5 ReleaseResult

```
enum UACS::ReleaseResult
```

Enumerator

| | |
|-------------------|---|
| RLES_SLT_CLSD | The passed slot (or all slots if slotIdx is nullopt) is closed. |
| RLES_SLT_EMPTY | The passed slot (or all slots if slotIdx is nullopt) is empty. |
| RLES_NO_EMPTY_POS | No empty position to release the cargo on the ground. |

5.6.2.6 TransferResult

```
enum UACS::TransferResult
```

Enumerator

| | |
|----------------------|---|
| TRNS_DOCK_UNDEF | The passed airlock (or all airlocks if airlockIdx is nullopt) has no docking port. |
| TRNS_DOCK_EMPTY | The passed airlock (or all airlocks if airlockIdx is nullopt) docking port is empty. |
| TRNS_TGT_ARLCK_UNDEF | The target vessel has no airlocks associated with the docking port. |
| TRNS_TGT_ARLCK_CLSD | The target airlock is closed. |
| TRNS_TGT_STN_UNDEF | The passed target station (or all target stations if tgtStationIdx is nullopt) is invalid. |
| TRNS_TGT_STN_OCCP | The passed target station (or all target stations if tgtStationIdx is nullopt) is occupied. |
| TRNS_VSL_REJC | The passed target vessel rejected the astronaut transfer. |

5.7 Module.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "Common.h"
00003
00009 namespace UACS
00010 {
00011     namespace Core { class Module; }
00012
00013     struct CargoInfo
00014     {
```

```

00015         OBJHANDLE handle;
00016
00018         VECTOR3 attachPos;
00019
00021         bool attached;
00022
00023         CargoType type;
00024
00026         bool unpackOnly;
00027
00028         bool unpacked;
00029
00035         bool breathable;
00036
00037         std::optional<std::string> resource;
00038     };
00039
00040     struct SlotInfo
00041     {
00042         ATTACHMENTHANDLE hAttach;
00043
00051         VECTOR3 holdDir;
00052
00053         bool open{ true };
00054
00056         double relVel{ 0 };
00057
00059         GroundInfo gndInfo{};
00060
00061         std::optional<CargoInfo> cargoInfo{};
00062     };
00063
00064     struct VslCargoInfo
00065     {
00066         std::vector<SlotInfo> slots;
00067
00077         bool astrMode{ false };
00078
00080         double grappleRange{ 50 };
00081
00083         double packRange{ 50 };
00084
00086         double drainRange{ 100 };
00087
00089         std::optional<double> maxCargoMass;
00090
00092         std::optional<double> maxTotalCargoMass;
00093     };
00094
00095     enum TransferResult
00096     {
00097         TRNS_SUCCEED,
00098         TRNS_STN_EMPTY,
00099         TRNS_ARLCK_CLSD,
00100
00102         TRNS_DOCK_UNDEF,
00103
00105         TRNS_DOCK_EMPTY,
00106
00108         TRNS_TGT_ARLCK_UNDEF,
00109
00111         TRNS_TGT_ARLCK_CLSD,
00112
00114         TRNS_TGT_STN_UNDEF,
00115
00117         TRNS_TGT_STN_OCCP,
00118
00120         TRNS_VSL_REJC,
00121
00122         TRNS_FAIL
00123     };
00124
00125     enum EgressResult
00126     {
00127         EGRS_SUCCEED,
00128         EGRS_STN_EMPTY,
00129         EGRS_ARLCK_CLSD,
00130
00132         EGRS_ARLCK_DCKD,
00133
00135         EGRS_NO_EMPTY_POS,
00136
00138         EGRS_INFO_NOT_SET,
00139
00140         EGRS_FAIL
00141     };
00142

```

```

00143     enum GrappleResult
00144     {
00145         GRPL_SUCCEEDED,
00146
00148         GRPL_SLT_CLSD,
00149
00151         GRPL_SLT_OCCP,
00152
00154         GRPL_NOT_IN_RNG,
00155
00157         GRPL_CRG_UNPCKD,
00158
00160         GRPL_CRG_ATCHD,
00161
00162         GRPL_MASS_EXCD,
00163         GRPL_TTL_MASS_EXCD,
00164         GRPL_FAIL
00165     };
00166
00167     enum ReleaseResult
00168     {
00169         RLES_SUCCEEDED,
00170
00172         RLES_SLT_CLSD,
00173
00175         RLES_SLT_EMPTY,
00176
00178         RLES_NO_EMPTY_POS,
00179
00180         RLES_FAIL
00181     };
00182
00183     enum PackResult
00184     {
00185         PACK_SUCCEEDED,
00186
00188         PACK_NOT_IN_RNG,
00189
00191         PACK_CRG_PCKD,
00192
00194         PACK_CRG_UNPCKD,
00195
00197         PACK_CRG_NOT_PCKABL,
00198
00200         PACK_CRG_ATCHD,
00201
00202         PACK_FAIL
00203     };
00204
00205     enum DrainResult
00206     {
00207         DRIN_SUCCEEDED,
00208
00210         DRIN_SLT_EMPTY,
00211
00213         DRIN_NOT_IN_RNG,
00214
00216         DRIN_VSL_NOT_RES,
00217
00219         DRIN_RES_NOT_FND,
00220
00222         DRIN_RES_ATCHD,
00223
00224         DRIN_FAIL
00225     };
00226
00228     class Module
00229     {
00230     public:
00240         Module(VESSEL* pVessel, VslAstrInfo* pVslAstrInfo, VslCargoInfo* pVslCargoInfo);
00241         ~Module();
00242
00247         std::string_view GetUACSVersion();
00248
00249         // The 3 methods below must be called if pVessel is defined.
00250
00256         bool ParseScenarioLine(char* line);
00257
00262         void clbkPostCreation();
00263
00268         void clbkSaveState(FILEHANDLE scn);
00269
00270         // Astronaut methods.
00271
00276         size_t GetScnAstrCount();
00277
00283         std::pair<OBJHANDLE, const AstrInfo*> GetAstrInfoByIndex(size_t astrIdx);

```

```

00284
00290     const AstrInfo* GetAstrInfoByHandle(OBJHANDLE hAstr);
00291
00297     const VslAstrInfo* GetVslAstrInfo(OBJHANDLE hVessel);
00298
00305     bool SetAstrInfoByIndex(size_t astrIdx, const AstrInfo& astrInfo);
00306
00313     bool SetAstrInfoByHandle(OBJHANDLE hAstr, const AstrInfo& astrInfo);
00314
00323     void DrawAstrInfo(const AstrInfo& astrInfo, oapi::Sketchpad* skp, int x, int& y, int
lineSpacing);
00324
00331     size_t GetAvailAstrCount();
00332
00338     std::string_view GetAvailAstrName(size_t availIdx);
00339
00340     // Astronaut methods below can only be used only if pVessel and pVslAstrInfo are defined.
00341
00346     double GetTotalAstrMass();
00347
00355     IngressResult AddAstronaut(size_t availIdx, std::optional<size_t> stationIdx = {},
std::optional<AstrInfo> astrInfo = {});
00356
00364     TransferResult TransferAstronaut(std::optional<size_t> stationIdx = {}, std::optional<size_t>
airlockIdx = {}, std::optional<size_t> tgtStationIdx = {});
00365
00372     EgressResult EgressAstronaut(std::optional<size_t> stationIdx = {}, std::optional<size_t>
airlockIdx = {});
00373
00374     // Cargo methods.
00375
00380     size_t GetScnCargoCount();
00381
00387     CargoInfo GetCargoInfoByIndex(size_t cargoIdx);
00388
00394     std::optional<CargoInfo> GetCargoInfoByHandle(OBJHANDLE hCargo);
00395
00401     std::optional<std::vector<std::string> GetStationResources(OBJHANDLE hStation);
00402
00411     void DrawCargoInfo(CargoInfo cargoInfo, oapi::Sketchpad* skp, int x, int& y, int lineSpacing);
00412
00419     size_t GetAvailCargoCount();
00420
00426     std::string_view GetAvailCargoName(size_t availIdx);
00427
00428     // Cargo methods below can only be used only if pVessel and pVslCargoInfo are defined.
00429
00434     double GetTotalCargoMass();
00435
00442     GrappleResult AddCargo(size_t availIdx, std::optional<size_t> slotIdx = {});
00443
00449     ReleaseResult DeleteCargo(std::optional<size_t> slotIdx = {});
00450
00457     GrappleResult GrappleCargo(OBJHANDLE hCargo = nullptr, std::optional<size_t> slotIdx = {});
00458
00464     ReleaseResult ReleaseCargo(std::optional<size_t> slotIdx = {});
00465
00471     PackResult PackCargo(OBJHANDLE hCargo = nullptr);
00472
00478     PackResult UnpackCargo(OBJHANDLE hCargo = nullptr);
00479
00487     std::pair<DrainResult, double> DrainGrappledResource(std::string_view resource, double mass,
std::optional<size_t> slotIdx = {});
00488
00496     std::pair<DrainResult, double> DrainScenarioResource(std::string_view resource, double mass,
OBJHANDLE hCargo = nullptr);
00497
00505     std::pair<DrainResult, double> DrainStationResource(std::string_view resource, double mass,
OBJHANDLE hStation = nullptr);
00506
00507     private:
00508         HINSTANCE coreDLL;
00509         Core::Module* pCoreModule{};
00510     };
00511 }

```

Index

AddAstronaut
 UACS::Module, [22](#)

AddCargo
 UACS::Module, [22](#)

API/Astronaut.h, [37](#)

API/Cargo.h, [38](#), [39](#)

API/Common.h, [39](#)

API/Module.h, [41](#), [43](#)

astrMode
 UACS::VslCargoInfo, [36](#)

breathable
 UACS::Cargo::CargoInfo, [17](#)
 UACS::CargoInfo, [18](#)

className
 UACS::AstrInfo, [9](#)

clbkDrainResource
 UACS::Cargo, [15](#)

clbkGetAstrInfo
 UACS::Astronaut, [10](#)

clbkGetCargoInfo
 UACS::Cargo, [15](#)

clbkPackCargo
 UACS::Cargo, [15](#)

clbkSaveState
 UACS::Module, [22](#)

clbkSetAstrInfo
 UACS::Astronaut, [10](#)

clbkUnpackCargo
 UACS::Cargo, [15](#)

colDir
 UACS::GroundInfo, [19](#)

DeleteCargo
 UACS::Module, [23](#)

dir
 UACS::AirlockInfo, [8](#)

DrainGrappledResource
 UACS::Module, [23](#)

DrainResult
 Module.h, [42](#)

DrainScenarioResource
 UACS::Module, [23](#)

DrainStationResource
 UACS::Module, [24](#)

DrawAstrInfo
 UACS::Module, [24](#)

DrawCargoInfo
 UACS::Module, [24](#)

DRIN_NOT_IN_RNG
 Module.h, [42](#)

DRIN_RES_ATCHD
 Module.h, [42](#)

DRIN_RES_NOT_FND
 Module.h, [42](#)

DRIN_SLT_EMPTY
 Module.h, [42](#)

DRIN_VSL_NOT_RES
 Module.h, [42](#)

EgressAstronaut
 UACS::Module, [25](#)

EgressResult
 Module.h, [42](#)

EGRS_ARLCK_DCKD
 Module.h, [42](#)

EGRS_INFO_NOT_SET
 Module.h, [42](#)

EGRS_NO_EMPTY_POS
 Module.h, [42](#)

frontPos
 UACS::Cargo::CargoInfo, [17](#)

GetAstrInfoByHandle
 UACS::Astronaut, [10](#)
 UACS::Module, [25](#)

GetAstrInfoByIndex
 UACS::Astronaut, [11](#)
 UACS::Module, [25](#)

GetAvailAstrCount
 UACS::Module, [27](#)

GetAvailAstrName
 UACS::Module, [27](#)

GetAvailCargoCount
 UACS::Module, [27](#)

GetAvailCargoName
 UACS::Module, [27](#)

GetCargoInfoByHandle
 UACS::Module, [28](#)

GetCargoInfoByIndex
 UACS::Module, [28](#)

GetNearestAction
 UACS::Astronaut, [11](#)

GetNearestAirlock
 UACS::Astronaut, [11](#)

GetNearestBreathable
 UACS::Astronaut, [12](#)

GetScnAstrCount

- UACS::Astronaut, [12](#)
- UACS::Module, [28](#)
- GetScnCargoCount
 - UACS::Module, [28](#)
- GetStationResources
 - UACS::Module, [29](#)
- GetTotalAstrMass
 - UACS::Module, [29](#)
- GetTotalCargoMass
 - UACS::Module, [29](#)
- GetUACSVersion
 - UACS::Astronaut, [12](#)
 - UACS::Cargo, [15](#)
 - UACS::Module, [29](#)
- GetVslAstrInfo
 - UACS::Astronaut, [12](#)
 - UACS::Module, [30](#)
- GrappleCargo
 - UACS::Module, [31](#)
- GrappleResult
 - Module.h, [42](#)
- GRPL_CRG_ATCHD
 - Module.h, [42](#)
- GRPL_CRG_UNPCKD
 - Module.h, [42](#)
- GRPL_NOT_IN_RNG
 - Module.h, [42](#)
- GRPL_SLT_CLSD
 - Module.h, [42](#)
- GRPL_SLT_OCCP
 - Module.h, [42](#)
- holdDir
 - UACS::SlotInfo, [35](#)
- InBreathable
 - UACS::Astronaut, [13](#)
- Ingress
 - UACS::Astronaut, [13](#)
- Module
 - UACS::Module, [21](#)
- Module.h
 - DrainResult, [42](#)
 - DRIN_NOT_IN_RNG, [42](#)
 - DRIN_RES_ATCHD, [42](#)
 - DRIN_RES_NOT_FND, [42](#)
 - DRIN_SLT_EMPTY, [42](#)
 - DRIN_VSL_NOT_RES, [42](#)
 - EgressResult, [42](#)
 - EGRS_ARLCK_DCKD, [42](#)
 - EGRS_INFO_NOT_SET, [42](#)
 - EGRS_NO_EMPTY_POS, [42](#)
 - GrappleResult, [42](#)
 - GRPL_CRG_ATCHD, [42](#)
 - GRPL_CRG_UNPCKD, [42](#)
 - GRPL_NOT_IN_RNG, [42](#)
 - GRPL_SLT_CLSD, [42](#)
 - GRPL_SLT_OCCP, [42](#)
 - PACK_CRG_ATCHD, [43](#)
 - PACK_CRG_NOT_PCKABL, [43](#)
 - PACK_CRG_PCKD, [43](#)
 - PACK_CRG_UNPCKD, [43](#)
 - PACK_NOT_IN_RNG, [43](#)
 - PackResult, [42](#)
 - ReleaseResult, [43](#)
 - RLES_NO_EMPTY_POS, [43](#)
 - RLES_SLT_CLSD, [43](#)
 - RLES_SLT_EMPTY, [43](#)
 - TransferResult, [43](#)
 - TRNS_DOCK_EMPTY, [43](#)
 - TRNS_DOCK_UNDEF, [43](#)
 - TRNS_TGT_ARLCK_CLSD, [43](#)
 - TRNS_TGT_ARLCK_UNDEF, [43](#)
 - TRNS_TGT_STN_OCCP, [43](#)
 - TRNS_TGT_STN_UNDEF, [43](#)
 - TRNS_VSL_REJC, [43](#)
- PACK_CRG_ATCHD
 - Module.h, [43](#)
- PACK_CRG_NOT_PCKABL
 - Module.h, [43](#)
- PACK_CRG_PCKD
 - Module.h, [43](#)
- PACK_CRG_UNPCKD
 - Module.h, [43](#)
- PACK_NOT_IN_RNG
 - Module.h, [43](#)
- PackCargo
 - UACS::Module, [31](#)
- PackResult
 - Module.h, [42](#)
- ParseScenarioLine
 - UACS::Module, [31](#)
- pos
 - UACS::GroundInfo, [19](#)
- ReleaseCargo
 - UACS::Module, [32](#)
- ReleaseResult
 - Module.h, [43](#)
- resource
 - UACS::Cargo::CargoInfo, [17](#)
- RLES_NO_EMPTY_POS
 - Module.h, [43](#)
- RLES_SLT_CLSD
 - Module.h, [43](#)
- RLES_SLT_EMPTY
 - Module.h, [43](#)
- rot
 - UACS::AirlockInfo, [8](#)
- rowDir
 - UACS::GroundInfo, [19](#)
- SetAstrInfoByHandle
 - UACS::Module, [32](#)
- SetAstrInfoByIndex
 - UACS::Module, [32](#)

- TransferAstronaut
 - UACS::Module, [33](#)
- TransferResult
 - Module.h, [43](#)
- TriggerAction
 - UACS::Astronaut, [13](#)
- TRNS_DOCK_EMPTY
 - Module.h, [43](#)
- TRNS_DOCK_UNDEF
 - Module.h, [43](#)
- TRNS_TGT_ARLCK_CLSD
 - Module.h, [43](#)
- TRNS_TGT_ARLCK_UNDEF
 - Module.h, [43](#)
- TRNS_TGT_STN_OCCP
 - Module.h, [43](#)
- TRNS_TGT_STN_UNDEF
 - Module.h, [43](#)
- TRNS_VSL_REJC
 - Module.h, [43](#)
- UACS::ActionInfo, [7](#)
- UACS::AirlockInfo, [7](#)
 - dir, [8](#)
 - rot, [8](#)
- UACS::AstrInfo, [8](#)
 - className, [9](#)
- UACS::Astronaut, [9](#)
 - clbkGetAstrInfo, [10](#)
 - clbkSetAstrInfo, [10](#)
 - GetAstrInfoByHandle, [10](#)
 - GetAstrInfoByIndex, [11](#)
 - GetNearestAction, [11](#)
 - GetNearestAirlock, [11](#)
 - GetNearestBreathable, [12](#)
 - GetScnAstrCount, [12](#)
 - GetUACSVersion, [12](#)
 - GetVslAstrInfo, [12](#)
 - InBreathable, [13](#)
 - Ingress, [13](#)
 - TriggerAction, [13](#)
- UACS::Cargo, [14](#)
 - clbkDrainResource, [15](#)
 - clbkGetCargoInfo, [15](#)
 - clbkPackCargo, [15](#)
 - clbkUnpackCargo, [15](#)
 - GetUACSVersion, [15](#)
- UACS::Cargo::CargoInfo, [16](#)
 - breathable, [17](#)
 - frontPos, [17](#)
 - resource, [17](#)
 - unpacked, [17](#)
- UACS::CargoInfo, [18](#)
 - breathable, [18](#)
- UACS::GroundInfo, [18](#)
 - colDir, [19](#)
 - pos, [19](#)
 - rowDir, [19](#)
- UACS::Module, [19](#)
- AddAstronaut, [22](#)
- AddCargo, [22](#)
- clbkSaveState, [22](#)
- DeleteCargo, [23](#)
- DrainGrappledResource, [23](#)
- DrainScenarioResource, [23](#)
- DrainStationResource, [24](#)
- DrawAstrInfo, [24](#)
- DrawCargoInfo, [24](#)
- EgressAstronaut, [25](#)
- GetAstrInfoByHandle, [25](#)
- GetAstrInfoByIndex, [25](#)
- GetAvailAstrCount, [27](#)
- GetAvailAstrName, [27](#)
- GetAvailCargoCount, [27](#)
- GetAvailCargoName, [27](#)
- GetCargoInfoByHandle, [28](#)
- GetCargoInfoByIndex, [28](#)
- GetScnAstrCount, [28](#)
- GetScnCargoCount, [28](#)
- GetStationResources, [29](#)
- GetTotalAstrMass, [29](#)
- GetTotalCargoMass, [29](#)
- GetUACSVersion, [29](#)
- GetVslAstrInfo, [30](#)
- GrappleCargo, [31](#)
- Module, [21](#)
- PackCargo, [31](#)
- ParseScenarioLine, [31](#)
- ReleaseCargo, [32](#)
- SetAstrInfoByHandle, [32](#)
- SetAstrInfoByIndex, [32](#)
- TransferAstronaut, [33](#)
- UnpackCargo, [33](#)
- UACS::NearestAction, [34](#)
- UACS::NearestAirlock, [34](#)
- UACS::SlotInfo, [34](#)
 - holdDir, [35](#)
- UACS::StationInfo, [35](#)
- UACS::VslAstrInfo, [35](#)
- UACS::VslCargoInfo, [36](#)
 - astrMode, [36](#)
- UnpackCargo
 - UACS::Module, [33](#)
- unpacked
 - UACS::Cargo::CargoInfo, [17](#)