Related Links

- Data Load Helper
- Visual Effects Service
- Event Service
- Global Data Service
- Rendering Services
- Window and Camera Services
- Weather System Service
- ISimObjects
- PDK Services
- PDK Types
- SDK Overview
- PDK Overview

ISimObject

Overview

The SimObject API utilizes a service-based methodology for building simulation behaviors to be visualized in Prepar3D. The API enables a solution developer to create a simulation object (SimObject) complete with customized behaviors, input properties (also referred to as events or triggers), and state properties (also referred to as simvars or simply properties). These properties can be referenced in content such as SimObject gauges, animations, and scenario scripts which are discussed in more detail in other parts of the SDK. The properties are text-based, and can be referenced in the same way as the stock simvars and events are referred to in other parts of the Prepar3D SDK.

Types.h

Includes common data types found throughout the ISimObject Samples.

PSaveLoadCallback() - Function pointer for state save/load (.FXML files). This pointer is passed to each ISimulation when it's time for save/load Section names will automatically be constructed as: [SectionName.Instance.SimObjectID] allowing for multiple instances of a system (e.g. multi-engines) Constructed section names are limited to a maximum of 128 characters. SIM_DATA_TYPE allows saving either numeric or string data.

```
typedef HRESULT (STDMETHODCALLTYPE *PSaveLoadCallback)(__in LPCTSTR pszSection, __in
      unsigned int uInstance, __in LPCTSTR pszKeyword, __inout void* pvVal, __in const
      SAVED_DATA_TYPE eDataType);\n\n
```

The property type enum used in the property string->ID lookup. See IBaseObject.

```
typedef enum
{
   PROPERTY_TYPE_EVENT,
   PROPERTY_TYPE_EVENT_WITH_SUBSTRING_INPUT,
   PROPERTY_TYPE_EVENT_VECTOR,
   PROPERTY_TYPE_EVENT_STRING,
   PROPERTY_TYPE_DOUBLE,
   PROPERTY_TYPE_DOUBLE_WITH_SUBSTRING_INPUT,
   PROPERTY_TYPE_STRING,
   PROPERTY_TYPE_VECTOR,
   NUM_PROPERTY_TYPES,
}  PROPERTY_TYPE;
```

Enum definition for data types used when saving / loading. See ISimulation and PSaveLoadCallback.

```
typedef enum
```

```
{
    SAVED_DATA_TYPE_DOUBLE,
    SAVED_DATA_TYPE_STRING,
    NUM_SAVED_DATA_TYPES
}   SAVED_DATA_TYPE;
```

Enum definition for the various modes in which an Artificially Intelligent (AI) object can be.

```
typedef enum
{
    UNITMODE_SLEEP,
    UNITMODE_ZOMBIE,
    UNITMODE_WAYPOINT,
    UNITMODE_TAKEOFF,
    UNITMODE_LANDING,
    UNITMODE_TAXI,
    UNITMODE_WORKING,
    UNITMODE_WAITING,
}   UNITMODE_TYPE;
```

A **BasicWaypoint** is generally used to define a point along a path. It is used primarily by the AI system.

```
class BasicWaypoint
{
public:
    DXYZ vLonAltLat; //Lat/Lon (Radians), Alt (Feet)
    double dHeading; //Radians
};
```

Enum definition for various network modes. See Object Mode Monitoring.

```
typedef enum
{
    NET_MODE_TYPE_NORMAL,    // The object is owned by the current client
    NET_MODE_TYPE_REMOTE,    // The object is owned by another client
    NET_MODE_TYPE_MASTER,    // Shared Cockpit: The object is owned by the current client
    NET_MODE_TYPE_SLAVE,     // Shared Cockpit: The object is owned by another client
} NET_MODE_TYPE;
```

Versioning Code

Compiled with the PDK and ISimObject interfaces will be expected to function in subsequent versions of the SDK. To do so, each interface name is appended with the version number, and is derived from the preceding version. The preceding versions will be maintained intact in the Legacy subfolder. It is recommended that all new code utilizes the latest version when defining your objects and each QueryInterface supports all versions. Internal to Prepar3D, the earliest version possible will be used.

Namespaces

P3D

Prepar3D SDK namespace used primarily for the PDK and its services.

Classes

class	**ISimObjectManagerV440**
class	**ISimObjectV440**
class	**ISimulationV310**
class	**IBaseObjectV450**
class	**ISubSystemFactoryV440**
class	**WorldConstants**
class	**SurfaceInfoV400**

class	**WeatherInfoV400**
class	**IMassPropertiesV01**
class	**IForceMomentsV01**
class	**ICollisionServiceV01**
class	**IAircraftServiceV01**
class	**IAirplaneServiceV01**
class	**IRotorcraftServiceV01**
class	**IBoatServiceV01**
class	**IGroundVehicleServiceV01**
class	**IAtcServiceV01**
class	**IRadarSignatureServiceV01**
class	**IDoorServiceV01**
class	**IFuelServiceV400**
class	**ISurfaceQueryManagerV400**
class	**IWaypointQueryManagerV400**
class	**IAvatarSimV01**
class	**IAnimationControllerV01**
class	**IAvatarAttachServiceV01**
class	**IMarkerManagerV310**
class	**IDesignatorServiceV340**
class	**IRayTraceManagerV340**
class	**IEmissionsServiceV340**
class	**IRadioSystemV400**
class	**IAttachmentServiceV430**
class	**IAIBehaviorManagerV01**
class	**IAIBehaviorWingmanFormationV01**
class	**IAIBehaviorAttackerV400**
class	**IAIBehaviorPursueV01**
class	**IAIBehaviorCombatAirPatrolV01**
class	**IAIBehaviorCloseAirSupportV01**
class	**IAIBehaviorSearchTrackV01**
class	**ISimObjectAIV02**
class	**IAirplaneAIServiceV02**
class	**IHelicopterAIServiceV420**
class	**IGroundVehicleAIServiceV01**
class	**IWeaponsSystemV440**
class	**IWeaponServiceV420**
class	**ICountermeasureSystemV01**
class	**ICountermeasureServiceV02**
class	**IGunSystemV440**
class	**IGunV400**
class	**IFireControlSystemV01**
class	**IGuidanceSystemV01**
class	**IPylonServiceV01**
class	**ArticulatedPart**
class	**ArticulatedParameter**

class	**IPduBuilderV440**
class	**IPduReaderV440**
class	**IPduCallbackV440**
class	**IDISManagerV450**
class	**IDISServiceV400**
union	**ArticulatedParameter.__unnamed__**

Class Documentation

§ P3D::ISimObjectManagerV440

class P3D::ISimObjectManagerV440

Handles tasks that are not associated with an instance of a simobject. This includes:

- Registration of implementation factories and associated properties
- Global application properties (e.g. world constants)
- Object queries

Inherits ISimObjectManagerV430.

Private Member Functions

virtual HRESULT	**RegisterSimulationCategory** (__in GUID guidCategory, __in LPCWSTR pszCategoryName, __in __notnull **PSimCreateFunc** pcbCreateFunction) **PURE**
virtual HRESULT	**RegisterProperty** (__in GUID guidCategory, __in LPCWSTR pszPropertyName, __in LPCWSTR pszPropertyBaseUnits, __in __notnull **PPropertyCallback** pcbProperty) **PURE**
virtual HRESULT	**RegisterProperty** (__in GUID guidCategory, __in LPCWSTR pszPropertyName, __in LPCWSTR pszPropertyBaseUnits, __in __notnull **PPropertyVectorCallback** pcbProperty) **PURE**
virtual HRESULT	**RegisterProperty** (__in GUID guidCategory, __in LPCWSTR pszPropertyName, __in __notnull **PPropertyStringCallback** pcbProperty) **PURE**
virtual HRESULT	**RegisterProperty** (__in GUID guidCategory, __in LPCWSTR pszPropertyName, __in LPCWSTR pszPropertyBaseUnits, __in __notnull **PPropertyCallbackWithSubString** pcbProperty) **PURE**
virtual HRESULT	**RegisterProperty** (__in GUID guidCategory, __in LPCWSTR pszPropertyName, __in LPCWSTR pszPropertyBaseUnits, __in __notnull **PEventCallback** pcbEvent, __in **EVENTTYPE** eType) **PURE**
virtual HRESULT	**RegisterProperty** (__in GUID guidCategory, __in LPCWSTR pszPropertyName, __in LPCWSTR pszPropertyBaseUnits, __in __notnull **PEventVectorCallback**) **PURE**
virtual HRESULT	**RegisterProperty** (__in GUID guidCategory, __in LPCWSTR pszPropertyName, __in __notnull **PEventStringCallback**) **PURE**
virtual HRESULT	**RegisterProperty** (__in GUID guidCategory, __in LPCWSTR pszPropertyName, __in LPCWSTR pszPropertyBaseUnits, __in __notnull **PEventCallbackWithSubString** pcbEvent) **PURE**
virtual HRESULT	**GetWorldConstants** (__out **WorldConstants** &) const **PURE**
virtual HRESULT	**GetUnitCode** (__in LPCWSTR pszPropertyUnits, __out int &iUnitCode) const **PURE**
virtual HRESULT	**GetObject** (__in UINT idObject, __out IBaseObjectV400 **ppObject) const **PURE**
virtual HRESULT	**GetObject** (__in UINT idObject, __in REFIID riid, __out void **ppvObject) const **PURE**
virtual HRESULT	**GetUserObject** (__out IBaseObjectV400 **ppUserObject) const **PURE**
virtual HRESULT	**GetUserObject** (__in REFIID riid, __out void **ppvUserObject) const **PURE**
virtual HRESULT	**RegisterOnObjectCreateCallback** (__in __notnull **POnObjectCreateCallback** pCb) **PURE**
virtual HRESULT	**RegisterOnObjectRemoveCallback** (__in __notnull **POnObjectRemoveCallback** pCb) **PURE**
virtual HRESULT	**RegisterOnUserObjectChangedCallback** (__in __notnull **POnUserObjectChangedCallback** pCb) **PURE**
virtual HRESULT	**GetObjectsInRadius** (__in const **DXYZ** &vLonAltLat, __in float fRadiusFeet, __inout UINT &nObjects, __out UINT *rgObjectIDs) const **PURE**
virtual HRESULT	**GetNonTrafficObjectsInRadius** (__in const **DXYZ** &vLonAltLat, __in float fRadiusFeet, __inout UINT &nObjects, __out UINT *rgObjectIDs) const **PURE**
virtual float	**GetRealismSetting** () const **PURE**

virtual **BOOL**	**IsCrashDetectionOn** () const **PURE**
virtual **BOOL**	**IsCollisionBetweenObjectsOn** () const **PURE**
virtual float	**GetCrashToleranceScalar** () const **PURE**
virtual HRESULT	**RemoveObject** (__in UINT idObject) **PURE**
virtual HRESULT	**CreateObject** (__in __notnull LPCWSTR pszTitle, __out UINT &idObject) **PURE**
virtual HRESULT	**GetUserAvatar** (__out IBaseObjectV400 **ppUserAvatar) const **PURE**
virtual HRESULT	**GetUserAvatar** (__in REFIID riid, __out void **ppvUserAvatar) const **PURE**
virtual UINT	**GetNumberOfCategories** () const **PURE**
virtual HRESULT	**GetCategoryId** (__out GUID &guidCategoryId, __out __notnull LPWSTR pszCategoryFriendlyName, __in UINT uNameLen, __out **BOOL** &bIsNativeSimulation, __in UINT iIndex) const **PURE**

Member Function Documentation

§ CreateObject()

virtual HRESULT CreateObject (__in __notnull LPCWSTR pszTitle,
__out UINT & idObject
)

private virtual

Attempts to create an object with the given container title.

Parameters

pszTitle The container title object to be created.

idObject The object id of the newly created object.

Returns

S_OK if the object was successfully created, E_FAIL otherwise.

§ GetCategoryId()

virtual HRESULT GetCategoryId (__out GUID & guidCategoryId,
__out __notnull LPWSTR pszCategoryFriendlyName,
__in UINT uNameLen,
__out **BOOL** & bIsNativeSimulation,
__in UINT iIndex
) const

private virtual

Gets the simulation category unique GUID and friendly string name.

Parameters

iIndex	The unique (0-based) index into the list of registered categories.
guidCategoryId	The GUID id unique to this simulation category.
pszCategoryFriendlyName	The friendly string name for the category. These are guaranteed to be unique only for native simulations in core Prepar3D.
uNameLen	The maximum length of the allocated string friendly name.
bIsNativeSimulation	Indicates if this simulation is natively implemented in core Prepar3D (TRUE) or externally developed (FALSE).

Returns

S_OK if the category is successfully found. E_INVALIDARG if the index exceeds the maximum index of the registered simulation list.

Remarks

Indexes are gauaranteed to remain unique for the lifetime of a Prepar3D instance.

§ GetCrashToleranceScalar()

virtual float GetCrashToleranceScalar ( ) const

private virtual

The user-selected scalar for determining crash tolerance

§ GetNonTrafficObjectsInRadius()

virtual HRESULT GetNonTrafficObjectsInRadius (__in const **DXYZ** & vLonAltLat,
__in float fRadiusFeet,
__inout UINT & nObjects,
__out UINT * rgObjectIDs
) const private virtual

Returns a list of object IDs for a given radius. Does not include traffic

Parameters

nObjects IN: The max number of elements requested. This must be no smaller than the size of the array pointed to by rgObjectIDs

nObjects OUT: The actual number of objects found.

rgObjectIDs Address of array in which object IDs are returned.
NOTE: It is the callers responsibility to allocate the array's required memory.

§ GetNumberOfCategories()

virtual UINT GetNumberOfCategories ( ) const private virtual

Gets the number of registered simulations.

Returns

Number of registered simulation categories

Remarks

Note that if this queried at startup during DLL loading, some externally developed categories may not yet be registered. Core Prepar3D native simulations will be registered by that time.

§ GetObject() [1/2]

virtual HRESULT GetObject (__in UINT idObject,
__out IBaseObjectV400 ** ppObject
) const private virtual

Gets another IBaseObject ref for a given ID

§ GetObject() [2/2]

virtual HRESULT GetObject (__in UINT idObject,
__in REFIID riid,
__out void ** ppvObject
) const private virtual

Gets another specific version of an IBaseObject ref for a given ID

§ GetObjectsInRadius()

virtual HRESULT GetObjectsInRadius (__in const **DXYZ** & vLonAltLat,
__in float fRadiusFeet,
__inout UINT & nObjects,
__out UINT * rgObjectIDs
) const private virtual

Returns a list of object IDs for a given radius.

Parameters

nObjects IN: the max number of elements requested. This must be no smaller than the size of the array pointed to by rgObjectIDs.

nObjects OUT: the actual number of objects found.

rgObjectIDs address of array in which object IDs are returned.
NOTE: It is the callers responsibility to allocate the array's required memory

§ GetRealismSetting()

virtual float GetRealismSetting ( ) const private virtual

The user-selected general realism scalar, where 0.0 is "easy" and 1.0 is "real". This can be used to scale your implementation as appropriate

§ GetUnitCode()

virtual HRESULT GetUnitCode (__in LPCWSTR pszPropertyUnits,
__out int & iUnitCode
) const private virtual

Decodes a string units to its integer ID. This can be useful to get at initialization as it is less performanct to query properties using the string version. e.g. "feet per second" to ID.

§ GetUserAvatar() [1/2]

virtual HRESULT GetUserAvatar (__out IBaseObjectV400 ** ppUserAvatar) const private virtual

Gets an IBaseObject ref for the current user avatar.

Parameters

ppUserAvatar The IBaseObject ref for the current user avatar.

Returns

S_OK if the object was successfully found, E_FAIL otherwise.

Remarks

The user avatar and the user object may be the same if the user has selected an avatar object.

§ GetUserAvatar() [2/2]

virtual HRESULT GetUserAvatar (__in REFIID riid,
__out void ** ppvUserAvatar
) const private virtual

Gets an IBaseObject ref for the current user avatar.

Parameters

ppvUserAvatar The IBaseObject ref for the current user avatar.

riid Interface ID.

Returns

S_OK if the object was successfully found, E_FAIL otherwise.

Remarks

The user avatar and the user object may be the same if the user has selected an avatar object.

§ GetUserObject() [1/2]

virtual HRESULT GetUserObject (__out IBaseObjectV400 ** ppUserObject) const private virtual

Gets an IBaseObject ref for the current user object.
NOTE: If the user object is the Viewer and there is a previous user object, this will return the previous user object. Otherwise, it will return the Viewer.

§ GetUserObject() [2/2]

virtual HRESULT GetUserObject (__in REFIID riid,
__out void ** ppvUserObject
) const

private virtual

Gets a specific version IBaseObject ref for the current user object.
NOTE: If the user object is the Viewer and there is a previous user object, this will return the previous user object. Otherwise, it will return the Viewer.

§ GetWorldConstants()

virtual HRESULT GetWorldConstants (__out **WorldConstants** &) const

private virtual

World Constants are constant values describing the Earth atmosphere and geometry.
NOTE: While this is accessed through IBaseObject, these values will be constant for all SimObjects, and a single static copy could be shared across multiple instances.

§ IsCollisionBetweenObjectsOn()

virtual **BOOL** IsCollisionBetweenObjectsOn ( ) const

private virtual

The user-selected flag for whether to detect crashes between simobjects or not

§ IsCrashDetectionOn()

virtual **BOOL** IsCrashDetectionOn ( ) const

private virtual

The user-selected flag that dictates whether to process a crash or not

§ RegisterOnObjectCreateCallback()

virtual HRESULT
RegisterOnObjectCreateCallback (__in __notnull **POnObjectCreateCallback** pCb)

private virtual

Used to register a callback function that is called upon creation of any new object. See **types.h** for callback definition.

§ RegisterOnObjectRemoveCallback()

virtual HRESULT
RegisterOnObjectRemoveCallback (__in __notnull **POnObjectRemoveCallback** pCb) private virtual

Used to register a callback function that is called upon destruction of any existing object. The call is just prior to destruction. See **types.h** for callback definition.

§ RegisterOnUserObjectChangedCallback()

virtual HRESULT
RegisterOnUserObjectChangedCallback (__in __notnull **POnUserObjectChangedCallback** pCb) private virtual

Used to register a callback function that is called whenever the user is moved from one object to another. See **types.h** for callback definition.

§ RegisterProperty() [1/8]

virtual HRESULT RegisterProperty (__in GUID guidCategory,
__in LPCWSTR pszPropertyName,
__in LPCWSTR pszPropertyBaseUnits,
__in __notnull **PPropertyCallback** pcbProperty
) private virtual

Property "simvar" and "event" registrations. For a specific simobject implementation (guid), associates: property string name, units, and callback pointer (defined above)

Note
Input: Double

§ RegisterProperty() [2/8]

virtual HRESULT
RegisterProperty (__in GUID guidCategory,
__in LPCWSTR pszPropertyName,
__in LPCWSTR pszPropertyBaseUnits,
__in __notnull **PPropertyVectorCallback** pcbProperty
) private virtual

Property "simvar" and "event" registrations. For a specific simobject implementation (guid), associates: property string name, units, and callback pointer (defined above)

Note
Input: Vector (DXYZ)

§ RegisterProperty() [3/8]

```
virtual HRESULT RegisterProperty ( __in GUID                          guidCategory,
                                   __in LPCWSTR                       pszPropertyName,
                                   __in __notnull PPropertyStringCallback pcbProperty
                                 )
```

private virtual

Property "simvar" and "event" registrations. For a specific simobject implementation (guid), associates: property string name, units, and callback pointer (defined above)

Note
Input: String

§ RegisterProperty() [4/8]

```
virtual HRESULT
RegisterProperty ( __in GUID                              guidCategory,
                   __in LPCWSTR                           pszPropertyName,
                   __in LPCWSTR                           pszPropertyBaseUnits,
                   __in __notnull PPropertyCallbackWithSubString pcbProperty
                 )
```

private virtual

Property "simvar" and "event" registrations. For a specific simobject implementation (guid), associates: property string name, units, and callback pointer (defined above)

Note
Input: Double (with secondary substring input)

§ RegisterProperty() [5/8]

```
virtual HRESULT RegisterProperty ( __in GUID                          guidCategory,
                                   __in LPCWSTR                       pszPropertyName,
                                   __in LPCWSTR                       pszPropertyBaseUnits,
                                   __in __notnull PEventCallback      pcbEvent,
                                   __in EVENTTYPE                     eType
                                 )
```

private virtual

Property "simvar" and "event" registrations. For a specific simobject implementation (guid), associates: property string name, units, and callback pointer (defined above)

Note
Input: Event

§ RegisterProperty() [6/8]

```
virtual HRESULT RegisterProperty ( __in GUID         guidCategory,
                                   __in LPCWSTR      pszPropertyName,
                                   __in LPCWSTR      pszPropertyBaseUnits,
                                   __in __notnull    PEventVectorCallback
                                 )
```

private virtual

Property "simvar" and "event" registrations. For a specific simobject implementation (guid), associates: property string name, units, and callback pointer (defined above)

Note
Input: Event vector

§ RegisterProperty() [7/8]

```
virtual HRESULT RegisterProperty ( __in GUID         guidCategory,
                                   __in LPCWSTR      pszPropertyName,
                                   __in __notnull    PEventStringCallback
                                 )
```

private virtual

Property "simvar" and "event" registrations. For a specific simobject implementation (guid), associates: property string name, units, and callback pointer (defined above)

Note
Input: Event string

§ RegisterProperty() [8/8]

```
virtual HRESULT
RegisterProperty        ( __in GUID                          guidCategory,
                          __in LPCWSTR                       pszPropertyName,
                          __in LPCWSTR                       pszPropertyBaseUnits,
                          __in __notnull PEventCallbackWithSubString pcbEvent
                        )
```

private virtual

Property "simvar" and "event" registrations. For a specific simobject implementation (guid), associates: property string name, units, and callback pointer (defined above)

Note
Input: Event double (with secondary substring input)

§ RegisterSimulationCategory()

virtual HRESULT
RegisterSimulationCategory (__in GUID guidCategory,
__in LPCWSTR pszCategoryName,
__in __notnull **PSimCreateFunc** pcbCreateFunction
) private virtual

Registers an ISimObject implementation at load time with: unique ID, friendly category name (e.g. "airplane"), and factory function pointer. The "pszCategoryName" is a high-level categorization used primarily for UI (e.g. "airplane"). Mainly, it is used as a filter to exclude objects from appearing in the Vehicle Select screen. If you create a unique category name, ensure you add the name to the User Objects key in the Prepar3D.cfg's [Main] section.

§ RemoveObject()

virtual HRESULT RemoveObject (__in UINT idObject) private virtual

RemoveObject - Remove any local non-user object

Parameters

idObject The id of the object to remove.

§ P3D::ISimObjectV440

class P3D::ISimObjectV440

Interface from which object implementations must derive.

Inherits ISimObjectV400.

Private Member Functions

virtual HRESULT	**LoadConstantData** (__inout void **ppConstantData) **PURE**
virtual HRESULT	**UnloadConstantData** (__inout void **ppConstantData) **PURE**
virtual HRESULT	**LoadDynamicData** () **PURE**
virtual HRESULT	**Init** () **PURE**
virtual HRESULT	**DeInit** () **PURE**
virtual **BOOL**	**SupportsLabels** () const **PURE**
virtual HRESULT	**SetSupportsLabels** (**BOOL** bOn) **PURE**
virtual void	**OnModeChange** (int bfNewModes) **PURE**
virtual void	**OnPositionInit** () **PURE**
virtual void	**OnSpeedInit** (float fSpeed) **PURE**
virtual HRESULT	**QueryBaseObject** (REFIID riid, void **ppv) **PURE**
virtual HRESULT	**GetMainSimRate** (__out float &fSimRate) const **PURE**
virtual HRESULT	**GetMainMinMaxSimRates** (__out float &fMinSimRate, __out float &fMaxSimRate) const **PURE**
virtual HRESULT	**SetMainMinMaxSimRates** (__in float fMinSimRate, __in float fMaxSimRate) **PURE**

Member Function Documentation

§ DeInit()

virtual HRESULT DeInit ( ) private virtual

Can be used to release inter-system references prior to the object being destroyed.

§ GetMainMinMaxSimRates()

virtual HRESULT GetMainMinMaxSimRates (__out float & fMinSimRate,
__out float & fMaxSimRate
) const private virtual

Provide the minimum and maximum main simulation rate (Hz). Typically the world position update rate.

§ GetMainSimRate()

virtual HRESULT GetMainSimRate (__out float & fSimRate) const

private virtual

Provide the main simulation rate (Hz). Typically the world position update rate. This assumes the simulation is registered at a constant simulation rate. Beginning with V440 of this interface, a min and max rate can be used if desired, so the following GetMainSimRate may give more accurate information.

§ Init()

virtual HRESULT Init ( )

private virtual

An appropriate place to initialize data and establish references between subsystems.

§ LoadConstantData()

virtual HRESULT LoadConstantData (__inout void ** ppConstantData)

private virtual

Where your object class should load data from the disk. The return data is cached for subsequent instances of this same object.

§ LoadDynamicData()

virtual HRESULT LoadDynamicData ( )

private virtual

Called on each object instance. This would be an appropriate place to create the object's runtime subsystems.

§ OnModeChange()

virtual void OnModeChange (int bfNewModes)

private virtual

Called upon change in modes (pause, slew, etc...)

§ OnPositionInit()

virtual void OnPositionInit ( )

private virtual

Called whenever Prepar3D has changed the position of this object outside of its own simulation implementation. Examples of this would be positioning from the User Interface, Slew Mode, or terrain resolution changing.

§ OnSpeedInit()

virtual void OnSpeedInit (float fSpeed) private virtual

Called whenever Prepar3D has changed the speed of this object outside of its own simulation implementation. This would occur normally if positioning from the User Interface. Accessor to get the base object from the ISimObject. NOTE: This previously required a return type of IBaseObjectV400**. Existing implementations will downcast to void** automatically.

§ QueryBaseObject()

```
virtual HRESULT QueryBaseObject ( REFIID  riid,
                                  void **  ppv
                                )
```
private virtual

§ SetMainMinMaxSimRates()

```
virtual HRESULT SetMainMinMaxSimRates ( __in float  fMinSimRate,
                                        __in float  fMaxSimRate
                                      )
```
private virtual

Sets minimum and maximum main simulation rate (Hz). Typically the world position update rate. This is typically called if there is a desire to synchronize the rates of two or more objects. For example an aircraft and an aircraft carrier, or to prevent perceived jitter when viewing an object from a camera attached to another object. Note: It is the responsibility of the ISimObject developer to determine if the rates are appropriate for the implementation, and subsequently call IBaseObject::RegisterSimulation() (again) with the new rates with the relevant ISimulations.

§ SetSupportsLabels()

virtual HRESULT SetSupportsLabels (**BOOL** bOn) private virtual

Requests this SimObject to support labels. Return S_OK if the new setting is accepted. If not, return an error code, such as E_FAIL. This value should maintain by this class and returned when requested by SupportsLabel (). You may choose for your class to not support labels. This setting will not override settings in the Traffic Settings.

§ SupportsLabels()

virtual **BOOL** SupportsLabels ( ) const private virtual

Defines if the SimObject will or will not support labels to be displayed.

§ UnloadConstantData()

virtual HRESULT UnloadConstantData (__inout void ** ppConstantData) private virtual

Where your object class should unload data from the disk. The return data is cached for subsequent instances of this same object.

§ P3D::ISimulationV310

class P3D::ISimulationV310

Interface to individual simulation subsystems.

Inherits ISimulationV01.

Private Member Functions

virtual HRESULT	**Update** (double dDeltaT) **PURE**
virtual HRESULT	**SaveLoadState** (__in __notnull **PSaveLoadCallback** pfnCallback, __in const **BOOL** bSave) **PURE**
virtual HRESULT	**Serialize** (__in NetOutPublic &netOut) **PURE**
virtual HRESULT	**Deserialize** (__in NetInPublic &netIn) **PURE**

Member Function Documentation

§ Deserialize()

virtual HRESULT Deserialize (__in NetInPublic & netIn) private virtual

Only called when in an active multiplayer session. This function can be implemented to deserialize network packets that have been sent by other clients for this ISimulation instance. The NetInPublic interface is defined in **NetInOutPublic.h**.

§ SaveLoadState()

virtual HRESULT SaveLoadState (__in __notnull **PSaveLoadCallback** pfnCallback,
__in const **BOOL** bSave
)
private virtual

Called when either saving or loading a Prepar3D scenario. The function pointer allows your code to save and load "name - value" pairs in the Prepar3D .fxml file. See the definition for PSaveLoadCallback and the supported data type enum SAVED_DATA_TYPE.

§ Serialize()

virtual HRESULT Serialize (__in NetOutPublic & netOut) private virtual

Only called when in an active multiplayer session. This function can be implemented to create network packets that are then broadcast to other clients for this ISimulation instance. The NetOutPublic interface is defined in **NetInOutPublic.h**

§ Update()

virtual HRESULT Update (double dDeltaT) private virtual

Called by Prepar3D at the iteration rate specified when this ISimulation interface is registered using RegisterSimulation() in the IBaseObject interface.

§ P3D::IBaseObjectV450

class P3D::IBaseObjectV450

Object interface on the host side for providing platform information and services for the object

Inherits IBaseObjectV440.

Private Member Functions

virtual UINT	**GetId** () const **PURE**
virtual HRESULT	**GetMissionId** (__out GUID &guid) const **PURE**
virtual **BOOL**	**IsUser** () const **PURE**
virtual UINT	**GetObjectGroupAssociationId** () const **PURE**
virtual void	**SetObjectGroupAssociationId** (UINT uAssociationId) **PURE**
virtual **BOOL**	**InObjectFoeList** (UINT id) const **PURE**
virtual void	**SetObjectFoeList** (UINT *uEnteredFoeID, UINT size) **PURE**
virtual **BOOL**	**InObjectFriendList** (UINT id) const **PURE**
virtual void	**SetObjectFriendList** (UINT *uEnteredFriendID, UINT size) **PURE**
virtual int	**GetMode** () const **PURE**
virtual HRESULT	**SetCrashMode** (double dDeltaT) **PURE**
virtual HRESULT	**GetPosition** (__out **DXYZ** &vLonAltLat, __out **DXYZ** &vPHB, __out **DXYZ** &vLonAltLatVel, __out **DXYZ** &vPHBVel) const **PURE**
virtual HRESULT	**SetPosition** (__in const **DXYZ** &vLonAltLat, __in const **DXYZ** &vPHB, __in const **DXYZ** &vLonAltLatVel, __in const **DXYZ** &vPHBVel, __in **BOOL** bIsOnGround, __in double dDeltaT) **PURE**
virtual HRESULT	**InitPosition** (__in const **DXYZ** *pvLonAltLat, __in const **DXYZ** *pvPHB, __in const **DXYZ** *pvLonAltLatVel, __in const **DXYZ** *pvPHBVel, __in **BOOL** bSetOnGround) **PURE**
virtual **BOOL**	**IsOnGround** () const **PURE**
virtual HRESULT	**RotateWorldToBody** (__in const **DXYZ** &vWorld, __out **DXYZ** &vBody) const **PURE**
virtual HRESULT	**RotateBodyToWorld** (__in const **DXYZ** &vBody, __out **DXYZ** &vWorld) const **PURE**
virtual HRESULT	**RegisterSimulation** (__in __notnull ISimulation *pSimulation, float fRateHz) **PURE**
virtual HRESULT	**RegisterSimulation** (__in __notnull ISimulation *pSimulation, float fMinRateHz, float fMaxRateHz) **PURE**
virtual HRESULT	**GetMainMinMaxSimRates** (__out float &fMinHz, __out float &fMaxHz) const **PURE**
virtual HRESULT	**SetMainMinMaxSimRates** (__in float fMinHz, __in float fMaxHz) **PURE**
virtual HRESULT	**RegisterService** (__in REFGUID guidService, __in __notnull IUnknown *punkService) **PURE**
virtual HRESULT	**UnregisterService** (__in REFGUID guidService) **PURE**
virtual HRESULT	**GetPropertyCodeAndIndex** (__in **PROPERTY_TYPE** eType, __in LPCWSTR pszPropertyName, __out int &iPropertyCode, __inout int &iIndex) const **PURE**
virtual HRESULT	**GetProperty** (__in int iPropertyCode, __in int iUnitCode, __out double &dProperty, __in int index=0) const **PURE**
virtual HRESULT	**GetProperty** (__in LPCWSTR pszPropertyName, __in int iUnitCode, __out double &dProperty, __in int index=0) const **PURE**
virtual HRESULT	**GetProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszUnitCode, __out double &dProperty, __in int index=0) const **PURE**
virtual HRESULT	

	GetProperty (__in int iPropertyCode, __in int iUnitCode, __out **DXYZ** &dProperty, __in int index=0) const **PURE**
virtual HRESULT	**GetProperty** (__in LPCWSTR pszPropertyName, __in int iUnitCode, __out **DXYZ** &dProperty, __in int index=0) const **PURE**
virtual HRESULT	**GetProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszUnitCode, __out **DXYZ** &dProperty, __in int index=0) const **PURE**
virtual HRESULT	**GetProperty** (__in int iPropertyCode, __out LPWSTR pszProperty, __in UINT uLength, __in int index=0) const **PURE**
virtual HRESULT	**GetProperty** (__in LPCWSTR pszPropertyName, __out LPWSTR pszProperty, __in UINT uLength, __in int index=0) const **PURE**
virtual HRESULT	**GetProperty** (__in int iPropertyCode, __in LPCWSTR pszSecondarySubstring, __in int iUnitCode, __out double &dProperty, __in int index=0) const **PURE**
virtual HRESULT	**GetProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszSecondarySubstring, __in int iUnitCode, __out double &dProperty, __in int index=0) const **PURE**
virtual HRESULT	**GetProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszSecondarySubstring, __in LPCWSTR pszUnitCode, __out double &dProperty, __in int index=0) const **PURE**
virtual HRESULT	**TriggerProperty** (__in int iPropertyCode, __in int iUnitCode, __in double dData, __in int index) const **PURE**
virtual HRESULT	**TriggerProperty** (__in LPCWSTR pszPropertyName, __in int iUnitCode, __in double dData, __in int index) const **PURE**
virtual HRESULT	**TriggerProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszUnitCode, __in double dData, __in int index) const **PURE**
virtual HRESULT	**TriggerProperty** (__in int iPropertyCode, __in int iUnitCode, __in const **DXYZ** &vData, __in int index) const **PURE**
virtual HRESULT	**TriggerProperty** (__in LPCWSTR pszPropertyName, __in int iUnitCode, __in const **DXYZ** &vData, __in int index) const **PURE**
virtual HRESULT	**TriggerProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszUnitCode, __in const **DXYZ** &vData, __in int index) const **PURE**
virtual HRESULT	**TriggerProperty** (__in int iPropertyCode, __in LPCWSTR pszData, __in int index) const **PURE**
virtual HRESULT	**TriggerProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszData, __in int index) const **PURE**
virtual HRESULT	**TriggerProperty** (__in int iPropertyCode, __in LPCWSTR pszSecondarySubstring, __in int iUnitCode, __in double dData, __in int index) const **PURE**
virtual HRESULT	**TriggerProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszSecondarySubstring, __in int iUnitCode, __in double dData, __in int index) const **PURE**
virtual HRESULT	**TriggerProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszSecondarySubstring, __in LPCWSTR pszUnitCode, __in double dData, __in int index) const **PURE**
virtual HRESULT	**RegisterProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszPropertyBaseUnits, __in __notnull **PPropertyCallback** pcbProperty) **PURE**
virtual HRESULT	**RegisterProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszPropertyBaseUnits, __in __notnull **PEventCallback** pcbEvent, __in **EVENTTYPE** eType) **PURE**
virtual HRESULT	**RegisterProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszPropertyBaseUnits, __in __notnull **PPropertyVectorCallback** pcbProperty) **PURE**

virtual HRESULT	**RegisterProperty** (__in LPCWSTR pszPropertyName, __in LPCWSTR pszPropertyBaseUnits, __in __notnull **PEventVectorCallback**) **PURE**
virtual HRESULT	**RegisterProperty** (__in LPCWSTR pszPropertyName, __in __notnull **PPropertyStringCallback** pcbProperty) **PURE**
virtual HRESULT	**RegisterProperty** (__in LPCWSTR pszPropertyName, __in __notnull **PEventStringCallback**) **PURE**
virtual HANDLE	**RegisterSystemMalfunction** (__in REFGUID guidMalfunction, __in LPCWSTR pszType, __in LPCWSTR pszBaseName, __in LPCWSTR pszInstanceName, __in int nSubIndex) **PURE**
virtual float	**GetSystemHealth** (HANDLE hSystem) const **PURE**
virtual HRESULT	**DecrementHealthPoints** (__in float fDamagePoints) **PURE**
virtual float	**GetHealthPoints** () const **PURE**
virtual void	**SetHealthPoints** (float fHealthPoints) **PURE**
virtual HRESULT	**GetSurfaceInformation** (__out **SurfaceInfoV400** &SurfaceInfo, __in_opt const **FXYZ** *pvOffsetFeet) **PURE**
virtual HRESULT	**GetSurfaceElevation** (__out float &fElevationFeet, __in_opt const **FXYZ** *pvOffsetFeet) **PURE**
virtual HRESULT	**GetBathymetryElevation** (__out float &fDepthFeet, __in_opt const **FXYZ** *pvOffsetFeet) **PURE**
virtual HRESULT	**GetWeatherInformation** (__out **WeatherInfoV400** &WeatherInfo) **PURE**
virtual float	**GetMagneticVariation** () const **PURE**
virtual HRESULT	**VisualEffectOn** (__in __notnull LPCWSTR pszEffectName, __in_opt const **FXYZ** *pvOffsetFeet, __out void **ppEffect) **PURE**
virtual HRESULT	**VisualEffectOff** (__in __notnull void *pEffect) **PURE**
virtual HRESULT	**TriggerSound** (__in __notnull LPCWSTR pszName, **BOOL** bOn) **PURE**
virtual HRESULT	**TriggerContactSound** (__in __notnull LPCWSTR pszName, __in const **FXYZ** *pvOffset, float fImpactSpeed) **PURE**
virtual HRESULT	**StopSound** (__in __notnull LPCWSTR pszName) **PURE**
virtual HRESULT	**LoadServiceConstantData** (__in REFGUID guidService) **PURE**
virtual HRESULT	**UnloadServiceConstantData** (__in REFGUID guidService) **PURE**
virtual HRESULT	**CreateServiceInstance** (__in REFGUID guidService) **PURE**
virtual HRESULT	**DestroyServiceInstance** (__in REFGUID guidService) **PURE**
virtual HRESULT	**UpdateServiceInstance** (__in REFGUID guidService, double dDeltaT) **PURE**
virtual HRESULT	**GetTitle** (__out LPWSTR pszCfgTitle, __in unsigned int uLength) const **PURE**
virtual HRESULT	**GetCfgDir** (__out LPWSTR pszCfgDir, __in unsigned int uLength) const **PURE**
virtual HRESULT	**GetCfgFilePath** (__out LPWSTR pszCfgFile, __in unsigned int uLength) const **PURE**
virtual HRESULT	**GetCfgSectionName** (__out LPWSTR pszCfgFile, __in unsigned int uLength) const **PURE**
virtual HRESULT	**Destroy** () **PURE**
virtual HRESULT	**CheckCollision** (__in float fRadiusFeet, __out **COLLISIONTYPE** &eCollision, __out IUnknown **ppUnkHitObject) const **PURE**
virtual HRESULT	**CheckCollision** (__in float fRadiusFeet, __in const **DXYZ** *pdxyzPoints, __in **UINT32** uPointCount, __out **COLLISIONTYPE** &eCollision, __out IUnknown **ppUnkHitObject) const **PURE**
virtual **NET_MODE_TYPE**	**GetNetworkMode** () const **PURE**
virtual HRESULT	**AttachObject** (__in const **DXYZ** &vOffsetFeetParent, __in const **DXYZ** &vOffsetRadiansParent, __in UINT idChild, __in const **DXYZ** &vOffsetFeetChild, __in const **DXYZ** &vOffsetRadiansChild) **PURE**

virtual HRESULT	**AttachObject** (__in LPCSTR pszAttachPointName, __in const **DXYZ** &vOffsetRadiansParent, __in UINT idChild, __in const **DXYZ** &vOffsetFeetChild, __in const **DXYZ** &vOffsetRadiansChild) **PURE**
virtual HRESULT	**DetachObject** (__in UINT idChild) **PURE**
virtual HRESULT	**GetCategoryName** (__out LPWSTR pszCategoryName, __in unsigned int uLength) const **PURE**
virtual HRESULT	**GetCategoryId** (__out GUID &guidCategory) const **PURE**
virtual UINT	**GetDamageState** () const **PURE**
virtual void	**SetDamageState** (UINT uDamageState) **PURE**
virtual HRESULT	**GetBoundingBox** (__out **DXYZ** &dxyzMin, __out **DXYZ** &dxyzMax) const **PURE**
virtual HRESULT	**GetCrashTreeBox** (__in UINT index, __out **DXYZ** &dxyzMin, __out **DXYZ** &dxyzMax) const **PURE**
virtual UINT	**GetCrashTreeBoxCount** () const **PURE**

Member Function Documentation

§ AttachObject() [1/2]

virtual HRESULT AttachObject (__in const **DXYZ** & vOffsetFeetParent,
__in const **DXYZ** & vOffsetRadiansParent,
__in UINT idChild,
__in const **DXYZ** & vOffsetFeetChild,
__in const **DXYZ** & vOffsetRadiansChild
)

private virtual

Attaches the given object via offsets.

Parameters

vOffsetFeetParent	The offset in feet from the parent model center.
vOffsetRadiansParent	The orientation offset in radians from the parent model center.
idChild	The child object id to be attached to the parent.
vOffsetFeetChild	The offset in feet from the parent attach point.
vOffsetRadiansChild	The orientation offset in radians from the parent attach point.

Returns

S_OK if the objects were successfully attached, E_FAIL otherwise.

§ AttachObject() [2/2]

```
virtual HRESULT AttachObject ( __in LPCSTR          pszAttachPointName,
                               __in const DXYZ &    vOffsetRadiansParent,
                               __in UINT            idChild,
                               __in const DXYZ &    vOffsetFeetChild,
                               __in const DXYZ &    vOffsetRadiansChild
                             )                                              private virtual
```

Attaches the given object via attach point name and offsets.

Parameters

pszAttachPointName	The name of the parent attach point.
vOffsetRadiansParent	The orientation offset in radians from the parent attach point.
idChild	The child object id to be attached to the parent attach point.
vOffsetFeetChild	The offset in feet from the parent attach point.
vOffsetRadiansChild	The orientation offset in radians from the parent attach point.

Returns

S_OK if the objects were successfully attached, E_FAIL otherwise.

§ CheckCollision() [1/2]

```
virtual HRESULT CheckCollision ( __in float                 fRadiusFeet,
                                 __out COLLISIONTYPE &      eCollision,
                                 __out IUnknown **          ppUnkHitObject
                               )                            const
                                                                        private virtual
```

Checks if this object's center point is colliding with a building or another SimObject.

Parameters

fRadiusFeet	The radius around the object to check for collisions (feet).
eCollision	The resulting collision type.
ppUnkHitObject	The object the collision happened with.

Returns

S_OK if successful, E_FAIL otherwise. Note: A return of S_OK does not mean there is a collision, only that the query operation encountered no errors. The eCollision should be checked for a positive collision, and ppUnkHitObject for whether it involved an (IUnknown) object

§ CheckCollision() [2/2]

virtual HRESULT CheckCollision (__in float fRadiusFeet,
__in const **DXYZ** * pdxyzPoints,
__in **UINT32** uPointCount,
__out **COLLISIONTYPE** & eCollision,
__out IUnknown ** ppUnkHitObject
) const

private virtual

Checks if any of the given points are colliding with a building or another SimObject.

Parameters

fRadiusFeet	The radius around the object to check for collisions (feet).
pdxyzPoints	Body relative offset points used for collision detection (feet).
uPointCount	The number of points in pdxyzPoints.
eCollision	The resulting collision type.
ppUnkHitObject	The object the collision happened with.

Returns

S_OK if successful, E_FAIL otherwise. Note: A return of S_OK does not mean there is a collision, only that the query operation encountered no errors. The eCollision should be checked for a positive collision, and ppUnkHitObject for whether it involved an (IUnknown) object

§ CreateServiceInstance()

virtual HRESULT CreateServiceInstance (__in REFGUID guidService) private virtual

Invokes the instantiation of the service, based on the loaded constant data. This should be called from your SimObject's LoadDynamicData();

§ DecrementHealthPoints()

virtual HRESULT DecrementHealthPoints (__in float fDamagePoints) private virtual

Apply damage points. Positive points passed in will be decremented from current health points, to a limit of zero.

§ Destroy()

virtual HRESULT Destroy ( ) private virtual

Destroy self. This will not be immediate, so it can be called from within itself. It will be destroyed as soon as the current simulation finishes.

§ DestroyServiceInstance()

virtual HRESULT DestroyServiceInstance (__in REFGUID guidService) private virtual

Causes Prepar3D to destroy the instance of the service. This should be called from your SimObject's Deinit();

§ DetachObject()

virtual HRESULT DetachObject (__in UINT idChild) private virtual

Detaches the given object.

Parameters

idChild The child object id to be detached from the parent.

Returns

S_OK if the object was successfully detached, S_FALSE if the object to be removed was not attached to the parent, E_FAIL otherwise.

§ GetBathymetryElevation()

virtual HRESULT GetBathymetryElevation (__out float & fDepthFeet,
__in_opt const **FXYZ** * pvOffsetFeet
) private virtual

Provides current depth for the requested offset from the model center.

Parameters

fDepthFeet Reference to the depth variable (Feet)

pvOffsetFeet The offset from model enter. A value of NULL will use the model's center

§ GetBoundingBox()

virtual HRESULT GetBoundingBox (__out **DXYZ** & dxyzMin,
__out **DXYZ** & dxyzMax
) const private virtual

Gets the bounding box of the object.

Parameters

dxyzMin The minimum x, y, z values of the box in feet.

dxyzMax The maximum x, y, z values of the box in feet.

Returns

E_FAIL on failure, S_OK on success.

§ GetCategoryId()

virtual HRESULT GetCategoryId (__out GUID & guidCategory) const

private virtual

Returns the given simobject's category ID.

Parameters

guidCategory The category ID of the object

Returns

S_OK if the object's category ID was returned

§ GetCategoryName()

virtual HRESULT GetCategoryName (__out LPWSTR pszCategoryName,
__in unsigned int uLength
) const

private virtual

Returns the given simobject's category name.

Parameters

pszCategoryName The buffer to store the category name of the object

uLength The length of the buffer to store the category name in characters

Returns

S_OK if the object's category name was returned

§ GetCfgDir()

virtual HRESULT GetCfgDir (__out LPWSTR pszCfgDir,
__in unsigned int uLength
) const

private virtual

Returns a fully qualified path to the object's content path. This is generally the folder where the sim.cfg (or aircraft.cfg) lives.

§ GetCfgFilePath()

virtual HRESULT GetCfgFilePath (__out LPWSTR pszCfgFile,
__in unsigned int uLength
) const

private virtual

Returns a fully qualified path to the sim.cfg file.

§ GetCfgSectionName()

virtual HRESULT GetCfgSectionName (__out LPWSTR pszCfgFile,
__in unsigned int uLength
) const private virtual

Returns the relevant section name in the sim.cfg. e.g. [fltsim.1].

§ GetCrashTreeBox()

virtual HRESULT GetCrashTreeBox (__in UINT index,
__out **DXYZ** & dxyzMin,
__out **DXYZ** & dxyzMax
) const private virtual

Gets the crash tree boxes of the object via index.

Parameters

index The index of the crash tree box.

dxyzMin The minimum x, y, z values of the box in feet.

dxyzMax The maximum x, y, z values of the box in feet.

Returns

E_FAIL on failure, S_OK on success.

§ GetCrashTreeBoxCount()

virtual UINT GetCrashTreeBoxCount ( ) const private virtual

Gets the crash tree box count.

Returns

The number of crash tree boxes.

Remarks

Returns 0 when no crash tree is found.

§ GetDamageState()

virtual UINT GetDamageState ( ) const

private virtual

Gets the damage state

Returns

The damage state

Remarks

0 = No Damage, 1 = Light, 2 = Moderate, 3 = Destroyed, 4-n = User Defined

§ GetHealthPoints()

virtual float GetHealthPoints ( ) const

private virtual

Retrieves the current health of the object.

§ GetId()

virtual UINT GetId ( ) const

private virtual

The ID of this object: NOTE: "0" is an invalid id

§ GetMagneticVariation()

virtual float GetMagneticVariation ( ) const

private virtual

Returns the object's current magnetic variation in radians. A positive is value is "east".

§ GetMainMinMaxSimRates()

virtual HRESULT GetMainMinMaxSimRates (__out float & fMinHz,
__out float & fMaxHz
) const

private virtual

Provides the minimum and maximum main simulation rate (Hz). Typically the world position update rate. An ISimObject implementation will be called by this if it exists.

§ GetMissionId()

virtual HRESULT GetMissionId (__out GUID & guid) const private virtual

The guid ID of the object defined in an object file. NOTE: If the object is not spawned by a scenario, the return will be E_FAIL and the ID will be GUID_NULL.

§ GetMode()

virtual int GetMode ( ) const private virtual

Returns bitwise flags for the current modes of the SimObject.

§ GetNetworkMode()

virtual **NET_MODE_TYPE** GetNetworkMode ( ) const private virtual

Returns the current network mode for this object.

§ GetObjectGroupAssociationId()

virtual UINT GetObjectGroupAssociationId ( ) const private virtual

Group Association ID can be used to set/get IDs for Friend/Foe or other types of groupings.
Note: Group association is arbitrary. It could be used for things like alliances or squadrons. Default is 0, which signifies a neutral grouping.

§ GetPosition()

virtual HRESULT GetPosition (__out **DXYZ** & vLonAltLat,
__out **DXYZ** & vPHB,
__out **DXYZ** & vLonAltLatVel,
__out **DXYZ** & vPHBVel
) const private virtual

Gets the current world relative position and velocity from the Prepar3D-side SimObject. This will provide the valid state upon initialization, as well as when another system such as a UI element or slew changes the position.

Parameters

vLonAltLat	Longitude, altitude, latitude (radians)
vPHB	Pitch, heading, bank (radians)
vLonAltLatVel	Longitude, altitude, latitude velocity (feet / second)
vPHBVel	Pitch, heading, bank velocity (radians / second)

§ GetProperty() [1/11]

```
virtual HRESULT GetProperty ( __in int          iPropertyCode,
                              __in int          iUnitCode,
                              __out double &    dProperty,
                              __in int          index = 0
                            )                   const
```

private virtual

Get Property - Doubles

§ GetProperty() [2/11]

```
virtual HRESULT GetProperty ( __in LPCWSTR      pszPropertyName,
                              __in int          iUnitCode,
                              __out double &    dProperty,
                              __in int          index = 0
                            )                   const
```

private virtual

Get Property - Doubles

§ GetProperty() [3/11]

```
virtual HRESULT GetProperty ( __in LPCWSTR      pszPropertyName,
                              __in LPCWSTR      pszUnitCode,
                              __out double &    dProperty,
                              __in int          index = 0
                            )                   const
```

private virtual

Get Property - Doubles

§ GetProperty() [4/11]

```
virtual HRESULT GetProperty ( __in int          iPropertyCode,
                              __in int          iUnitCode,
                              __out DXYZ &      dProperty,
                              __in int          index = 0
                            )                   const
```

private virtual

Get Property - Vectors

§ GetProperty() [5/11]

```
virtual HRESULT GetProperty ( __in LPCWSTR  pszPropertyName,
                              __in int      iUnitCode,
                              __out DXYZ &  dProperty,
                              __in int      index = 0
                            )               const
```

private virtual

Get Property - Vectors

§ GetProperty() [6/11]

```
virtual HRESULT GetProperty ( __in LPCWSTR  pszPropertyName,
                              __in LPCWSTR  pszUnitCode,
                              __out DXYZ &  dProperty,
                              __in int      index = 0
                            )               const
```

private virtual

Get Property - Vectors

§ GetProperty() [7/11]

```
virtual HRESULT GetProperty ( __in int      iPropertyCode,
                              __out LPWSTR  pszProperty,
                              __in UINT     uLength,
                              __in int      index = 0
                            )               const
```

private virtual

Get Property - Strings

§ GetProperty() [8/11]

```
virtual HRESULT GetProperty ( __in LPCWSTR  pszPropertyName,
                              __out LPWSTR  pszProperty,
                              __in UINT     uLength,
                              __in int      index = 0
                            )               const
```

private virtual

Get Property - Strings

§ GetProperty() [9/11]

```
virtual HRESULT GetProperty ( __in int          iPropertyCode,
                              __in LPCWSTR      pszSecondarySubstring,
                              __in int          iUnitCode,
                              __out double &    dProperty,
                              __in int          index = 0
                            )                   const
```

private virtual

Get Property - Doubles (with secondary substring input)

§ GetProperty() [10/11]

```
virtual HRESULT GetProperty ( __in LPCWSTR      pszPropertyName,
                              __in LPCWSTR      pszSecondarySubstring,
                              __in int          iUnitCode,
                              __out double &    dProperty,
                              __in int          index = 0
                            )                   const
```

private virtual

Get Property - Doubles (with secondary substring input)

§ GetProperty() [11/11]

```
virtual HRESULT GetProperty ( __in LPCWSTR      pszPropertyName,
                              __in LPCWSTR      pszSecondarySubstring,
                              __in LPCWSTR      pszUnitCode,
                              __out double &    dProperty,
                              __in int          index = 0
                            )                   const
```

private virtual

Get Property - Doubles (with secondary substring input)

§ GetPropertyCodeAndIndex()

```
virtual HRESULT GetPropertyCodeAndIndex ( __in PROPERTY_TYPE  eType,
                                          __in LPCWSTR        pszPropertyName,
                                          __out int &         iPropertyCode,
                                          __inout int &       iIndex
                                        )                     const
```

private virtual

Get Properties

§ GetSurfaceElevation()

```
virtual HRESULT GetSurfaceElevation ( __out float &              fElevationFeet,
                                      __in_opt const FXYZ *      pvOffsetFeet
                                    )
```

private virtual

Provides current surface elevation (above Mean Sea Level) for the requested offset from the model center. This will be more efficient than GetSurfaceInformation when only the elevation is needed. A return value of E_FAIL means that Prepar3D's terrain system failed to process the request properly. This could happen if it is not initialized fully.

Parameters

fElevationFeet Reference to the elevation variable (Feet)

pvOffsetFeet The offset from model enter. A value of NULL will use the model's center

§ GetSurfaceInformation()

```
virtual HRESULT GetSurfaceInformation ( __out SurfaceInfoV400 &  SurfaceInfo,
                                        __in_opt const FXYZ *    pvOffsetFeet
                                      )
```

private virtual

Provides current surface information for the requested offset from the model center. See the **ISimObject.h** for the definition of the SurfaceInfo data structure. A return value of E_FAIL means that Prepar3D's terrain system failed to process the request properly. This could happen if it is not initialized fully.

Parameters

SurfaceInfo Reference to local SurfaceInfo data structure

pvOffsetFeet The offset from model enter.A value of NULL will use the model's center

§ GetSystemHealth()

```
virtual float GetSystemHealth ( HANDLE  hSystem ) const
```

private virtual

Returns the health percentage (0.0 - 1.0) for a given malfunction.

§ GetTitle()

```
virtual HRESULT GetTitle ( __out LPWSTR    pszCfgTitle,
                           __in unsigned int  uLength
                         )                    const
```

private virtual

Returns the unique string that identifies this object.

§ GetWeatherInformation()

virtual HRESULT GetWeatherInformation (__out **WeatherInfoV400** & WeatherInfo) private virtual

Provides current weather information for the object's current position. See the **ISimObject.h** for the definition of the WeatherInfo data structure. A return value of E_FAIL means that Prepar3D's weather system failed to process the request properly. This could happen if it is not initialized fully.

Parameters

WeatherInfo Reference to local WeatherInfo data structure

§ InitPosition()

```
virtual HRESULT InitPosition ( __in const DXYZ *  pvLonAltLat,
                               __in const DXYZ *  pvPHB,
                               __in const DXYZ *  pvLonAltLatVel,
                               __in const DXYZ *  pvPHBVel,
                               __in BOOL          bSetOnGround
                             )
```

private virtual

Sets the current world relative position and velocity to the Prepar3D-side SimObject. All parameters are optional. Any parameter set to NULL will be ignored, and current object values will be retained.

Parameters

pvLonAltLat	Longitude, altitude, latitude (radians)
pvPHB	Pitch, heading, bank (radians)
pvLonAltLatVel	Longitude, altitude, latitude velocity (feet / second)
pvPHBVel	Pitch, heading, bank velocity (radians / second)
bSetOnGround	Flag indicating if the object is to be set on the ground, in which case the on-ground height and pitch attitude will be set.

§ InObjectFoeList()

virtual **BOOL** InObjectFoeList (UINT id) const private virtual

Group ObjectFoeList Hosts a list of ID's that are considered foe's to the current entity

§ InObjectFriendList()

virtual **BOOL** InObjectFriendList (UINT id) const private virtual

Group ObjectFoeList Hosts a list of ID's that are considered friends's to the current entity

§ IsOnGround()

virtual **BOOL** IsOnGround ( ) const

private virtual

IsOnGround Returns the on-ground flag value currently in the core base object. This can be useful for determining if the object has been placed on the ground through a non-simulated means such as the UI.

§ IsUser()

virtual **BOOL** IsUser ( ) const

private virtual

Returns if the object is the user or not.

§ LoadServiceConstantData()

virtual HRESULT LoadServiceConstantData (__in REFGUID guidService)

private virtual

Invokes the loading of the relevant constant data. This should be called from your SimObject's LoadConstantData();

§ RegisterProperty() [1/6]

virtual HRESULT RegisterProperty (__in LPCWSTR pszPropertyName,
__in LPCWSTR pszPropertyBaseUnits,
__in __notnull **PPropertyCallback** pcbProperty
)

private virtual

Registers a simulation property for this instance of the object. This differs from the RegisterProperty method in the ISimObjectManager interface which registers a property that exists for all instances associated with the given guid. Properties registered here, as well as their ID, exist only for the lifetime of this object. Otherwise they function the same when used for gauges, animations, and missions.

Note
Double "get"

§ RegisterProperty() [2/6]

virtual HRESULT RegisterProperty (__in LPCWSTR pszPropertyName,
__in LPCWSTR pszPropertyBaseUnits,
__in __notnull **PEventCallback** pcbEvent,
__in **EVENTTYPE** eType
)

private virtual

Note
Double "set"

§ RegisterProperty() [3/6]

```
virtual HRESULT
RegisterProperty ( __in LPCWSTR                          pszPropertyName,
                   __in LPCWSTR                          pszPropertyBaseUnits,
                   __in __notnull PPropertyVectorCallback pcbProperty
                 )
```

private virtual

Note
Vector "get"

§ RegisterProperty() [4/6]

```
virtual HRESULT RegisterProperty ( __in LPCWSTR  pszPropertyName,
                                   __in LPCWSTR  pszPropertyBaseUnits,
                                   __in __notnull PEventVectorCallback
                                 )
```

private virtual

Note
Vector "set"

§ RegisterProperty() [5/6]

```
virtual HRESULT RegisterProperty ( __in LPCWSTR                  pszPropertyName,
                                   __in __notnull PPropertyStringCallback pcbProperty
                                 )
```

private virtual

Note
String "get"

§ RegisterProperty() [6/6]

```
virtual HRESULT RegisterProperty ( __in LPCWSTR  pszPropertyName,
                                   __in __notnull PEventStringCallback
                                 )
```

private virtual

Note
String "set"

§ RegisterService()

virtual HRESULT RegisterService (__in REFGUID guidService,
__in __notnull IUnknown * punkService
) private virtual

Registers a service that can be queried for on this object. A service should be an IUnknown-derived object and registered with a unique GUID.

Parameters

guidService Unique GUID to identify this service.

punkService Reference to an instance of this service.

§ RegisterSimulation() [1/2]

virtual HRESULT RegisterSimulation (__in __notnull ISimulation * pSimulation,
float fRateHz
) private virtual

Registers an ISimulation callback for real-time updates (discussed in Creating Behaviors.) ISimulation registration will be locked after the ISimObject Init() function has been called. All ISimulation objects must be registered before this point.

Parameters

pSimulation Address of simulation system

fRateHz Specified iteration rate

§ RegisterSimulation() [2/2]

virtual HRESULT RegisterSimulation (__in __notnull ISimulation * pSimulation,
float fMinRateHz,
float fMaxRateHz
) private virtual

Registers an ISimulation callback for real-time updates (discussed in Creating Behaviors.) ISimulation registration will be locked after the ISimObject Init() function has been called. All ISimulation objects must be registered before this point.

Parameters

pSimulation Address of simulation system

fMinRateHz Specified minimum iteration rate

fMaxRateHz Specified maximum iteration rate

§ RegisterSystemMalfunction()

```
virtual HANDLE RegisterSystemMalfunction ( __in REFGUID   guidMalfunction,
                                           __in LPCWSTR   pszType,
                                           __in LPCWSTR   pszBaseName,
                                           __in LPCWSTR   pszInstanceName,
                                           __in int       nSubIndex
                                         )
```

private virtual

Registers a specific malfunction that can be set through the UI, scenarios, or missions.

Parameters

guidMalfunction	Unique malfunction ID
pszType	UI Type. Choices: Instruments, Systems, Radios, Engines, Controls, Structural, Miscellaneous
pszBaseName	Name used for mission file reference. Should be generic (no index), such as "Engine"
pszInstanceName	Specific malfunction name for the UI, such as "Total Failure Engine 1"
nSubIndex	Sub-index. For example, engine 0, 1, etc...

§ RotateBodyToWorld()

```
virtual HRESULT RotateBodyToWorld ( __in const DXYZ &  vBody,
                                    __out DXYZ &       vWorld
                                  )                    const
```

private virtual

Rotates a vector from the body frame of reference to the world frame of reference.

§ RotateWorldToBody()

```
virtual HRESULT RotateWorldToBody ( __in const DXYZ &  vWorld,
                                    __out DXYZ &       vBody
                                  )                    const
```

private virtual

Rotates a vector from the world frame of reference to the body frame of reference.

§ SetCrashMode()

```
virtual HRESULT SetCrashMode ( double  dDeltaT )
```

private virtual

Should be called when it is desired to put Prepar3D into "crash" mode. By default, the application will go through it's crash cycle and reset. It is the developer's responsibility to program the behavior of the object when crash in crash mode.

§ SetDamageState()

virtual void SetDamageState (UINT uDamageState) private virtual

Sets the damage state

Parameters

eDamageState The damage state

Remarks

0 = No Damage, 1 = Light, 2 = Moderate, 3 = Destroyed, 4-n = User Defined

§ SetHealthPoints()

virtual void SetHealthPoints (float fHealthPoints) private virtual

Sets current health of the object.

§ SetMainMinMaxSimRates()

virtual HRESULT SetMainMinMaxSimRates (__in float fMinHz,
__in float fMaxHz
) private virtual

Sets minimum and maximum main simulation rate (Hz). Typically the world position update rate. This is typically called if there is a desire to synchronize the rates of two or more objects. For example an aircraft and an aircraft carrier, or to prevent perceived jitter when viewing an object from a camera attached to another object. An ISimObject implementation will be called by this if it exists.

§ SetObjectFoeList()

virtual void SetObjectFoeList (UINT * uEnteredFoeID,
UINT size
) private virtual

Group Association ID can be used to set/get IDs for Friend/Foe or other types of groupings.

§ SetObjectFriendList()

virtual void SetObjectFriendList (UINT * uEnteredFriendID,
UINT size
) private virtual

Group Association ID can be used to set/get IDs for Friend/Foe or other types of groupings.

§ SetObjectGroupAssociationId()

virtual void SetObjectGroupAssociationId (UINT uAssociationId) private virtual

Group Association ID can be used to set/get IDs for Friend/Foe or other types of groupings.
Note: Group association is arbitrary. It could be used for things like alliances or squadrons. Default is 0, which signifies a neutral grouping.

§ SetPosition()

virtual HRESULT SetPosition (__in const **DXYZ** & vLonAltLat,
__in const **DXYZ** & vPHB,
__in const **DXYZ** & vLonAltLatVel,
__in const **DXYZ** & vPHBVel,
__in **BOOL** bIsOnGround,
__in double dDeltaT
) private virtual

Sets the current world relative position and velocity to the Prepar3D-side SimObject.

Parameters

vLonAltLat	Longitude, altitude, latitude (radians)
vPHB	Pitch, heading, bank (radians)
vLonAltLatVel	Longitude, altitude, latitude velocity (feet / second)
vPHBVel	Pitch, heading, bank velocity (radians / second)
bIsOnGround	Flag indicating if the object is on the ground. This is important during terrain updates.
dDeltaT	The time between object updates used to track how much time has accumulated between camera frames

§ StopSound()

virtual HRESULT StopSound (__in __notnull LPCWSTR pszName) private virtual

Stops a sound configured in the object's sound.cfg. This function will stop a looping or a one shot sound.

Parameters

pszName Sound reference name from Sound.cfg

§ TriggerContactSound()

virtual HRESULT TriggerContactSound (__in __notnull LPCWSTR pszName,
__in const **FXYZ** * pvOffset,
float fImpactSpeed
) private virtual

Triggers a sound specifically for a ground contact point.

Parameters

pszName	Sound reference name from Sound.cfg
pvOffset,The	offset from model center. A value of NULL will use the model's center
fImpactSpeed	Speed used by sound system to scale sound

§ TriggerProperty() [1/11]

virtual HRESULT TriggerProperty (__in int iPropertyCode,
__in int iUnitCode,
__in double dData,
__in int index
) const private virtual

Numeric trigger

§ TriggerProperty() [2/11]

virtual HRESULT TriggerProperty (__in LPCWSTR pszPropertyName,
__in int iUnitCode,
__in double dData,
__in int index
) const private virtual

Numeric trigger

§ TriggerProperty() [3/11]

virtual HRESULT TriggerProperty (__in LPCWSTR pszPropertyName,
__in LPCWSTR pszUnitCode,
__in double dData,
__in int index
) const private virtual

Numeric trigger

§ TriggerProperty() [4/11]

```
virtual HRESULT TriggerProperty ( __in int          iPropertyCode,
                                  __in int          iUnitCode,
                                  __in const DXYZ & vData,
                                  __in int          index
                                ) const
```

private virtual

Vector trigger

§ TriggerProperty() [5/11]

```
virtual HRESULT TriggerProperty ( __in LPCWSTR      pszPropertyName,
                                  __in int          iUnitCode,
                                  __in const DXYZ & vData,
                                  __in int          index
                                ) const
```

private virtual

Vector trigger

§ TriggerProperty() [6/11]

```
virtual HRESULT TriggerProperty ( __in LPCWSTR      pszPropertyName,
                                  __in LPCWSTR      pszUnitCode,
                                  __in const DXYZ & vData,
                                  __in int          index
                                ) const
```

private virtual

Vector trigger

§ TriggerProperty() [7/11]

```
virtual HRESULT TriggerProperty ( __in int          iPropertyCode,
                                  __in LPCWSTR      pszData,
                                  __in int          index
                                ) const
```

private virtual

String strigger

§ TriggerProperty() [8/11]

```
virtual HRESULT TriggerProperty ( __in LPCWSTR pszPropertyName,
                                  __in LPCWSTR pszData,
                                  __in int     index
                                ) const
```

private virtual

String strigger

§ TriggerProperty() [9/11]

```
virtual HRESULT TriggerProperty ( __in int     iPropertyCode,
                                  __in LPCWSTR pszSecondarySubstring,
                                  __in int     iUnitCode,
                                  __in double  dData,
                                  __in int     index
                                ) const
```

private virtual

Trigger - Doubles (with secondary substring input)

§ TriggerProperty() [10/11]

```
virtual HRESULT TriggerProperty ( __in LPCWSTR pszPropertyName,
                                  __in LPCWSTR pszSecondarySubstring,
                                  __in int     iUnitCode,
                                  __in double  dData,
                                  __in int     index
                                ) const
```

private virtual

Trigger - Doubles (with secondary substring input)

§ TriggerProperty() [11/11]

```
virtual HRESULT TriggerProperty ( __in LPCWSTR pszPropertyName,
                                  __in LPCWSTR pszSecondarySubstring,
                                  __in LPCWSTR pszUnitCode,
                                  __in double  dData,
                                  __in int     index
                                ) const
```

private virtual

Trigger - Doubles (with secondary substring input)

§ TriggerSound()

virtual HRESULT TriggerSound (__in __notnull LPCWSTR pszName,
BOOL bOn
) private virtual

Triggers a sound configured in the object's sound.cfg.

Parameters

pszName Sound reference name from Sound.cfg

bOn Turns on/off a looping sound. This value has no effect on one shot sounds.

Remarks

To turn off a one shot sound, use the StopSound function.

§ UnloadServiceConstantData()

virtual HRESULT UnloadServiceConstantData (__in REFGUID guidService) private virtual

Causes Prepar3D to unload the relevant constant data. This should be called from your SimObject's UnLoadConstantData();

§ UnregisterService()

virtual HRESULT UnregisterService (__in REFGUID guidService) private virtual

Removes a service that has been register with **RegisterService()**.

Parameters

guidService Unique GUID to identify this service.

§ UpdateServiceInstance()

virtual HRESULT UpdateServiceInstance (__in REFGUID guidService,
double dDeltaT
) private virtual

The real-time update of the service. Your SimObject is responsible for calling it with an accurate delta time.

§ VisualEffectOff()

virtual HRESULT VisualEffectOff (__in __notnull void * pEffect) private virtual

Turns a visual effect off.

Parameters

pEffect Reference pointer obtained in out parameter of **VisualEffectOn()**

§ VisualEffectOn()

virtual HRESULT VisualEffectOn (__in __notnull LPCWSTR pszEffectName,
__in_opt const **FXYZ** * pvOffsetFeet,
__out void ** ppEffect
) private virtual

Turns a visual effect on. The out parameter allows you to hold a reference to a visual effect to subsequently turn off.

Parameters

pszEffectName File name for requested visual effect

pvOffsetFeet The offset from model center. A value of NULL will use the model's center

ppEffect Reference pointer for turning the visual effect off with **VisualEffectOff()**

§ P3D::ISubSystemFactoryV440

class P3D::ISubSystemFactoryV440

Factory class interface for creating supplemental subsystems on an existing simobject implementation

Inherits IUnknown.

Private Member Functions

virtual HRESULT **Create** (__in IBaseObjectV400 *pBaseObject, __in LPCWSTR pszSecondaryData) **PURE**

Member Function Documentation

§ Create()

virtual HRESULT Create (__in IBaseObjectV400 * pBaseObject,
__in LPCWSTR pszSecondaryData
) private virtual

Creates a new subsystem during object loading

Parameters

pBaseObject The object on which the subsytem is being attached

pszSecondaryData (optional) This allows unique subsystems to be specified using the same factory.

Remarks

The SubSystemFactory should be registered through the **IPdk** interface at DLL load time.

Supplemental subsystems can be specifed in the aircraft.cfg/sim.cfg file. For example:

[SupplementalSystems]

System.0 = {bc95b363-1d22-42aa-82b1-f10905b22c40}, Engine

System.1 = {bc95b363-1d22-42aa-82b1-f10905b22c40}, Propeller

where the guid is the registered Servide ID (SID)

§ P3D::WorldConstants

class P3D::WorldConstants

Constant values describing the Earth atmosphere and geometry

Note

While this is accessed through IBaseObject, these values will be constant for all simobjects, and a single static copy could be shared across multiple instances.

Class Members		
double	m_dEquatorialRadius	Feet
double	m_dPolarRadius	Feet
float	m_fGravitySeaLevel	ft/s^2
float	m_fSpecificGasConstant	R
float	m_fSpecificHeatRatio	Gamma (Cp/Cv for air (specific heat ratio)
float	m_fStandardSeaLevelDensity	slugs/ft^3
float	m_fStandardSeaLevelPressure	Lbs/SqFt
float	m_fStandardSeaLevelTemperature	Rankine

§ P3D::SurfaceInfoV400

class P3D::SurfaceInfoV400

Contains terrain/surface information for a given object's offset

Class Members		
	SURFACE_CONDITION_CATEGORY	Surface conditions
	SURFACE_TYPE_CATEGORY	Surface categories

Class Members		
BOOL	m_bOnPlatform	TRUE or FALSE
SURFACE_TYPE_CATEGORY	m_eSurfaceCategory	SURFACE_TYPE_CATEGORY
SURFACE_CONDITION_CATEGORY	m_eSurfaceCondition	SURFACE_CONDITION_CATEGORY
float	m_fElevation	Feet
float	m_fWaveHeight	Feet
FXYZ	m_vNormal	Unit vector
FXYZ	m_vRotVelocity	Radians per Second
FXYZ	m_vVelocity	Feet per Second

§ P3D::WeatherInfoV400

class P3D::WeatherInfoV400

Contains terrain/surface information for a given object's offset

Class Members		
BOOL	m_bIsInCloud	TRUE or FALSE.
BOOL	m_bIsRaining	TRUE or FALSE.
BOOL	m_bIsSnowing	TRUE or FALSE.
BOOL	m_bUnsteadyWind	TRUE if gusting.
float	m_fAmbientPressure	PSF.
float	m_fIcingIntensityPercent	0.0 = none, 1.0 = max
float	m_fPrecipIntensityPercent	0.0 = min, 1.0 = max
float	m_fSeaLevelPressure	PSF.
float	m_fTemperature	Degrees Rankine.
float	m_fTurbulencePercent	0.0 = none, 1.0 = max
float	m_fVisibility	Feet.
float	m_fWindDirection	Radians True.
float	m_fWindSpeed	Feet per Second.
FXYZ	m_vWind	Feet per Second.

§ P3D::IMassPropertiesV01

class P3D::IMassPropertiesV01

Interface for getting mass properties from SimObject implementation

Inherits IMassProperties.

Private Member Functions

virtual float	**GetWeight** () const **PURE**
virtual **BOOL**	**RegisterMass** (__in const IMassElement *pElement) **PURE**
virtual **BOOL**	**UnRegisterMass** (__in const IMassElement *pElement) **PURE**
virtual void	**ForceUpdate** () **PURE**

Member Function Documentation

§ ForceUpdate()

virtual void ForceUpdate ( ) private virtual

Force an update of properties

§ GetWeight()

virtual float GetWeight ( ) const private virtual

Returns weight of SimObject in pounds.

§ RegisterMass()

virtual **BOOL** RegisterMass (__in const IMassElement * pElement) private virtual

Registers mass of SimObject.

§ UnRegisterMass()

virtual **BOOL** UnRegisterMass (__in const IMassElement * pElement) private virtual

Unregisters mass of SimObject

§ P3D::IForceMomentsV01

class P3D::IForceMomentsV01

Interface for getting physical forces from a SimObject implementation.

Inherits IForceMoments.

Private Member Functions

virtual **BOOL**	**RegisterElement** (__in IForceElement *pElement) **PURE**
virtual **BOOL**	**UnRegisterElement** (__in IForceElement *pElement) **PURE**
virtual UINT	**ElementCount** () const **PURE**
virtual IForceElement *	**GetElement** (int index) const **PURE**

Member Function Documentation

§ ElementCount()

virtual UINT ElementCount ( ) const [private] [virtual]

Returns the total number of force elements

§ GetElement()

virtual IForceElement* GetElement (int index) const [private] [virtual]

Gets a force element

§ RegisterElement()

virtual **BOOL** RegisterElement (__in IForceElement * pElement) [private] [virtual]

Registers a force element

§ UnRegisterElement()

virtual **BOOL** UnRegisterElement (__in IForceElement * pElement) [private] [virtual]

Unregisters a force element

§ P3D::ICollisionServiceV01

class P3D::ICollisionServiceV01

Interface for getting crash parameters for this object

Inherits ICollisionService.

Private Member Functions

virtual **BOOL**	**InvokesCrashOnOtherObjects** () **PURE**
virtual void	**SetInvokesCrashOnOtherObjects** (__in **BOOL** invokesCrash) **PURE**

Member Function Documentation

§ InvokesCrashOnOtherObjects()

virtual **BOOL** InvokesCrashOnOtherObjects ( ) private virtual

Whether crash will invoke on other objects

§ SetInvokesCrashOnOtherObjects()

virtual void SetInvokesCrashOnOtherObjects (__in **BOOL** invokesCrash) private virtual

Sets whether to invoke crash on other objects

§ P3D::IAircraftServiceV01

class P3D::IAircraftServiceV01

Inherits IAircraftService.

Inherited by **IAirplaneServiceV01**, and **IRotorcraftServiceV01**.

Private Member Functions

virtual float **GetIndicatedAirspeed** () const **PURE**

Member Function Documentation

§ GetIndicatedAirspeed()

virtual float GetIndicatedAirspeed ( ) const private virtual

Gets the indicated airspeed. (feet per second)

§ P3D::IAirplaneServiceV01

class P3D::IAirplaneServiceV01

Inherits **IAircraftServiceV01**.

§ P3D::IRotorcraftServiceV01

class P3D::IRotorcraftServiceV01

Inherits **IAircraftServiceV01**.

§ P3D::IBoatServiceV01

class P3D::IBoatServiceV01

Inherits IBoatService.

§ P3D::IGroundVehicleServiceV01

class P3D::IGroundVehicleServiceV01

Inherits IGroundVehicleService.

§ P3D::IAtcServiceV01

class P3D::IAtcServiceV01

Interface for getting ATC parameters from this object

Inherits IAtcService.

§ P3D::IRadarSignatureServiceV01

class P3D::IRadarSignatureServiceV01

Interface for getting the radar signature of this object

Inherits IRadarSignatureService.

§ P3D::IDoorServiceV01

class P3D::IDoorServiceV01

Interface for getting door parameters for this object

Inherits IDoorService.

Private Member Functions

virtual float **GetDoorPercentOpen** (__in int doorIndex) const **PURE**

Member Function Documentation

§ GetDoorPercentOpen()

virtual float GetDoorPercentOpen (__in int doorIndex) const private virtual

Returns percentage of how open a door currently is

Parameters

doorIndex Index that associated with a door

§ P3D::IFuelServiceV400

class P3D::IFuelServiceV400

Interface for setting/getting fuel levels, such as in the UI

Inherits IUnknown.

Private Member Functions

virtual float	**GetWeightPerGallon** () const **PURE**
virtual UINT	**GetNumberOfTanks** () const **PURE**
virtual float	**GetTotalCapacityGallons** () const **PURE**
virtual float	**GetTotalLevelPercent** () const **PURE**
virtual HRESULT	**SetTotalLevelPercent** (float fPct) **PURE**
virtual HRESULT	**GetTankName** (__in int iTankIndex, __out LPWSTR pszName, __in UINT uLength) const **PURE**
virtual float	**GetTankCapacityGallons** (__in int iTankIndex) const **PURE**
virtual float	**GetTankLevelPercent** (__in int iTankIndex) const **PURE**
virtual HRESULT	**SetTankLevelPercent** (__in int iTankIndex, float fPct) **PURE**

Member Function Documentation

§ GetNumberOfTanks()

virtual UINT GetNumberOfTanks ( ) const [private] [virtual]

Number of tanks with capacity greater than zero.

§ GetTankCapacityGallons()

virtual float GetTankCapacityGallons (__in int iTankIndex) const [private] [virtual]

Gallons for specified tank.

§ GetTankLevelPercent()

virtual float GetTankLevelPercent (__in int iTankIndex) const [private] [virtual]

Percentage (0 - 1) for specified tank.

§ GetTankName()

virtual HRESULT GetTankName (__in int iTankIndex,
__out LPWSTR pszName,
__in UINT uLength
) const

private virtual

Returns string name used in the Fuel User Interface for specified tank.

§ GetTotalCapacityGallons()

virtual float GetTotalCapacityGallons ( ) const

private virtual

Total capacity for all tanks

§ GetTotalLevelPercent()

virtual float GetTotalLevelPercent ( ) const

private virtual

Total percentage (0-1) for all tanks combined.

§ GetWeightPerGallon()

virtual float GetWeightPerGallon ( ) const

private virtual

Pounds per gallon

§ SetTankLevelPercent()

virtual HRESULT SetTankLevelPercent (__in int iTankIndex,
float fPct
)

private virtual

Sets the percentage (0 - 1) for specified tank.

§ SetTotalLevelPercent()

virtual HRESULT SetTotalLevelPercent (float fPct)

private virtual

Sets total percentage (0-1) for all tanks combined.

§ P3D::ISurfaceQueryManagerV400

class P3D::ISurfaceQueryManagerV400

General surface queries, not associated with a specific object. This service interface is available from the PDK.

Inherits IUnknown.

Private Member Functions

virtual HRESULT	**QuerySurfaceInformation** (__out **SurfaceInfoV400** &SurfaceInfo, __in const **DXYZ** &vWorldPosRadiansFeet) const **PURE**
virtual HRESULT	**QuerySurfaceElevation** (__out float &fElevationFeet, __in const **DXYZ** &vWorldPosRadiansFeet) const **PURE**
virtual HRESULT	**QueryBathymetryElevation** (__out float &fDepthFeet, __in const **DXYZ** &vWorldPosRadiansFeet) const **PURE**

Member Function Documentation

§ QueryBathymetryElevation()

virtual HRESULT QueryBathymetryElevation (__out float & fDepthFeet,
__in const **DXYZ** & vWorldPosRadiansFeet
) const [private] [virtual]

Provides bathymetry elevation (in feet) for a given world-relative position. (X = Longitude in radians, Y = Altitude in feet, Z = Latitude in radians). Returns E_FAIL if query fails.

§ QuerySurfaceElevation()

virtual HRESULT QuerySurfaceElevation (__out float & fElevationFeet,
__in const **DXYZ** & vWorldPosRadiansFeet
) const [private] [virtual]

Provides surface elevation (in feet) for a given world-relative position. (X = Longitude in radians, Y = Altitude in feet, Z = Latitude in radians). Returns E_FAIL if query fails.

§ QuerySurfaceInformation()

virtual HRESULT QuerySurfaceInformation (__out **SurfaceInfoV400** & SurfaceInfo,
__in const **DXYZ** & vWorldPosRadiansFeet
) const [private] [virtual]

Provides surface information for a given world-relative position. (X = Longitude in radians, Y = Altitude in feet, Z = Latitude in radians). Returns E_FAIL if query fails.

§ P3D::IWaypointQueryManagerV400

class P3D::IWaypointQueryManagerV400

Interface that provides waypoint information specifically for the user in a mission. This service interface is available from the PDK.

Inherits IUnknown.

Private Member Functions

virtual UINT	**GetNumberOfWaypointLists** () const **PURE**
virtual HRESULT	**GetWaypointListIndexFromDescription** (__in LPCWSTR pszDescription, __out UINT &iWaypointList) const **PURE**
virtual UINT	**GetNumberOfWaypoints** (__in UINT iWaypointList) const **PURE**
virtual HRESULT	**GetWaypointListDescription** (__in UINT iWaypointList, __out LPWSTR pszDesc, __in UINT uLength) const **PURE**
virtual int	**GetWaypointID** (__in UINT iWaypointList, __in UINT iWaypoint) const **PURE**
virtual HRESULT	**GetWaypointDescription** (__in UINT iWaypointList, __in UINT iWaypoint, __out LPWSTR pszDescription, __in UINT uLength) const **PURE**
virtual HRESULT	**GetWaypointPosition** (__in UINT iWaypointList, __in UINT iWaypoint, __out **P3D::DXYZ** &vWorldPosition) const **PURE**
virtual HRESULT	**GetWaypointOrientation** (__in UINT iWaypointList, __in UINT iWaypoint, __out **P3D::DXYZ** &vOrientation) const **PURE**
virtual **BOOL**	**IsAltitudeAGL** (__in UINT iWaypointList, __in UINT iWaypoint) const **PURE**
virtual HRESULT	**GetWaypointCustomValue** (__in UINT iWaypointList, __in UINT iWaypoint, __out LPWSTR pszCustomVal, __in UINT uLength) const **PURE**

Member Function Documentation

§ GetNumberOfWaypointLists()

virtual UINT GetNumberOfWaypointLists ( ) const [private] [virtual]

Returns the number of waypoint lists the manager is holding.

§ GetNumberOfWaypoints()

virtual UINT GetNumberOfWaypoints (__in UINT iWaypointList) const [private] [virtual]

Returns the number of waypoints for the list specified by the list index input.

§ GetWaypointCustomValue()

```
virtual HRESULT GetWaypointCustomValue ( __in UINT    iWaypointList,
                                         __in UINT    iWaypoint,
                                         __out LPWSTR pszCustomVal,
                                         __in UINT    uLength
                                       )              const
```

private virtual

Provides the text string specified in the Custom Value field for the waypoint specified by the list and waypoint index inputs. Returns E_FAIL if waypoint not found.

§ GetWaypointDescription()

```
virtual HRESULT GetWaypointDescription ( __in UINT    iWaypointList,
                                         __in UINT    iWaypoint,
                                         __out LPWSTR pszDescription,
                                         __in UINT    uLength
                                       )              const
```

private virtual

Returns the specified description of the waypoint.

§ GetWaypointID()

```
virtual int GetWaypointID ( __in UINT iWaypointList,
                            __in UINT iWaypoint
                          )           const
```

private virtual

Returns the specified integer ID of the waypoint.

§ GetWaypointListDescription()

```
virtual HRESULT GetWaypointListDescription ( __in UINT    iWaypointList,
                                             __out LPWSTR pszDesc,
                                             __in UINT    uLength
                                           )              const
```

private virtual

Provides the description string for the list specified by the list index input. Returns E_FAIL if waypoint list not found.

§ GetWaypointListIndexFromDescription()

```
virtual HRESULT GetWaypointListIndexFromDescription ( __in LPCWSTR  pszDescription,
                                                      __out UINT &  iWaypointList
                                                    )               const
```

private virtual

Provides the index (0 - based) of the waypoint list with the description string passed in. Returns E_FAIL if waypoint list not found.

§ GetWaypointOrientation()

```
virtual HRESULT GetWaypointOrientation ( __in UINT              iWaypointList,
                                         __in UINT              iWaypoint,
                                         __out P3D::DXYZ &      vOrientation
                                       )                        const
```

private virtual

Provides the orientation for the waypoint specified by the list and waypoint index inputs. (X = Pitch in radians, Y = Heading in radians, Z = Bank in radians). Returns E_FAIL if waypoint not found.

§ GetWaypointPosition()

```
virtual HRESULT GetWaypointPosition ( __in UINT              iWaypointList,
                                      __in UINT              iWaypoint,
                                      __out P3D::DXYZ &      vWorldPosition
                                    )                        const
```

private virtual

Provides the world-relative position for the waypoint specified by the list and waypoint index inputs. (X = Longitude in radians, Y = Altitude in feet, Z = Latitude in radians). Returns E_FAIL if waypoint not found.

§ IsAltitudeAGL()

```
virtual BOOL IsAltitudeAGL ( __in UINT  iWaypointList,
                             __in UINT  iWaypoint
                           )            const
```

private virtual

Returns whether the specified altitude of the waypoint is Above Ground Level or Mean Sea Level.

§ P3D::IAvatarSimV01

class P3D::IAvatarSimV01

This interface is to be implemented on any simulation object that is used for the user's avatar.

Inherits IUnknown.

Private Member Functions

virtual HRESULT	**OnAttach** () **PURE**
virtual HRESULT	**OnDetach** () **PURE**

Member Function Documentation

§ OnAttach()

virtual HRESULT OnAttach ( ) private virtual

Called when the avatar attaches to the user's object being controlled.

§ OnDetach()

virtual HRESULT OnDetach ( ) private virtual

Called when the avatar detaches to the user's object being controlled.

§ P3D::IAnimationControllerV01

class P3D::IAnimationControllerV01

This service can queried for on a simulation object to play pre-defined animations built into the 3-D visual model. For example, a walking animation.

Inherits IUnknown.

Private Member Functions

virtual HRESULT	**Play** (const GUID &guidAnimationID, **BOOL** bLoop) **PURE**
virtual HRESULT	**TransitionAndPlay** (const GUID &guidCurrentAnimationID, const GUID &guidNextAnimationID, **BOOL** bLoop, double fBlendDuration) **PURE**

Member Function Documentation

§ Play()

```
virtual HRESULT Play ( const GUID &  guidAnimationID,
                       BOOL          bLoop
                     )
```

private virtual

Called to invoke a specified animation.

§ TransitionAndPlay()

```
virtual HRESULT TransitionAndPlay ( const GUID &  guidCurrentAnimationID,
                                    const GUID &  guidNextAnimationID,
                                    BOOL          bLoop,
                                    double        fBlendDuration
                                  )
```

private virtual

Called to transition from one animation to another.

§ P3D::IAvatarAttachServiceV01

class P3D::IAvatarAttachServiceV01

This service allows configuring conditions or constraints in which the avatar can be attached and detached on this object. For example, implement this on an airplane to prevent detaching at high speeds.

Inherits IUnknown.

Private Member Functions

virtual **BOOL** **CanAvatarAttach** () const **PURE**

virtual **BOOL** **CanAvatarDetach** () const **PURE**

Member Function Documentation

§ CanAvatarAttach()

virtual **BOOL** CanAvatarAttach ( ) const private virtual

Return if conditions are appropriate for attaching.

§ CanAvatarDetach()

virtual **BOOL** CanAvatarDetach ( ) const private virtual

Return if conditions are appropriate for detaching.

§ P3D::IMarkerManagerV310

class P3D::IMarkerManagerV310

This service allows for placing a graphical orthogonal marker on a simobject at a specified offset. This is useful for visualizing physical offsets relative to the visual model. For example, wheel and engine positions. By default, the marker consists of red 30 meter orthogonal lines in both the positive and negative directions of the X,Y, and Z axis. This service is available through the PDK service provider.

Inherits IMarkerManagerV01.

Private Member Functions

virtual HRESULT	**CreateObjectMarker** (__in UINT idObject, __out HANDLE &hHandle) **PURE**
virtual HRESULT	**UpdateObjectMarkerOffset** (__in const HANDLE hHandle, __in const **FXYZ** &vOffset) **PURE**
virtual HRESULT	**UpdateObjectMarkerOrientation** (__in const HANDLE hHandle, __in const **FXYZ** &vOrientation) **PURE**
virtual HRESULT	**UpdateObjectMarkerOffsetAndOrientation** (__in const HANDLE hHandle, __in const **FXYZ** &vOffset, __in const **FXYZ** &vOrientation) **PURE**
virtual HRESULT	**RemoveMarker** (__inout HANDLE &hHandle) **PURE**

Member Function Documentation

§ CreateObjectMarker()

virtual HRESULT CreateObjectMarker (__in UINT idObject,
__out HANDLE & hHandle
) private virtual

Creates a new marker with the manager. It is advised to pass "0" for the handle, as that will verify with the manager that this is a new marker. A valid handle will be returned when successfully created. idObject is the Object ID in which to attach the marker.

§ RemoveMarker()

virtual HRESULT RemoveMarker (__inout HANDLE & hHandle) private virtual

Called to remove the marker. This will unregister the marker. The handle will be returned to a value of "0", and use of the original should be avoided.

§ UpdateObjectMarkerOffset()

```
virtual HRESULT UpdateObjectMarkerOffset ( __in const HANDLE  hHandle,
                                           __in const FXYZ &  vOffset
                                         )
```

private virtual

This is called to update the offset from the object's center in which to draw the marker.

§ UpdateObjectMarkerOffsetAndOrientation()

```
virtual HRESULT UpdateObjectMarkerOffsetAndOrientation ( __in const HANDLE  hHandle,
                                                         __in const FXYZ &  vOffset,
                                                         __in const FXYZ &  vOrientation
                                                       )
```

private virtual

This is called to update both the offset and orientation from the object's center and body axis in which to draw the marker.

§ UpdateObjectMarkerOrientation()

```
virtual HRESULT UpdateObjectMarkerOrientation ( __in const HANDLE  hHandle,
                                                __in const FXYZ &  vOrientation
                                              )
```

private virtual

This is called to update the orientation relative to the object's body axis in which to draw the marker.

§ P3D::IDesignatorServiceV340

class P3D::IDesignatorServiceV340

Interface implemented on a SimObject in order for core Prepar3D to interface with it for the purposes of broadcasting DIS related PDUs. This interface may also be queried by an ISimObject to gather designator related information.

Inherits IUnknown.

Private Member Functions

virtual UINT	**GetDesignatorCount** () const **PURE**
virtual **BOOL**	**IsActive** (__in UINT iDesignator) const **PURE**
virtual USHORT	**GetCodeName** (__in UINT iDesignator) const **PURE**
virtual UINT	**GetDesignatedObjectId** (__in UINT iDesignator) const **PURE**
virtual USHORT	**GetDesignatorCode** (__in UINT iDesignator) const **PURE**
virtual float	**GetDesignatorPower** (__in UINT iDesignator) const **PURE**
virtual float	**GetDesignatorWaveLength** (__in UINT iDesignator) const **PURE**
virtual HRESULT	**GetDesignatorSpotLocation** (__in UINT iDesignator, __out **DXYZ** &vWorldPosRadiansFeet) const **PURE**
virtual HRESULT	**GetDesignatorSpotAcceleration** (__in UINT iDesignator, __out **DXYZ** &vWorldAccelerationFpss) const **PURE**

Member Function Documentation

§ GetCodeName()

virtual USHORT GetCodeName (__in UINT iDesignator) const [private] [virtual]

The DIS code name

§ GetDesignatedObjectId()

virtual UINT GetDesignatedObjectId (__in UINT iDesignator) const [private] [virtual]

The object ID of the designated target

§ GetDesignatorCode()

virtual USHORT GetDesignatorCode (__in UINT iDesignator) const [private] [virtual]

The DIS designator code

§ GetDesignatorCount()

virtual UINT GetDesignatorCount ( ) const private virtual

The number of designators

§ GetDesignatorPower()

virtual float GetDesignatorPower (__in UINT iDesignator) const private virtual

The power of the designator in watts

§ GetDesignatorSpotAcceleration()

virtual HRESULT GetDesignatorSpotAcceleration (__in UINT iDesignator,
__out **DXYZ** & vWorldAccelerationFpss
) const private virtual

The world acceleration of the spot designation in feet/sec

§ GetDesignatorSpotLocation()

virtual HRESULT GetDesignatorSpotLocation (__in UINT iDesignator,
__out **DXYZ** & vWorldPosRadiansFeet
) const private virtual

The world location of the spot designation in radians/feet

§ GetDesignatorWaveLength()

virtual float GetDesignatorWaveLength (__in UINT iDesignator) const private virtual

The wavelength of the designator in microns

§ IsActive()

virtual **BOOL** IsActive (__in UINT iDesignator) const private virtual

The active state of the given designator

§ P3D::IRayTraceManagerV340

class P3D::IRayTraceManagerV340

This service allows the user to perform collision based ray tracing. Ray tracing can be performed on either objects, terrain, or both based on the given interrogation type. When casting from an object location, that object's object id should be in the ignore field to prevent it from casting on itself.

Object interrogation is typically more expensive than terrain interrogation. Ray length and granularity can be used to help control performance depending on the need of the ray trace. Ray trace calls should typically be done on both objects and terrain at a shorted ray length and a more precise granularity first. If no collision is detected, a higher ray length and less precise terrain based ray trace should be performed.

Note

Ray tracing is an expensive operation. Ray trace calls should be limited whenever possible.

Inherits IUnknown.

Private Member Functions

virtual HRESULT **InterrogateWorldRay** (__in DWORD dwInterrogationTypes, __in_opt UINT iIgnoreObjectId, __in const **DXYZ** &vWorldRadiansFeet, __in const **DXYZ** &xyzWorldUnitRayDir, __in float fRayLengthMax, __in float fGranularityMin, __out_opt UINT *pResultObjectId, __out_opt **DXYZ** *pResultWorldRadiansFeet, __inout DWORD &dwInterogationResults) const **PURE**

Member Function Documentation

§ InterrogateWorldRay()

```
virtual HRESULT InterrogateWorldRay ( __in DWORD          dwInterrogationTypes,
                                      __in_opt UINT       iIgnoreObjectId,
                                      __in const DXYZ &   vWorldRadiansFeet,
                                      __in const DXYZ &   xyzWorldUnitRayDir,
                                      __in float          fRayLengthMax,
                                      __in float          fGranularityMin,
                                      __out_opt UINT *    pResultObjectId,
                                      __out_opt DXYZ *    pResultWorldRadiansFeet,
                                      __inout DWORD &     dwInterogationResults
                                    )                     const
```

private virtual

Performs a world space based collision ray trace.

Parameters

dwInterrogationTypes	INTEROGATIONTYPE flags. See **Types.h**
iIgnoreObjectId	This object will be ignored when performing the ray trace, likely the casting object (Optional)
vWorldRadiansFeet	The initial LonAltLat of the ray trace in world radians/feet
xyzWorldUnitRayDir	A unit vector representing the orientation in world space
fRayLengthMax	The maximum length of the ray trace in meters
fGranularityMin	The minimum step distance of the ray trace in meters
pResultObjectId	The resulting object id of the collision (Optional)
pResultWorldRadiansFeet	The resulting LonAltLat of the collision in world radians/feet (Optional)
dwInterogationResults	The resulting INTEROGATIONTYPE flags. See **Types.h**

§ P3D::IEmissionsServiceV340

class P3D::IEmissionsServiceV340

Interface implemented on a SimObject in order for core Prepar3D to interface with it for the purposes of broadcasting DIS related PDUs. This interface may also be queried by an ISimObject to gather electromagnetic emission related information.

Inherits IUnknown.

Private Member Functions

virtual UINT	**GetEmissionSystemCount** () const **PURE**
virtual UINT	**GetEmitterBeamCount** (__in UINT iSystem) const **PURE**
virtual UINT	**GetEmissionSystemName** (__in UINT iSystem) const **PURE**
virtual UINT	**GetEmissionSystemFunction** (__in UINT iSystem) const **PURE**
virtual UINT	**GetEmitterNumber** (__in UINT iSystem) const **PURE**
virtual HRESULT	**GetEmissionSystemBodyOffset** (__in UINT iSystem, __out **P3D::P3DDXYZ** &vBodyOffsetFeet) const **PURE**
virtual UINT	**GetTrackJamCount** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual UINT	**GetEmitterBeamNumber** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual UINT	**GetEmitterBeamParameter** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual float	**GetEmitterBeamFrequency** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual float	**GetEmitterBeamFrequencyRange** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual float	**GetEmitterBeamEffectiveRadiatedPower** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual float	**GetEmitterBeamPulseRepetitionFrequency** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual float	**GetEmitterBeamPulseWidth** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual float	**GetEmitterBeamAzimuthCenter** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual float	**GetEmitterBeamAzimuthSweep** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual float	**GetEmitterBeamElevationCenter** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual float	**GetEmitterBeamElevationSweep** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual float	**GetEmitterBeamSweepSync** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual UINT	**GetEmitterBeamFunction** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual **BOOL**	**GetEmitterBeamIsHighDensityTrack** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual UINT	**GetEmitterBeamJammingMode** (__in UINT iSystem, __in UINT iBeam) const **PURE**
virtual UINT	**GetTrackJamObjectId** (__in UINT iSystem, __in UINT iBeam, __in UINT iTrackJam) const **PURE**
virtual UINT	**GetTrackJamEmitterNumber** (__in UINT iSystem, __in UINT iBeam, __in UINT iTrackJam) const **PURE**
virtual UINT	**GetTrackJamBeamNumber** (__in UINT iSystem, __in UINT iBeam, __in UINT iTrackJam) const **PURE**

Member Function Documentation

§ GetEmissionSystemBodyOffset()

virtual HRESULT GetEmissionSystemBodyOffset (__in UINT iSystem,
__out **P3D::P3DDXYZ** & vBodyOffsetFeet
) const private virtual

The offset of the beam source in body coordinates (feet)

§ GetEmissionSystemCount()

virtual UINT GetEmissionSystemCount ( ) const private virtual

The number of emission systems

§ GetEmissionSystemFunction()

virtual UINT GetEmissionSystemFunction (__in UINT iSystem) const private virtual

The DIS emission system function

§ GetEmissionSystemName()

virtual UINT GetEmissionSystemName (__in UINT iSystem) const private virtual

The DIS emission system name

§ GetEmitterBeamAzimuthCenter()

virtual float GetEmitterBeamAzimuthCenter (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The center azimuth of the beam in radians relative to the emitter

§ GetEmitterBeamAzimuthSweep()

virtual float GetEmitterBeamAzimuthSweep (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The half-angle sweep of the azimuth in radians relative to the center azimuth

§ GetEmitterBeamCount()

virtual UINT GetEmitterBeamCount (__in UINT iSystem) const private virtual

The number of emitter beam in the given emission system

§ GetEmitterBeamEffectiveRadiatedPower()

virtual float GetEmitterBeamEffectiveRadiatedPower (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The average effective radiated power of the beam in dBm

§ GetEmitterBeamElevationCenter()

virtual float GetEmitterBeamElevationCenter (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The center elevation of the beam in radians relative to the emitter

§ GetEmitterBeamElevationSweep()

virtual float GetEmitterBeamElevationSweep (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The half-angle sweep of the elevation in radians relative to the center elevation

§ GetEmitterBeamFrequency()

virtual float GetEmitterBeamFrequency (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The frequency of the beam in hertz

§ GetEmitterBeamFrequencyRange()

virtual float GetEmitterBeamFrequencyRange (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The frequency range of the beam in hertz

§ GetEmitterBeamFunction()

virtual UINT GetEmitterBeamFunction (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The DIS beam function

§ GetEmitterBeamIsHighDensityTrack()

virtual **BOOL** GetEmitterBeamIsHighDensityTrack (__in UINT iSystem,
__in UINT iBeam
) const private virtual

True if all targets in the scan pattern are to be considered tracked or jammed

§ GetEmitterBeamJammingMode()

virtual UINT GetEmitterBeamJammingMode (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The DIS jamming mode sequence for the given emitter in string representation

§ GetEmitterBeamNumber()

virtual UINT GetEmitterBeamNumber (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The beam identification number

§ GetEmitterBeamParameter()

virtual UINT GetEmitterBeamParameter (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The DIS beam parameter

§ GetEmitterBeamPulseRepetitionFrequency()

virtual float GetEmitterBeamPulseRepetitionFrequency (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The average pulse repetition frequency of the beam in hertz

§ GetEmitterBeamPulseWidth()

virtual float GetEmitterBeamPulseWidth (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The average pulse width of the beam in microseconds

§ GetEmitterBeamSweepSync()

virtual float GetEmitterBeamSweepSync (__in UINT iSystem,
__in UINT iBeam
) const private virtual

The range from 0.0 to 100.0 representing the percentage of the pattern scanned

§ GetEmitterNumber()

virtual UINT GetEmitterNumber (__in UINT iSystem) const private virtual

The emitter identification number

§ GetTrackJamBeamNumber()

virtual UINT GetTrackJamBeamNumber (__in UINT iSystem,
__in UINT iBeam,
__in UINT iTrackJam
) const

private virtual

The beam identification number associated with the target

§ GetTrackJamCount()

virtual UINT GetTrackJamCount (__in UINT iSystem,
__in UINT iBeam
) const

private virtual

The number targets being tracked/jammed

§ GetTrackJamEmitterNumber()

virtual UINT GetTrackJamEmitterNumber (__in UINT iSystem,
__in UINT iBeam,
__in UINT iTrackJam
) const

private virtual

The emitter identification number associated with the target

§ GetTrackJamObjectId()

virtual UINT GetTrackJamObjectId (__in UINT iSystem,
__in UINT iBeam,
__in UINT iTrackJam
) const

private virtual

The object id of the target being tracked/jammed

§ P3D::IRadioSystemV400

class P3D::IRadioSystemV400

IRadioSystemV400 This interface can be queried for on any object to determine if a radio system is available.

Inherits IUnknown.

Private Member Functions

virtual **BOOL** **AreRadiosActive** () const **PURE**

Member Function Documentation

§ AreRadiosActive()

virtual **BOOL** AreRadiosActive ( ) const private virtual

Queries if any radios are currently actove

Returns

TRUE if any radios active

§ P3D::IAttachmentServiceV430

class P3D::IAttachmentServiceV430

IAttachmentServiceV430 This interface can be queried for on an object to obtain attachment data.

Inherits IAttachmentServiceV420.

Private Member Functions

virtual HRESULT	**GetAttachPointIndex** (__in __notnull LPCSTR pszAttachPointName, __out UINT &uIndex) const **PURE**
virtual HRESULT	**GetAttachPointCount** (__out UINT &uCount) const **PURE**
virtual HRESULT	**GetAttachPointOffset** (__in UINT uIndex, __out **DXYZ** &vOffsetMeters) const **PURE**
virtual HRESULT	**GetAttachPointOrientation** (__in UINT uIndex, __out **DXYZ** &vOrientationRadians) const **PURE**
virtual HRESULT	**GetAttachedObjectCount** (__out UINT &nObjects) const **PURE**
virtual HRESULT	**GetAttachedObjects** (__inout UINT &nObjects, __out UINT *rgObjectIDs) const **PURE**
virtual HRESULT	**GetAttachedObjectIndex** (__in UINT uObjectId, __out UINT &uIndex) const **PURE**
virtual HRESULT	**GetAttachedObjectId** (__in UINT uIndex, __out UINT &uObjectId) const **PURE**
virtual HRESULT	**UpdateAttachments** () **PURE**
virtual UINT	**GetParentId** () **PURE**
virtual HRESULT	**SetOffsetFeet** (__in const **DXYZ** &vOffsetFeet) **PURE**
virtual HRESULT	**GetOffsetFeet** (__out **DXYZ** &vOffsetFeet) **PURE**
virtual HRESULT	**SetOffsetRadians** (__in const **DXYZ** &vOffsetRadians) **PURE**
virtual HRESULT	**GetOffsetRadians** (__out **DXYZ** &vOffsetRadians) **PURE**

Member Function Documentation

§ GetAttachedObjectCount()

virtual HRESULT GetAttachedObjectCount (__out UINT & nObjects) const [private] [virtual]

Gets the number of attached objects.

Parameters

nObjects The number of attached objects.

Returns

S_OK if the attached objects were successfully found, E_FAIL otherwise.

§ GetAttachedObjectId()

```
virtual HRESULT GetAttachedObjectId ( __in UINT      uIndex,
                                      __out UINT &   uObjectId
                                    )                const
```
private virtual

Gets the attached object id for the given attach point index.

Parameters

uIndex The attach point index.

uObjectId The attached object's id.

Returns

S_OK if the attach point index's attached object id is successfully found, E_FAIL otherwise.

§ GetAttachedObjectIndex()

```
virtual HRESULT GetAttachedObjectIndex ( __in UINT      uObjectId,
                                         __out UINT &   uIndex
                                       )                const
```
private virtual

Gets the attach point index for the given attached object id.

Parameters

uObjectId The attached object's id.

uIndex The attached object's attach point index.

Returns

S_OK if the attached object's attach point index is successfully found, E_FAIL otherwise.

§ GetAttachedObjects()

```
virtual HRESULT GetAttachedObjects ( __inout UINT &   nObjects,
                                     __out UINT *     rgObjectIDs
                                   )                  const
```
private virtual

Gets a list of all attached objects.

Parameters

nObjects IN: The max number of objects requested. This must be no smaller than the size of the array pointed to by rgObjectIDs.

nObjects OUT: The actual number of objects found.

rgObjectIDs Address of array in which object IDs are returned.

Returns

S_OK if the attached objects were successfully found, E_FAIL otherwise.

Note

It is the callers responsibility to allocate the array's required memory.

§ GetAttachPointCount()

virtual HRESULT GetAttachPointCount (__out UINT & uCount) const [private] [virtual]

Gets the attach point count.

Parameters

uCount The number of attach points.

Returns

S_OK if the attach point count was successfully found, E_FAIL otherwise.

§ GetAttachPointIndex()

virtual HRESULT GetAttachPointIndex (__in __notnull LPCSTR pszAttachPointName,
__out UINT & uIndex
) const [private] [virtual]

Gets the attach point index from the given name.

Parameters

pszAttachPointName The name of the attach point.

uIndex The index of the given attach point.

Returns

S_OK if the attach point index was successfully found, E_FAIL otherwise.

§ GetAttachPointOffset()

virtual HRESULT GetAttachPointOffset (__in UINT uIndex,
__out **DXYZ** & vOffsetMeters
) const [private] [virtual]

Gets the offset of the attach point with the given index.

Parameters

uIndex The attach point index.

vOffsetMeters The body offset of the attach point in meters.

Returns

S_OK if the attach point offset was successfully found, E_FAIL otherwise.

Note

Units for this function are in meters, not feet.

§ GetAttachPointOrientation()

virtual HRESULT GetAttachPointOrientation (__in UINT uIndex,
__out **DXYZ** & vOrientationRadians
) const private virtual

Gets the orientation of the attach point with the given index.

Parameters

uIndex The attach point index.

vOrientationRadians The body orientation offset of the attach point in radians.

Returns

S_OK if the attach point orientation was successfully found, E_FAIL otherwise.

§ GetOffsetFeet()

virtual HRESULT GetOffsetFeet (__out **DXYZ** & vOffsetFeet) private virtual

If this object is attached to another object, this function will provide the offset in feet relative to the parent object and return S_OK. If this object is not attached to another object, this function will return E_FAIL.

Parameters

vOffsetFeet The offset relative to the parent object in feet.

Returns

S_OK if attached and the offset is valid, E_FAIL otherwise.

§ GetOffsetRadians()

virtual HRESULT GetOffsetRadians (__out **DXYZ** & vOffsetRadians) private virtual

If this object is attached to another object, this function will provide the orientation offset in radians relative to the parent object and return S_OK. If this object is not attached to another object, this function will return E_FAIL.

Parameters

vOffsetRadians The orientation offset relative to the parent object in radians.

Returns

S_OK if attached and the offset is valid, E_FAIL otherwise.

§ GetParentId()

virtual UINT GetParentId ( ) private virtual

If this object is attached to another object, this function will return the parent's object ID. If this object is not attached to another object, this function will return zero.

Returns

The parent's object id if attached, zero otherwise.

§ SetOffsetFeet()

virtual HRESULT SetOffsetFeet (__in const **DXYZ** & vOffsetFeet) private virtual

If this object is attached to another object, this function will set the offset in feet relative to the parent object and return S_OK. If this object is not attached to another object (or this object is attached via an attachpoint), this function will return E_FAIL.

Parameters

vOffsetFeet The offset relative to the parent object in feet.

Returns

S_OK if attached and the offset was correctly set, E_FAIL otherwise.

§ SetOffsetRadians()

virtual HRESULT SetOffsetRadians (__in const **DXYZ** & vOffsetRadians) private virtual

If this object is attached to another object, this function will set the orientation offset in radians relative to the parent object and return S_OK. If this object is not attached to another object (or this object is attached via an attachpoint), this function will return E_FAIL.

Parameters

vOffsetRadians The orientation offset relative to the parent object in radians.

Returns

S_OK if attached and the offset was correctly set, E_FAIL otherwise.

§ UpdateAttachments()

virtual HRESULT UpdateAttachments ( ) private virtual

Updates the location and orientation of all attach points. Should be called after an object has simulated to ensure attached objects are in the correct location.

Returns

S_OK if at least on attach point was updated, S_FALSE otherwise.

§ P3D::IAIBehaviorManagerV01

class P3D::IAIBehaviorManagerV01

Professional Plus Only

Interface for this object's AIBehaviorManager

Inherits IAIBehaviorManager.

Private Member Functions

virtual HRESULT	**ActivateBehavior** (__in const GUID &BehaviorGuid, **BOOL** bActivate) **PURE**
virtual **BOOL**	**IsBehaviorActive** (__in const GUID &BehaviorGuid) const **PURE**
virtual UINT	**GetNumberOfBehaviors** () const **PURE**
virtual HRESULT	**GetBehavior** (__in UINT uIndex, __out IAIBehavior **ppBehavior) const **PURE**

Member Function Documentation

§ ActivateBehavior()

virtual HRESULT ActivateBehavior (__in const GUID & BehaviorGuid,
BOOL bActivate
) private virtual

Activates or deactivates a behavior

§ GetBehavior()

virtual HRESULT GetBehavior (__in UINT uIndex,
__out IAIBehavior ** ppBehavior
) const private virtual

Gets a behavior from an index

§ GetNumberOfBehaviors()

virtual UINT GetNumberOfBehaviors ( ) const private virtual

Returns the total number of behaviors

§ IsBehaviorActive()

virtual **BOOL** IsBehaviorActive (__in const GUID & BehaviorGuid) const private virtual

Returns whether a behavior is active (true) or inactive (false)

§ P3D::IAIBehaviorWingmanFormationV01

class P3D::IAIBehaviorWingmanFormationV01

Professional Plus Only

Interface for the Wingman Formation AI Behavior

Inherits IAIBehaviorWingmanFormation.

Private Member Functions

virtual GUID	**GetBehaviorGuid** () const **PURE**
virtual void	**SetLeaderObjectID** (int objectID) **PURE**
virtual int	**GetLeaderObjectID** () const **PURE**
virtual void	**SetOffsetPosition** (const **P3D::DXYZ** &offsetFeet) **PURE**
virtual void	**GetOffsetPosition** (**P3D::DXYZ** &offsetFeet) const **PURE**
virtual void	**WarpToTarget** () **PURE**

Member Function Documentation

§ GetBehaviorGuid()

virtual GUID GetBehaviorGuid ( ) const [private] [virtual]

Gets the Guid ID of a behavior

§ GetLeaderObjectID()

virtual int GetLeaderObjectID ( ) const [private] [virtual]

Gets the ID of the object's leader object

§ GetOffsetPosition()

virtual void GetOffsetPosition (**P3D::DXYZ** & offsetFeet) const [private] [virtual]

Gets the offset position of a behavior (in feet)

§ SetLeaderObjectID()

virtual void SetLeaderObjectID (int objectID) [private] [virtual]

Sets the ID of the object's leader object

§ SetOffsetPosition()

virtual void SetOffsetPosition (const **P3D::DXYZ** & offsetFeet) private virtual

Sets the offset position of a behavior (in feet)

§ WarpToTarget()

virtual void WarpToTarget ( ) private virtual

Moves the object to its target's location

§ P3D::IAIBehaviorAttackerV400

class P3D::IAIBehaviorAttackerV400

Professional Plus Only

Interface for the Attacker AI Behavior

Inherits IAIBehavior.

Private Member Functions

virtual GUID	**GetBehaviorGuid** () const **PURE**
virtual void	**SetFireRadiusMin** (float radiusFeet) **PURE**
virtual float	**GetFireRadiusMin** () const **PURE**
virtual void	**SetFireRadiusMax** (float radiusFeet) **PURE**
virtual float	**GetFireRadiusMax** () const **PURE**
virtual void	**SetFireFOVDegrees** (float degrees) **PURE**
virtual float	**GetFireFOVDegrees** () const **PURE**
virtual void	**SetWeaponTitle** (WCHAR *weaponTitle) **PURE**
virtual const WCHAR *	**GetWeaponTitle** () const **PURE**
virtual void	**SetWeaponType** (WCHAR *weaponType) **PURE**
virtual const WCHAR *	**GetWeaponType** () const **PURE**
virtual void	**SetGunTitle** (WCHAR *gunTitle) **PURE**
virtual const WCHAR *	**GetGunTitle** () const **PURE**
virtual void	**SetGunType** (WCHAR *gunType) **PURE**
virtual const WCHAR *	**GetGunType** () const **PURE**
virtual void	**SetAttackDelay** (float delaySeconds) **PURE**
virtual float	**GetAttackDelay** () const **PURE**
virtual void	**SetGunBurstDuration** (float durationSeconds) **PURE**
virtual float	**GetGunBurstDuration** () const **PURE**
virtual **BOOL**	**IsWithinFireZone** () const **PURE**

Member Function Documentation

§ GetAttackDelay()

virtual float GetAttackDelay ( ) const [private] [virtual]

Gets the delay between an attacker's attacks (in seconds)

§ GetBehaviorGuid()

virtual GUID GetBehaviorGuid ( ) const [private] [virtual]

Gets the Guid ID of a behavior

§ GetFireFOVDegrees()

virtual float GetFireFOVDegrees ( ) const private virtual

Gets the fire FOV of an attacker (in degrees)

§ GetFireRadiusMax()

virtual float GetFireRadiusMax ( ) const private virtual

Gets the maximum fire radius of an attacker (in feet)

§ GetFireRadiusMin()

virtual float GetFireRadiusMin ( ) const private virtual

Gets the minimum fire radius of an attacker (in feet)

§ GetGunBurstDuration()

virtual float GetGunBurstDuration ( ) const private virtual

Gets the duration of an attacker's gun burst (in seconds)

§ GetGunTitle()

virtual const WCHAR* GetGunTitle ( ) const private virtual

Gets the title of an attacker's gun

§ GetGunType()

virtual const WCHAR* GetGunType ( ) const private virtual

Gets the type of an attacker's gun

§ GetWeaponTitle()

virtual const WCHAR* GetWeaponTitle ( ) const private virtual

Gets the title of an attacker's weapon

§ GetWeaponType()

virtual const WCHAR* GetWeaponType ( ) const private virtual

Gets the type of an attacker's weapon

§ IsWithinFireZone()

virtual **BOOL** IsWithinFireZone ( ) const private virtual

Returns true if attacker is within fire zone, false otherwise

§ SetAttackDelay()

virtual void SetAttackDelay (float delaySeconds) private virtual

Sets the delay between an attacker's attacks (in seconds)

§ SetFireFOVDegrees()

virtual void SetFireFOVDegrees (float degrees) private virtual

Sets the fire FOV of an attacker (in degrees)

§ SetFireRadiusMax()

virtual void SetFireRadiusMax (float radiusFeet) private virtual

Sets the maximum fire radius of an attacker (in feet)

§ SetFireRadiusMin()

virtual void SetFireRadiusMin (float radiusFeet) private virtual

Sets the minimum fire radius of an attacker (in feet)

§ SetGunBurstDuration()

virtual void SetGunBurstDuration (float durationSeconds) private virtual

Sets the duration of an attacker's gun burst (in seconds)

§ SetGunTitle()

virtual void SetGunTitle (WCHAR * gunTitle) private virtual

Sets the title of an attacker's gun

§ SetGunType()

virtual void SetGunType (WCHAR * gunType) private virtual

Sets the type of an attacker's gun

§ SetWeaponTitle()

virtual void SetWeaponTitle (WCHAR * weaponTitle) private virtual

Sets the title of an attacker's weapon

§ SetWeaponType()

virtual void SetWeaponType (WCHAR * weaponType) private virtual

Sets the type of an attacker's weapon

§ P3D::IAIBehaviorPursueV01

class P3D::IAIBehaviorPursueV01

Professional Plus Only

Interface for the Pursue AI Behavior

Inherits IAIBehaviorPursue.

Private Member Functions

virtual GUID	**GetBehaviorGuid** () const **PURE**
virtual void	**SetInvestigateRadius** (float radiusFeet) **PURE**
virtual float	**GetInvestigateRadius** () const **PURE**
virtual void	**SetPursueObjectID** (int objectID) **PURE**
virtual int	**GetPursueObjectID** () const **PURE**
virtual void	**SetMaxPursuitDurationSeconds** (float maxSeconds) **PURE**
virtual float	**GetSetMaxPursuitDurationSeconds** () const **PURE**
virtual void	**SetInterceptGroundObjects** (**BOOL** intercept) **PURE**
virtual **BOOL**	**GetInterceptGroundObjects** () const **PURE**

Member Function Documentation

§ GetBehaviorGuid()

virtual GUID GetBehaviorGuid ( ) const [private] [virtual]

Gets the Guid ID of a pursue behavior

§ GetInterceptGroundObjects()

virtual **BOOL** GetInterceptGroundObjects ( ) const [private] [virtual]

Gets if object can intercept ground objects

§ GetInvestigateRadius()

virtual float GetInvestigateRadius ( ) const [private] [virtual]

Gets the how far an object will investigate from its current location (in feet)

§ GetPursueObjectID()

virtual int GetPursueObjectID ( ) const private virtual

Gets the ID of the object being pursued

§ GetSetMaxPursuitDurationSeconds()

virtual float GetSetMaxPursuitDurationSeconds ( ) const private virtual

Gets the maximum time an object will pursue for (in seconds)

§ SetInterceptGroundObjects()

virtual void SetInterceptGroundObjects (**BOOL** intercept) private virtual

Sets if object can intercept ground objects

§ SetInvestigateRadius()

virtual void SetInvestigateRadius (float radiusFeet) private virtual

Sets the how far an object will investigate from its current location (in feet)

§ SetMaxPursuitDurationSeconds()

virtual void SetMaxPursuitDurationSeconds (float maxSeconds) private virtual

Sets the maximum time an object will pursue for (in seconds)

§ SetPursueObjectID()

virtual void SetPursueObjectID (int objectID) private virtual

Sets the ID of the object being pursued

§ P3D::IAIBehaviorCombatAirPatrolV01

class P3D::IAIBehaviorCombatAirPatrolV01

Professional Plus Only

Interface for the Combat-Air-Patrol AI Behavior

Inherits IAIBehaviorCombatAirPatrol.

Private Member Functions

virtual GUID	**GetBehaviorGuid** () const **PURE**
virtual void	**SetPatrolObjectID** (int objectID) **PURE**
virtual int	**GetPatrolObjectID** () const **PURE**
virtual void	**SetPatrolOrigin** (const **P3D::DXYZ** &lonAltLat) **PURE**
virtual void	**GetPatrolOrigin** (**P3D::DXYZ** &lonAltLat) const **PURE**
virtual void	**SetPatrolRadius** (float radiusFeet) **PURE**
virtual float	**GetPatrolRadius** () const **PURE**

Member Function Documentation

§ GetBehaviorGuid()

virtual GUID GetBehaviorGuid ( ) const [private] [virtual]

Gets the Guid ID of a combat-air-patrol behavior

§ GetPatrolObjectID()

virtual int GetPatrolObjectID ( ) const [private] [virtual]

Gets the ID of the patrol object

§ GetPatrolOrigin()

virtual void GetPatrolOrigin (**P3D::DXYZ** & lonAltLat) const [private] [virtual]

Gets the origin point for the patrol (in radians and feet)

§ GetPatrolRadius()

virtual float GetPatrolRadius ( ) const [private] [virtual]

Gets the radius for the patrol (in feet)

§ SetPatrolObjectID()

virtual void SetPatrolObjectID (int objectID) private virtual

Sets the ID of the patrol object

§ SetPatrolOrigin()

virtual void SetPatrolOrigin (const **P3D::DXYZ** & lonAltLat) private virtual

Sets the origin point for the patrol (in radians and feet)

§ SetPatrolRadius()

virtual void SetPatrolRadius (float radiusFeet) private virtual

Sets the radius for the patrol (in feet)

§ P3D::IAIBehaviorCloseAirSupportV01

class P3D::IAIBehaviorCloseAirSupportV01

Professional Plus Only

Interface for the Close-Air-Support AI Behavior

Inherits IAIBehaviorCloseAirSupport.

Private Member Functions

virtual GUID	**GetBehaviorGuid** () const **PURE**
virtual void	**SetSupportObjectID** (int objectID) **PURE**
virtual int	**GetSupportObjectID** () const **PURE**
virtual void	**SetSupportPosition** (const **P3D::DXYZ** &lonAltLat) **PURE**
virtual void	**GetSupportPosition** (**P3D::DXYZ** &lonAltLat) const **PURE**

Member Function Documentation

§ GetBehaviorGuid()

virtual GUID GetBehaviorGuid ( ) const [private] [virtual]

Gets the Guid ID of a close-air-support behavior

§ GetSupportObjectID()

virtual int GetSupportObjectID ( ) const [private] [virtual]

Gets the ID of the support object

§ GetSupportPosition()

virtual void GetSupportPosition (**P3D::DXYZ** & lonAltLat) const [private] [virtual]

Gets the position for the support (in radians and feet)

§ SetSupportObjectID()

virtual void SetSupportObjectID (int objectID) [private] [virtual]

Sets the ID of the support object

§ SetSupportPosition()

virtual void SetSupportPosition (const **P3D::DXYZ** & lonAltLat) private virtual

Sets the position for the support (in radians and feet)

§ P3D::IAIBehaviorSearchTrackV01

class P3D::IAIBehaviorSearchTrackV01

Professional Plus Only

Interface for the Search and Track AI Behavior

Inherits IAIBehaviorSearchTrack.

Private Member Functions

virtual GUID	**GetBehaviorGuid** () const **PURE**
virtual void	**SetSearchRadius** (float radiusFeet) **PURE**
virtual float	**GetSearchRadius** () const **PURE**
virtual void	**SetSearchFOVDegrees** (float degrees) **PURE**
virtual float	**GetSearchFOVDegrees** () const **PURE**
virtual void	**SetTrackRadius** (float radiusFeet) **PURE**
virtual float	**GetTrackRadius** () const **PURE**
virtual void	**SetTrackFOVDegrees** (float degrees) **PURE**
virtual float	**GetTrackFOVDegrees** () const **PURE**
virtual **BOOL**	**IsWithinSearchZone** () const **PURE**
virtual **BOOL**	**IsWithinTrackZone** () const **PURE**

Member Function Documentation

§ GetBehaviorGuid()

virtual GUID GetBehaviorGuid ( ) const [private] [virtual]

Gets the Guid ID of a search and track behavior

§ GetSearchFOVDegrees()

virtual float GetSearchFOVDegrees ( ) const [private] [virtual]

Gets the FOV of a search (in degrees)

§ GetSearchRadius()

virtual float GetSearchRadius ( ) const [private] [virtual]

Gets the radius to search (in feet)

§ GetTrackFOVDegrees()

virtual float GetTrackFOVDegrees ( ) const private virtual

Gets the FOV of a track (in degrees)

§ GetTrackRadius()

virtual float GetTrackRadius ( ) const private virtual

Gets the radius to track (in feet)

§ IsWithinSearchZone()

virtual **BOOL** IsWithinSearchZone ( ) const private virtual

Returns true if behavior is within search zone, false otherwise

§ IsWithinTrackZone()

virtual **BOOL** IsWithinTrackZone ( ) const private virtual

Returns true if behavior is within track zone, false otherwise

§ SetSearchFOVDegrees()

virtual void SetSearchFOVDegrees (float degrees) private virtual

Sets the FOV of a search (in degrees)

§ SetSearchRadius()

virtual void SetSearchRadius (float radiusFeet) private virtual

Sets the radius to search (in feet)

§ SetTrackFOVDegrees()

virtual void SetTrackFOVDegrees (float degrees) private virtual

Sets the FOV of a track (in degrees)

§ SetTrackRadius()

virtual void SetTrackRadius (float radiusFeet) private virtual

Sets the radius to track (in feet)

§ P3D::ISimObjectAIV02

class P3D::ISimObjectAIV02

The ISimObjectAI interface is an interface on the AI "pilot" implementation for a simobject. A custom AI can be implemented on simobjects created using the ISimObject SDK. The interface may be accessed by systems such as the Traffic Manager or ATC.

Inherits ISimObjectAIV01.

Private Member Functions

virtual HRESULT	**UpdateSimulationFrame** (__in double dDeltaT) **PURE**
virtual **UNITMODE**	**GetPilotMode** () const **PURE**
virtual void	**SetPilotMode** (**UNITMODE** eMode, **BOOL** bOn=**TRUE**) **PURE**
virtual void	**Deactivate** () **PURE**
virtual void	**Activate** () **PURE**
virtual HRESULT	**SetWaypoint** (__in **BasicWaypoint**) **PURE**
virtual HRESULT	**SetDesiredHeading** (double dTrueHeading) **PURE**
virtual HRESULT	**SetDesiredPitch** (double dPitch) **PURE**
virtual HRESULT	**SetDesiredSpeed** (double dSpeed) **PURE**
virtual HRESULT	**SetDesiredAltitude** (double dAltitudeMSL) **PURE**
virtual double	**GetDesiredHeading** () const **PURE**
virtual double	**GetDesiredPitch** () const **PURE**
virtual double	**GetDesiredSpeed** () const **PURE**
virtual double	**GetDesiredAltitude** () const **PURE**

Member Function Documentation

§ Activate()

virtual void Activate ( ) private virtual

Enables the AI control

§ Deactivate()

virtual void Deactivate ( ) private virtual

Disables the AI control

§ GetDesiredAltitude()

virtual double GetDesiredAltitude ( ) const private virtual

Gets the current altitude to be maintained (in feet)

§ GetDesiredHeading()

virtual double GetDesiredHeading ( ) const private virtual

Gets the current heading to be maintained (in radians)

§ GetDesiredPitch()

virtual double GetDesiredPitch ( ) const private virtual

Gets the current pitch to be maintained (in radians)

§ GetDesiredSpeed()

virtual double GetDesiredSpeed ( ) const private virtual

Gets the current speed to be maintained (in feet per second)

§ GetPilotMode()

virtual **UNITMODE** GetPilotMode ( ) const private virtual

Returns the current mode of the AI

§ SetDesiredAltitude()

virtual HRESULT SetDesiredAltitude (double dAltitudeMSL) private virtual

Sets the altitude to be maintained (in feet)

§ SetDesiredHeading()

virtual HRESULT SetDesiredHeading (double dTrueHeading) private virtual

Sets the heading to be maintained (in radians)

§ SetDesiredPitch()

virtual HRESULT SetDesiredPitch (double dPitch) private virtual

Sets the pitch angle to be maintained (in radians)

§ SetDesiredSpeed()

virtual HRESULT SetDesiredSpeed (double dSpeed) private virtual

Sets the desired speed to be maintained (Indicated airspeed for aircraft) (in feet per second)

§ SetPilotMode()

```
virtual void SetPilotMode ( UNITMODE eMode,
                            BOOL     bOn = TRUE
                          )
```
private virtual

Sets the current mode of the AI

§ SetWaypoint()

virtual HRESULT SetWaypoint (__in BasicWaypoint) private virtual

Sets the next waypoint for the AI to track to

§ UpdateSimulationFrame()

virtual HRESULT UpdateSimulationFrame (__in double dDeltaT) private virtual

To be called from the simulation loop to keep the low level AI controllers in sync with the simulation

§ P3D::IAirplaneAIServiceV02

class P3D::IAirplaneAIServiceV02

The IAirplaneAIService should be implemented on any airplane intended to be controlled by Prepar3D's internal AI Pilot.

Inherits IAirplaneAIServiceV01.

Private Member Functions

virtual float	**GetStallSpeedDirty** () const **PURE**
virtual float	**GetStallSpeedClean** () const **PURE**
virtual float	**GetMinDragSpeed** () const **PURE**
virtual float	**GetZeroLiftAngleOfAttack** () const **PURE**
virtual float	**GetCriticalAngleOfAttack** () const **PURE**
virtual float	**GetLinearCLAlpha** () const **PURE**
virtual float	**GetWingArea** () const **PURE**
virtual float	**GetWingSpan** () const **PURE**
virtual float	**GetTotalLongitudinalThrust** () const **PURE**
virtual float	**GetLiftForce** () const **PURE**
virtual double	**GetThrottlePercent** () const **PURE**
virtual double	**GetElevatorPercent** () const **PURE**
virtual double	**GetAileronPercent** () const **PURE**
virtual double	**GetRudderPercent** () const **PURE**
virtual double	**GetSpoilersPercent** () const **PURE**
virtual double	**GetFlapsPercent** () const **PURE**
virtual void	**SetThrottlePercent** (double dPct) **PURE**
virtual void	**SetElevatorPercent** (double dPct) **PURE**
virtual void	**SetAileronPercent** (double dPct) **PURE**
virtual void	**SetRudderPercent** (double dPct) **PURE**
virtual void	**SetFlapsPercent** (double dPct) **PURE**
virtual void	**SetSpoilersPercent** (double dPct) **PURE**
virtual double	**CalculateDesiredBank** (double dHeadingError, double dDeltaT) **PURE**
virtual double	**CalculateDeltaThrottle** (double dSpeedError, double dDeltaT) **PURE**
virtual void	**SetTaxiHeading** (float fHeading) **PURE**
virtual void	**SetTaxiSpeed** (float fSpeed) **PURE**
virtual void	**StopTaxi** () **PURE**
virtual void	**SetPushBack** (**BOOL** bOn) **PURE**
virtual void	**ExtendTailhook** () **PURE**
virtual void	**RetractTailhook** () **PURE**
virtual float	**GetTailhookPosition** () const **PURE**
virtual **BOOL**	**HasTailhook** () const **PURE**
virtual void	**ExtendLaunchBar** () **PURE**
virtual void	**RetractLaunchBar** () **PURE**
virtual float	**GetLaunchBarPosition** () const **PURE**
virtual **BOOL**	**HasLaunchBar** () const **PURE**
virtual void	**FoldWings** () **PURE**

virtual void	**UnfoldWings** () **PURE**
virtual float	**GetLeftWingPosition** () const **PURE**
virtual float	**GetRightWingPosition** () const **PURE**
virtual float	**GetFullMilitaryThrottlePosition** () const **PURE**
virtual void	**ArmNearestCatapult** (**BOOL** bArm) **PURE**
virtual void	**FireArmedCatapult** () **PURE**

Member Function Documentation

§ ArmNearestCatapult()

virtual void ArmNearestCatapult (**BOOL** bArm) private virtual

Attempts to attach and arm the nearest catapult.

§ CalculateDeltaThrottle()

virtual double CalculateDeltaThrottle (double dSpeedError,
double dDeltaT
) private virtual

Returns the amount the throttle should be moved this frame based on the current speed error This is typically calculated with a PID controller based on the airplane and engine dynamics. (-1 - +1)

§ CalculateDesiredBank()

virtual double CalculateDesiredBank (double dHeadingError,
double dDeltaT
) private virtual

Returns a desired bank for the AI pilot based on a heading error. This is typically calculated with a PID controller based on the airplane dynamics (radians)

§ ExtendLaunchBar()

virtual void ExtendLaunchBar ( ) private virtual

Extends the aircraft's launch bar inorder to attach to catapults.

§ ExtendTailhook()

virtual void ExtendTailhook ( ) private virtual

Extends the aircraft's tailhook inorder to catch arrestor cables.

§ FireArmedCatapult()

virtual void FireArmedCatapult ( ) private virtual

Fires the currently armed catapult.

§ FoldWings()

virtual void FoldWings ( ) private virtual

Folds the aircraft's wings, if available.

§ GetAileronPercent()

virtual double GetAileronPercent ( ) const private virtual

Returns the aileron position (-1 left - +1 right)

§ GetCriticalAngleOfAttack()

virtual float GetCriticalAngleOfAttack ( ) const private virtual

Returns angle-of-attack at which the aircraft will stall (radians)

§ GetElevatorPercent()

virtual double GetElevatorPercent ( ) const private virtual

Returns the elevator position (-1 down - +1 up)

§ GetFlapsPercent()

virtual double GetFlapsPercent ( ) const private virtual

Returns the flap percent deflection (0 - 1)

§ GetFullMilitaryThrottlePosition()

virtual float GetFullMilitaryThrottlePosition ( ) const

private virtual

Returns the throttle position that is considered to be full military power (no afterburner). (position from 0.0 to 1.0)

§ GetLaunchBarPosition()

virtual float GetLaunchBarPosition ( ) const

private virtual

Returns the position of the aircraft's launch bar. (retracted=0.0; extended=1.0)

§ GetLeftWingPosition()

virtual float GetLeftWingPosition ( ) const

private virtual

Returns the current folded position of the aircraft's left wing. (unfolded=0.0; folded=1.0)

§ GetLiftForce()

virtual float GetLiftForce ( ) const

private virtual

Returns the lift force generated by the airplane, including wing, tail, and fuselage) (pounds)

§ GetLinearCLAlpha()

virtual float GetLinearCLAlpha ( ) const

private virtual

Returns the slope of the CL vs. angle-of-attack curve in the linear region, typically between zero lift and the critical angle-of-attack (radians)

§ GetMinDragSpeed()

virtual float GetMinDragSpeed ( ) const

private virtual

Returns speed at which total drag is at its minimum (feet per second)

§ GetRightWingPosition()

virtual float GetRightWingPosition ( ) const

private virtual

Returns the current folded position of the aircraft's right wing. (unfolded=0.0; folded=1.0)

§ GetRudderPercent()

virtual double GetRudderPercent ( ) const

private virtual

Returns rudder percent (-1 left - +1 right)

§ GetSpoilersPercent()

virtual double GetSpoilersPercent ( ) const

private virtual

Return the spoiler position (0 - 1)

§ GetStallSpeedClean()

virtual float GetStallSpeedClean ( ) const

private virtual

Returns stall speed with gear and flaps retracted (feet per second)

§ GetStallSpeedDirty()

virtual float GetStallSpeedDirty ( ) const

private virtual

Returns stall speed with gear and flaps extended (feet per second)

§ GetTailhookPosition()

virtual float GetTailhookPosition ( ) const

private virtual

Returns the position of the aircraft's tailhook. (retracted=0.0; extended=1.0)

§ GetThrottlePercent()

virtual double GetThrottlePercent ( ) const

private virtual

Returns the throttle position (0 - 1)

§ GetTotalLongitudinalThrust()

virtual float GetTotalLongitudinalThrust ( ) const private virtual

Returns the thrust force in the longitudinal axis (pounds)

§ GetWingArea()

virtual float GetWingArea ( ) const private virtual

Returns the total area of the main wing (feet squared)

§ GetWingSpan()

virtual float GetWingSpan ( ) const private virtual

Return the wingspan of the main wing (feet)

§ GetZeroLiftAngleOfAttack()

virtual float GetZeroLiftAngleOfAttack ( ) const private virtual

Return angle-of-attack at which zero lift is generated (radians)

§ HasLaunchBar()

virtual **BOOL** HasLaunchBar ( ) const private virtual

Returns TRUE if the aircraft has a valid launch bar, FALSE otherwise.

§ HasTailhook()

virtual **BOOL** HasTailhook ( ) const private virtual

Returns TRUE if the aircraft has a valid tailhook, FALSE otherwise.

§ RetractLaunchBar()

virtual void RetractLaunchBar ( ) private virtual

Retracts the aircraft's launch bar.

§ RetractTailhook()

virtual void RetractTailhook ( ) private virtual

Retracts the aircraft's tailhook.

§ SetAileronPercent()

virtual void SetAileronPercent (double dPct) private virtual

Sets the aileron to a specific position (-1 left - +1 right)

§ SetElevatorPercent()

virtual void SetElevatorPercent (double dPct) private virtual

Sets the elevator to a specific position (-1 down - +1 up)

§ SetFlapsPercent()

virtual void SetFlapsPercent (double dPct) private virtual

Sets the flaps to a specific position (0 - +1 extended)

§ SetPushBack()

virtual void SetPushBack (**BOOL** bOn) private virtual

Triggers the ground handling simulation to move backwards. For example, pushing back from a gate. NOTE: Ground handling is assumed to be tightly coupled and calculated within the airplane simulation.

§ SetRudderPercent()

virtual void SetRudderPercent (double dPct) private virtual

Sets the rudder to a specific position (-1 left - +1 right)

§ SetSpoilersPercent()

virtual void SetSpoilersPercent (double dPct) private virtual

Sets the spoilers to a specific position (0 - +1 extended)

§ SetTaxiHeading()

virtual void SetTaxiHeading (float fHeading) private virtual

Sets the taxi heading (radians) NOTE: Ground handling is assumed to be tightly coupled and calculated within the airplane simulation.

§ SetTaxiSpeed()

virtual void SetTaxiSpeed (float fSpeed) private virtual

Sets the taxi speed (feet per second) NOTE: Ground handling is assumed to be tightly coupled and calculated within the airplane simulation.

§ SetThrottlePercent()

virtual void SetThrottlePercent (double dPct) private virtual

Sets the throttle to a specific position (0 - 1)

§ StopTaxi()

virtual void StopTaxi ( ) private virtual

Triggers the taxi operation to stop. NOTE: Ground handling is assumed to be tightly coupled and calculated within the airplane simulation.

§ UnfoldWings()

virtual void UnfoldWings ( ) private virtual

Unfolds the aircraft's wings, if available.

§ P3D::IHelicopterAIServiceV420

class P3D::IHelicopterAIServiceV420

Inherits IHelicopterAIServiceV01.

Private Member Functions

virtual double	**GetThrottlePercent** () const **PURE**
virtual double	**GetCollectivePercent** () const **PURE**
virtual double	**GetTorquePedalPercent** () const **PURE**
virtual double	**GetCyclicLateralPercent** () const **PURE**
virtual double	**GetCyclicLongitudinalPercent** () const **PURE**
virtual void	**SetThrottlePercent** (double dPct) **PURE**
virtual void	**SetCollectivePercent** (double dPct) **PURE**
virtual void	**SetTorquePedalPercent** (double dPct) **PURE**
virtual void	**SetCyclicLateralPercent** (double dPct) **PURE**
virtual void	**SetCyclicLongitudinalPercent** (double dPct) **PURE**

Member Function Documentation

§ GetCollectivePercent()

virtual double GetCollectivePercent ( ) const private virtual

Returns the collective position (0 - 1)

§ GetCyclicLateralPercent()

virtual double GetCyclicLateralPercent ( ) const private virtual

Returns the cyclic L/R position (0 - 1)

§ GetCyclicLongitudinalPercent()

virtual double GetCyclicLongitudinalPercent ( ) const private virtual

Returns the cyclic F/A position (0 - 1)

§ GetThrottlePercent()

virtual double GetThrottlePercent ( ) const private virtual

Returns the throttle position (0 - 1)

§ GetTorquePedalPercent()

virtual double GetTorquePedalPercent ( ) const

private virtual

Returns the torque pedals position (-1 - 1)

§ SetCollectivePercent()

virtual void SetCollectivePercent (double dPct)

private virtual

Sets the collective to a specific position (0 - 1)

§ SetCyclicLateralPercent()

virtual void SetCyclicLateralPercent (double dPct)

private virtual

Sets the cyclic L/R position (0 - 1)

§ SetCyclicLongitudinalPercent()

virtual void SetCyclicLongitudinalPercent (double dPct)

private virtual

Sets the cyclic F/A position (0 - 1)

§ SetThrottlePercent()

virtual void SetThrottlePercent (double dPct)

private virtual

Sets the throttle to a specific position (0 - 1)

§ SetTorquePedalPercent()

virtual void SetTorquePedalPercent (double dPct)

private virtual

Sets the torque pedals to a specific position (-1 - 1)

§ P3D::IGroundVehicleAIServiceV01

class P3D::IGroundVehicleAIServiceV01

This interface enables ground vehicle implementations to be utilized by Prepar3D's internal AI controllers.

Inherits IGroundVehicleAIService.

§ P3D::IWeaponsSystemV440

class P3D::IWeaponsSystemV440

Professional Plus Only

Interface to the Prepar3D native weapon system. Can also be used to implement a custom weapon system

Inherits IWeaponsSystemV430.

Private Member Functions

virtual **BOOL**	**GetIgnoreAttachmentForces** () const **PURE**
virtual void	**SetIgnoreAttachmentForces** (**BOOL** enabled) **PURE**
virtual **BOOL**	**GetIgnoreAttachmentWeight** () const **PURE**
virtual void	**SetIgnoreAttachmentWeight** (**BOOL** enabled) **PURE**
virtual UINT	**GetNumberOfStations** () const **PURE**
virtual UINT	**GetStationQuantity** (UINT iStationIndex) const **PURE**
virtual **BOOL**	**HasPylon** (UINT iStationIndex) const **PURE**
virtual UINT	**GetNumberOfPylonPoints** (UINT iStationIndex) const **PURE**
virtual HRESULT	**GetWeapon** (__in UINT iStationIndex, __in UINT iPylonIndex, __out UINT &uObjectId, __out IWeaponServiceV400 **ppWeapon) const **PURE**
virtual HRESULT	**GetPylon** (__in UINT iStationIndex, __out UINT &uObjectId, __out IPylonService **ppPylon) const **PURE**
virtual void	**SetStationLoadOut** (__in **UINT32** stationIndex, __in LPCTSTR pszWeaponTitle, __in **UINT32** roundsRemaining, __in **UINT32** roundsDefault, __in LPCTSTR pszPylonTitle) **PURE**
virtual **BOOL**	**IsSystemOn** () const **PURE**
virtual **BOOL**	**IsSystemArmed** () const **PURE**
virtual **BOOL**	**IsSafetyOn** () const **PURE**
virtual void	**ToggleSystem** () **PURE**
virtual void	**ToggleArmed** () **PURE**
virtual void	**ToggleSafety** () **PURE**
virtual void	**EngageTrigger** (**BOOL** bSingleShot) **PURE**
virtual void	**DisengageTrigger** () **PURE**
virtual void	**TriggerJettison** () **PURE**
virtual **BOOL**	**IsStationSelected** (UINT iStationIndex) const **PURE**
virtual void	**SelectNextStation** () **PURE**
virtual void	**SelectPreviousStation** () **PURE**
virtual void	**SetSelectedWeaponTypeIndex** (UINT uData) **PURE**
virtual void	**ToggleStation** (UINT iStationIndex) **PURE**
virtual void	**SelectStationOn** (UINT iStationIndex, **BOOL** bExclusiveOn) **PURE**
virtual void	**SelectStationOff** (UINT iStationIndex, **BOOL** bAllOff) **PURE**
virtual void	**SelectPylonPointOn** (UINT iStationIndex, UINT iPylonPoint, **BOOL** bExclusiveOn) **PURE**
virtual void	**SelectPylonPointOff** (UINT iStationIndex, UINT iPylonPoint, **BOOL** bAllOff) **PURE**
virtual void	**SelectNextWeapon** () **PURE**
virtual void	**SelectPreviousWeapon** () **PURE**
virtual void	**ResetWeapons** () **PURE**

Member Function Documentation

§ DisengageTrigger()

virtual void DisengageTrigger ( ) private virtual

Disengages a weapon system trigger

§ EngageTrigger()

virtual void EngageTrigger (**BOOL** bSingleShot) private virtual

Engages a weapon system trigger

Parameters

bSingleShot Sets max of one shot per trigger engage

Note

In order to work properly, **EngageTrigger()** cannot be called every frame. At least one frame must pass without a call to **EngageTrigger()** to allow the trigger state to reset.

§ GetIgnoreAttachmentForces()

virtual **BOOL** GetIgnoreAttachmentForces ( ) const private virtual

Gets the Global Attachment Setting for ignoring forces

§ GetIgnoreAttachmentWeight()

virtual **BOOL** GetIgnoreAttachmentWeight ( ) const private virtual

Gets the Global Attachment Setting for ignoring weight

§ GetNumberOfPylonPoints()

virtual UINT GetNumberOfPylonPoints (UINT iStationIndex) const private virtual

Gets the total number of pylon points that a station contains

§ GetNumberOfStations()

virtual UINT GetNumberOfStations ( ) const private virtual

Gets the number of stations available to the weapon system

§ GetPylon()

virtual HRESULT GetPylon (__in UINT iStationIndex,
__out UINT & uObjectId,
__out IPylonService ** ppPylon
) const private virtual

Gets a pylon

§ GetStationQuantity()

virtual UINT GetStationQuantity (UINT iStationIndex) const private virtual

Gets the the total number of weapons loaded at the given

§ GetWeapon()

virtual HRESULT GetWeapon (__in UINT iStationIndex,
__in UINT iPylonIndex,
__out UINT & uObjectId,
__out IWeaponServiceV400 ** ppWeapon
) const private virtual

Gets a weapon

§ HasPylon()

virtual **BOOL** HasPylon (UINT iStationIndex) const private virtual

Returns true if a station contains a pylon, false otherwise

§ IsSafetyOn()

virtual **BOOL** IsSafetyOn ( ) const private virtual

Returns whether or not system's safety is on.

§ IsStationSelected()

virtual **BOOL** IsStationSelected (UINT iStationIndex) const

private virtual

Returns true if a station if selected

§ IsSystemArmed()

virtual **BOOL** IsSystemArmed ( ) const

private virtual

Returns whether or not the system is armed.

§ IsSystemOn()

virtual **BOOL** IsSystemOn ( ) const

private virtual

Returns whether or not the system is on.

§ ResetWeapons()

virtual void ResetWeapons ( )

private virtual

Reset the weapon loadouts to their original state. This does not change station/pylon point selection.

§ SelectNextStation()

virtual void SelectNextStation ( )

private virtual

Selects the next station, even if empty

§ SelectNextWeapon()

virtual void SelectNextWeapon ( )

private virtual

Select the next available weapon regardless of type

§ SelectPreviousStation()

virtual void SelectPreviousStation ( ) private virtual

Selects the previous station, even if empty

§ SelectPreviousWeapon()

virtual void SelectPreviousWeapon ( ) private virtual

Select the previous available weapon regardless of type

§ SelectPylonPointOff()

```
virtual void SelectPylonPointOff ( UINT  iStationIndex,
                                   UINT  iPylonPoint,
                                   BOOL  bAllOff
                                 )
```
private virtual

Turns selection pylon point off

Parameters

bAllOff Turns all pylon points off at station

§ SelectPylonPointOn()

```
virtual void SelectPylonPointOn ( UINT  iStationIndex,
                                  UINT  iPylonPoint,
                                  BOOL  bExclusiveOn
                                )
```
private virtual

Turns selection pylon point on

Parameters

bExclusiveOn Turns all other pylon points off at station

§ SelectStationOff()

virtual void SelectStationOff (UINT iStationIndex,
BOOL bAllOff
) private virtual

Turns selected station off

Parameters

bAllOff Turns all stations off

§ SelectStationOn()

virtual void SelectStationOn (UINT iStationIndex,
BOOL bExclusiveOn
) private virtual

Turns selected station on

Parameters

bExclusiveOn Turns all other stations off

§ SetIgnoreAttachmentForces()

virtual void SetIgnoreAttachmentForces (BOOL enabled) private virtual

Sets the Global Attachment Setting for ignoring forces

§ SetIgnoreAttachmentWeight()

virtual void SetIgnoreAttachmentWeight (BOOL enabled) private virtual

Sets the Global Attachment Setting for ignoring forces

§ SetSelectedWeaponTypeIndex()

virtual void SetSelectedWeaponTypeIndex (UINT uData) private virtual

Selects the next station for the weapon type corresponding to the index defined in the WeaponSelectorTypes list in attachments.xml

§ SetStationLoadOut()

virtual void SetStationLoadOut (__in **UINT32** stationIndex,
__in LPCTSTR pszWeaponTitle,
__in **UINT32** roundsRemaining,
__in **UINT32** roundsDefault,
__in LPCTSTR pszPylonTitle
) private virtual

Sets the station weapon and pylon

§ ToggleArmed()

virtual void ToggleArmed ( ) private virtual

Arms and disarms a weapon system

§ ToggleSafety()

virtual void ToggleSafety ( ) private virtual

Toggles a weapon system safety on and off

§ ToggleStation()

virtual void ToggleStation (UINT iStationIndex) private virtual

Toggles the selection of the given station on and off

§ ToggleSystem()

virtual void ToggleSystem ( ) private virtual

Toggles a weapon system on and off

§ TriggerJettison()

virtual void TriggerJettison ( ) private virtual

Triggers jettison of a currently selected weapon system

Note

In order to work properly, **TriggerJettison()** cannot be called every frame.At least one frame must pass without a call to **TriggerJettison()** to allow the trigger state to reset.

§ P3D::IWeaponServiceV420

class P3D::IWeaponServiceV420

Professional Plus Only

Interface for getting weapon parameters for this object

Inherits IWeaponServiceV400.

Private Member Functions

virtual HRESULT	**SetIsAttachedToOwner** (**BOOL** bAttached, UINT uOwnerId, **BOOL** bJettisoned) **PURE**
virtual **BOOL**	**IsAttachedToOwner** () const **PURE**
virtual UINT	**GetOwnerId** () const **PURE**
virtual HRESULT	**GetAttachOffsetFeet** (__out **P3D::DXYZ** &vOffset) const **PURE**
virtual **BOOL**	**CanWeaponBeReleased** () const **PURE**
virtual float	**GetAerodynamicsDragCoefficient** (float fMach) const **PURE**
virtual HRESULT	**GetType** (__out LPWSTR pszType, __in unsigned int uLength) const **PURE**
virtual **BOOL**	**GetCausesWeaponCollision** () const **PURE**

Member Function Documentation

§ CanWeaponBeReleased()

virtual **BOOL** CanWeaponBeReleased ( ) const [private] [virtual]

Called upon firing of weapon. The weapon implementation can block being released

§ GetAerodynamicsDragCoefficient()

virtual float GetAerodynamicsDragCoefficient (float fMach) const [private] [virtual]

Gets the aerodynamic drag for the weapon loadout UI in SimDirector

§ GetAttachOffsetFeet()

virtual HRESULT GetAttachOffsetFeet (__out **P3D::DXYZ** & vOffset) const [private] [virtual]

Gets the offset on the weapon in which it is attached to the parent

§ GetCausesWeaponCollision()

virtual **BOOL** GetCausesWeaponCollision ( ) const

private virtual

Gets whether or not the weapon should collide with other weapons

§ GetOwnerId()

virtual UINT GetOwnerId ( ) const

private virtual

ID of object in which weapon is attached (should remain valid even after detached)

§ GetType()

virtual HRESULT GetType (__out LPWSTR pszType,
__in unsigned int uLength
) const

private virtual

Gets the string type of weapon (e.g. "AAM", "SAM"). These are defined for native weapons in sim.cfg. It is dependent on the weapon implementation, but can be used for arbitrary categorization

§ IsAttachedToOwner()

virtual **BOOL** IsAttachedToOwner ( ) const

private virtual

Is weapon currently attached to parent object

§ SetIsAttachedToOwner()

virtual HRESULT SetIsAttachedToOwner (**BOOL** bAttached,
UINT uOwnerId,
BOOL bJettisoned
)

private virtual

Called from weapon system when attached, jettisoned, or fired (0 = invalid id)

§ P3D::ICountermeasureSystemV01

class P3D::ICountermeasureSystemV01

Professional Plus Only

Used to implement or query for a countermeasure system

Inherits ICountermeasureSystem.

Private Member Functions

virtual UINT	**GetNumberOfStations** () const **PURE**
virtual UINT	**GetStationQuantity** (UINT iStationIndex) const **PURE**
virtual HRESULT	**GetCountermeasure** (__in UINT iStationIndex, __in UINT iPylonIndex, __out UINT &uObjectId, __out ICountermeasureService **ppCM) const **PURE**
virtual **BOOL**	**IsSystemOn** () const **PURE**
virtual **BOOL**	**IsSystemArmed** () const **PURE**
virtual void	**ToggleSystem** () **PURE**
virtual void	**ToggleArmed** () **PURE**
virtual void	**EngageTrigger** (**BOOL** bSingleShot) **PURE**
virtual void	**DisengageTrigger** () **PURE**
virtual **BOOL**	**IsStationSelected** (UINT iStationIndex) const **PURE**
virtual void	**SelectNextStation** () **PURE**
virtual void	**SelectPreviousStation** () **PURE**
virtual void	**ToggleStation** (UINT iStationIndex) **PURE**
virtual void	**SelectStationOn** (UINT iStationIndex, **BOOL** bExclusiveOn) **PURE**
virtual void	**SelectStationOff** (UINT iStationIndex, **BOOL** bAllOff) **PURE**
virtual void	**SelectNextCountermeasure** () **PURE**
virtual void	**SelectPreviousCountermeasure** () **PURE**
virtual void	**ResetCountermeasures** () **PURE**

Member Function Documentation

§ DisengageTrigger()

virtual void DisengageTrigger ( ) private virtual

Detriggers a countermeasure

§ EngageTrigger()

virtual void EngageTrigger (**BOOL** bSingleShot) private virtual

Triggers a countermeasure

Parameters

bSingleShot Sets max of one shot per trigger engage

§ GetCountermeasure()

virtual HRESULT GetCountermeasure (__in UINT iStationIndex,
__in UINT iPylonIndex,
__out UINT & uObjectId,
__out ICountermeasureService ** ppCM
) const private virtual

Gets a countermeasure system

§ GetNumberOfStations()

virtual UINT GetNumberOfStations ( ) const private virtual

Gets number of countermeasure stations

§ GetStationQuantity()

virtual UINT GetStationQuantity (UINT iStationIndex) const private virtual

§ IsStationSelected()

virtual **BOOL** IsStationSelected (UINT iStationIndex) const private virtual

Returns true if a station if selected

§ IsSystemArmed()

virtual **BOOL** IsSystemArmed ( ) const private virtual

Returns whether or not the system is armed.

§ IsSystemOn()

virtual **BOOL** IsSystemOn ( ) const

private virtual

Returns whether or not the system is on.

§ ResetCountermeasures()

virtual void ResetCountermeasures ( )

private virtual

Reset all countermeasures

§ SelectNextCountermeasure()

virtual void SelectNextCountermeasure ( )

private virtual

Select the next countermeasure

§ SelectNextStation()

virtual void SelectNextStation ( )

private virtual

Selects the next station

§ SelectPreviousCountermeasure()

virtual void SelectPreviousCountermeasure ( )

private virtual

Select the previous countermeasure

§ SelectPreviousStation()

virtual void SelectPreviousStation ( )

private virtual

Selects the previous station

§ SelectStationOff()

virtual void SelectStationOff (UINT iStationIndex,
BOOL bAllOff
) private virtual

Turns selected station off

Parameters

bAllOff Turns all stations off

§ SelectStationOn()

virtual void SelectStationOn (UINT iStationIndex,
BOOL bExclusiveOn
) private virtual

Turns selected station on

Parameters

bExclusiveOn Turns all other stations off

§ ToggleArmed()

virtual void ToggleArmed ( ) private virtual

Arms and disarms a countermeasure system

§ ToggleStation()

virtual void ToggleStation (UINT iStationIndex) private virtual

Toggles a specific station on and off

§ ToggleSystem()

virtual void ToggleSystem ( ) private virtual

Toggles a countermeasure system on and off

§ P3D::ICountermeasureServiceV02

class P3D::ICountermeasureServiceV02

Professional Plus Only

Interface for getting countermeasure parameters for this object

Inherits ICountermeasureServiceV01.

Private Member Functions

virtual HRESULT	**SetIsAttachedToOwner** (**BOOL** bAttached, UINT uOwnerId) **PURE**
virtual **BOOL**	**IsAttachedToOwner** () const **PURE**
virtual UINT	**GetOwnerId** () const **PURE**
virtual HRESULT	**GetAttachOffsetFeet** (__out **P3D::DXYZ** &vOffset) const **PURE**
virtual **BOOL**	**GetCausesWeaponCollision** () const **PURE**

Member Function Documentation

§ GetAttachOffsetFeet()

virtual HRESULT GetAttachOffsetFeet (__out **P3D::DXYZ** & vOffset) const [private] [virtual]

Gets the offset on the weapon in which it is attached to the parent

§ GetCausesWeaponCollision()

virtual **BOOL** GetCausesWeaponCollision ( ) const [private] [virtual]

Gets whether or not the countermeasure should collide with weapons

§ GetOwnerId()

virtual UINT GetOwnerId ( ) const [private] [virtual]

ID of object in which countermeasure is attached (shoud remain valid even after detached)

§ IsAttachedToOwner()

virtual **BOOL** IsAttachedToOwner ( ) const [private] [virtual]

Is weapon currently attached to parent object

§ SetIsAttachedToOwner()

virtual HRESULT SetIsAttachedToOwner (**BOOL** bAttached,
UINT uOwnerId
) private virtual

Called from countermeasure system when attached, jettisoned, or fired (0 = invalid id)

§ P3D::IGunSystemV440

class P3D::IGunSystemV440

Professional Plus Only

Interface for getting gun system parameters for this object

Inherits IGunSystemV400.

Private Member Functions

virtual **BOOL**	**AddGun** (__in **IGunV400** *pGun, __in UINT stationIndex) **PURE**
virtual **BOOL**	**RemoveGun** (__in UINT stationIndex) **PURE**
virtual const **IGunV400** *	**GetGun** (__in UINT stationIndex) const **PURE**
virtual void	**SetIsActive** (__in **BOOL** isActive) **PURE**
virtual **BOOL**	**GetIsActive** () const **PURE**
virtual **BOOL**	**GetIsFiring** () const **PURE**
virtual void	**SetIsAutomatedGunsEnabled** (__in **BOOL** isAutomatedGunsEnabled) **PURE**
virtual **BOOL**	**GetIsAutomatedGunsEnabled** () const **PURE**
virtual void	**EngageTrigger** () **PURE**
virtual void	**DisengageTrigger** () **PURE**
virtual UINT	**GetNumberOfStations** () const **PURE**
virtual UINT	**GetNumberOfGuns** () const **PURE**
virtual **BOOL**	**IsStationIndexValid** (__in UINT stationIndex) const **PURE**
virtual **BOOL**	**GetIsGunPresentAtStation** (__in UINT stationIndex) const **PURE**
virtual **BOOL**	**GetIsGunSelectedAtStation** (__in UINT stationIndex) const **PURE**
virtual **BOOL**	**IsSystemOn** () const **PURE**
virtual void	**ToggleSystem** () **PURE**
virtual void	**ToggleStation** (__in UINT stationIndex) **PURE**
virtual void	**SelectStationOn** (UINT iStationIndex, **BOOL** bExclusiveOn) **PURE**
virtual void	**SelectStationOff** (UINT iStationIndex, **BOOL** bAllOff) **PURE**
virtual void	**CreateTracerEffect** (__in __notnull const WCHAR *pszEffectName, __in const **DXYZ** *pvLonAltLat, __in const **DXYZ** *pvPHB, __out void **ppEffect) **PURE**
virtual void	**MoveTracerEffect** (__in const **DXYZ** *pvLonAltLat, __in const **DXYZ** *pvPHB, __in void *pEffect) **PURE**
virtual void	**DestroyTracerEffect** (__in void *pEffect) **PURE**
virtual **BOOL**	**CheckBulletCollision** (__in const **DXYZ** *pvLonAltLat, __in const **DXYZ** *pvDeltaOffset, __out **COLLISIONTYPE** &eCollision, __out IUnknown **ppUnkObjectHit) **PURE**
virtual void	**ToggleAutomaticGuns** () **PURE**
virtual void	**SetPitchPercent** (float fPercent) **PURE**
virtual void	**SetHeadingPercent** (float fPercent) **PURE**
virtual void	**IncrementLeft** () **PURE**
virtual void	**IncrementRight** () **PURE**
virtual void	**IncrementUp** () **PURE**
virtual void	**IncrementDown** () **PURE**
virtual void	**IncrementLeftAndUp** () **PURE**
virtual void	**IncrementLeftAndDown** () **PURE**
virtual void	**IncrementRightAndUp** () **PURE**

virtual void	**IncrementRightAndDown** () **PURE**
virtual void	**ResetGuns** () **PURE**
virtual void	**SetCrosshairTarget** (__in const double &lat, __in const double &lon, __in const double &alt) **PURE**
virtual void	**ClearCrosshairTarget** () **PURE**

Member Function Documentation

§ AddGun()

virtual **BOOL** AddGun (__in **IGunV400** * pGun,
__in UINT stationIndex
)

private virtual

Adds an IGun implementation to the internal GunSystem at the given station. Returns TRUE if successfully added, and FALSE otherwise. If successfully added, the IGun reference count will be increased by one.

§ CheckBulletCollision()

virtual **BOOL** CheckBulletCollision (__in const **DXYZ** * pvLonAltLat,
__in const **DXYZ** * pvDeltaOffset,
__out **COLLISIONTYPE** & eCollision,
__out IUnknown ** ppUnkObjectHit
)

private virtual

§ ClearCrosshairTarget()

virtual void ClearCrosshairTarget ( )

private virtual

§ CreateTracerEffect()

virtual void CreateTracerEffect (__in __notnull const WCHAR * pszEffectName,
__in const **DXYZ** * pvLonAltLat,
__in const **DXYZ** * pvPHB,
__out void ** ppEffect
)

private virtual

Creates a tracer visual effect at the given location. Longitude, altitude, and latitude units are radians and feet. Pitch, heading, and bank units are radians

§ DestroyTracerEffect()

virtual void DestroyTracerEffect (__in void * pEffect) private virtual

Removes a tracer effect if it is still alive Performs collision detection at a location in the form longitude, altitude, and latitude, where the units are radians and feet. The delta offset should be given from this location in feet

§ DisengageTrigger()

virtual void DisengageTrigger ( ) private virtual

Disengages the trigger of a gun system

§ EngageTrigger()

virtual void EngageTrigger ( ) private virtual

Engages the trigger of a gun system

§ GetGun()

virtual const **IGunV400*** GetGun (__in UINT stationIndex) const private virtual

Gets a pointer to an IGun implementation at the given station if it exists, otherwise returns NULL.

§ GetIsActive()

virtual **BOOL** GetIsActive ( ) const private virtual

Gets the active state of the gun system

§ GetIsAutomatedGunsEnabled()

virtual **BOOL** GetIsAutomatedGunsEnabled ( ) const private virtual

Gets the state of the automatic gun targeting.

§ GetIsFiring()

virtual **BOOL** GetIsFiring ( ) const private virtual

Gets if a gun system is currently firing

§ GetIsGunPresentAtStation()

virtual **BOOL** GetIsGunPresentAtStation (__in UINT stationIndex) const

private virtual

Checks if a station contains a gun

§ GetIsGunSelectedAtStation()

virtual **BOOL** GetIsGunSelectedAtStation (__in UINT stationIndex) const

private virtual

Checks if a station contains a selected gun

§ GetNumberOfGuns()

virtual UINT GetNumberOfGuns ( ) const

private virtual

Returns the number of guns loaded by the gun system

§ GetNumberOfStations()

virtual UINT GetNumberOfStations ( ) const

private virtual

Returns the number of stations available to the gun system

§ IncrementDown()

virtual void IncrementDown ( )

private virtual

Moves gun down

§ IncrementLeft()

virtual void IncrementLeft ( )

private virtual

Moves gun left

§ IncrementLeftAndDown()

virtual void IncrementLeftAndDown ( ) private virtual

Moves gun left and down

§ IncrementLeftAndUp()

virtual void IncrementLeftAndUp ( ) private virtual

Moves gun left and up

§ IncrementRight()

virtual void IncrementRight ( ) private virtual

Moves gun right

§ IncrementRightAndDown()

virtual void IncrementRightAndDown ( ) private virtual

Moves gun right and down

§ IncrementRightAndUp()

virtual void IncrementRightAndUp ( ) private virtual

Moves gun right and up

§ IncrementUp()

virtual void IncrementUp ( ) private virtual

Moves gun up

§ IsStationIndexValid()

virtual **BOOL** IsStationIndexValid (__in UINT stationIndex) const private virtual

Checks if an index refers to an existing station

§ IsSystemOn()

virtual **BOOL** IsSystemOn ( ) const

private virtual

Returns whether or not the system is on.

§ MoveTracerEffect()

virtual void MoveTracerEffect (__in const **DXYZ** * pvLonAltLat,
__in const **DXYZ** * pvPHB,
__in void * pEffect
)

private virtual

Moves an existing visual effect to the given location. Same position and rotation values above apply

§ RemoveGun()

virtual **BOOL** RemoveGun (__in UINT stationIndex)

private virtual

Removes an IGun implementation from the internal GunSystem at the given station. Returns TRUE if successfully removed, and FALSE otherwise. If successfully removed, the IGun reference count will be decreased by one.

§ ResetGuns()

virtual void ResetGuns ( )

private virtual

Resets the gun loadout to its original state. This does not change station/pylon point selection.

§ SelectStationOff()

virtual void SelectStationOff (UINT iStationIndex,
BOOL bAllOff
)

private virtual

Turns selected station off

Parameters

bAllOff Turns all stations off

§ SelectStationOn()

virtual void SelectStationOn (UINT iStationIndex,
BOOL bExclusiveOn
) private virtual

Turns selected station on

Parameters

bExclusiveOn Turns all other stations off

§ SetCrosshairTarget()

virtual void SetCrosshairTarget (__in const double & lat,
__in const double & lon,
__in const double & alt
) private virtual

§ SetHeadingPercent()

virtual void SetHeadingPercent (float fPercent) private virtual

Change the heading of guns based on percentage. [-1, 1]

§ SetIsActive()

virtual void SetIsActive (__in BOOL isActive) private virtual

Sets a gun system to active (true) or inactive (false)

§ SetIsAutomatedGunsEnabled()

virtual void SetIsAutomatedGunsEnabled (__in BOOL isAutomatedGunsEnabled) private virtual

Sets the state of the automatic gun targeting.

§ SetPitchPercent()

virtual void SetPitchPercent (float fPercent) private virtual

Change the pitch of guns based on percentage. [-1, 1]

§ ToggleAutomaticGuns()

virtual void ToggleAutomaticGuns ( ) private virtual

Toggles the state of automatic gun targeting on and off

§ ToggleStation()

virtual void ToggleStation (__in UINT stationIndex) private virtual

Toggles a station on and off

§ ToggleSystem()

virtual void ToggleSystem ( ) private virtual

Toggles a gun system on and off

§ P3D::IGunV400

class P3D::IGunV400

Professional Plus Only

Interface for getting gun parameters for this object

Inherits IUnknown.

Private Member Functions

virtual void	**Simulate** (__in double deltaT) **PURE**
virtual HRESULT	**Fire** (__in double deltaT) **PURE**
virtual void	**Purge** () **PURE**
virtual void	**Stop** () **PURE**
virtual void	**SetRoundsRemaining** (__in UINT ammoCount) **PURE**
virtual UINT	**GetRoundsRemaining** () const **PURE**
virtual void	**ResetRounds** () **PURE**
virtual const WCHAR *	**GetName** () const **PURE**
virtual const WCHAR *	**GetGunType** () const **PURE**
virtual void	**Rotate** (__in double xAxisOffset, __in double yAxisOffset, __in double deltaT) **PURE**
virtual void	**ProcessTargeting** (__in const **P3D::DXYZ** &targetLla, __in const **P3D::DXYZ** &targetBodyVelocity, __in const **P3D::DXYZ** &targetBodyAcceleration, __in const **P3D::DXYZ** &targetOrientation, __in double deltaT) **PURE**

Member Function Documentation

§ Fire()

virtual HRESULT Fire (__in double deltaT) private virtual

Called once per step on selected guns. **Fire()** will be repeatedly called while the trigger is engaged. Users can use an HRESULT return type

§ GetGunType()

virtual const WCHAR* GetGunType ( ) const private virtual

Gets the type of a gun

§ GetName()

virtual const WCHAR* GetName ( ) const private virtual

Gets the name of a gun

§ GetRoundsRemaining()

virtual UINT GetRoundsRemaining ( ) const

private virtual

Gets the total number of rounds in a gun

§ ProcessTargeting()

```
virtual void ProcessTargeting ( __in const P3D::DXYZ &  targetLla,
                                __in const P3D::DXYZ &  targetBodyVelocity,
                                __in const P3D::DXYZ &  targetBodyAcceleration,
                                __in const P3D::DXYZ &  targetOrientation,
                                __in double             deltaT
                              )
```

private virtual

called if automatic guns are enabled, providing the developer with information on the target. Target position is longitude, altitude and latitude in that order, where the units are radians and feet. Target velocity and acceleration are in feet per second while orientation is in world coordinate radians.

§ Purge()

virtual void Purge ( )

private virtual

Called on all guns when the trigger is released

§ ResetRounds()

virtual void ResetRounds ( )

private virtual

Resets the total number of rounds in a gun. Called on each gun when ResetGuns() is called by the GunSystem.

§ Rotate()

```
virtual void Rotate ( __in double  xAxisOffset,
                      __in double  yAxisOffset,
                      __in double  deltaT
                    )
```

private virtual

Called when the user provides input to rotate guns. Values passed should be expected to be mapped from -1.0 to 1.0.

§ SetRoundsRemaining()

virtual void SetRoundsRemaining (__in UINT ammoCount) private virtual

Sets the total number of rounds in a gun

§ Simulate()

virtual void Simulate (__in double deltaT) private virtual

Called once per step on all guns. Delta time is in seconds

§ Stop()

virtual void Stop ( ) private virtual

Called on all guns while there are no user inputs for gun rotations and automatic targeting is disabled

§ P3D::IFireControlSystemV01

class P3D::IFireControlSystemV01

Professional Plus Only

Interface for getting fire control system parameters for this object

Inherits IFireControlSystem.

Private Member Functions

virtual UINT	**GetSelectedTargetID** () const **PURE**
virtual void	**SetSelectedTargetID** (UINT id) **PURE**
virtual HRESULT	**GetSelectedTargetMissionID** (__out GUID &guid) const **PURE**
virtual HRESULT	**SetSelectedTargetMissionID** (__in const GUID &guid) **PURE**
virtual **BOOL**	**GetTargetLLA** (__out **P3D::DXYZ** &vLLA) const **PURE**
virtual void	**SetTargetLLA** (__in const **P3D::DXYZ** &vLLA) **PURE**

Member Function Documentation

§ GetSelectedTargetID()

virtual UINT GetSelectedTargetID ( ) const [private] [virtual]

Get the ID of the target selected by the fire control system

§ GetSelectedTargetMissionID()

virtual HRESULT GetSelectedTargetMissionID (__out GUID & guid) const [private] [virtual]

Gets the instance ID of the target selected by the fire control system (Structured scenarios with objects only)

§ GetTargetLLA()

virtual **BOOL** GetTargetLLA (__out **P3D::DXYZ** & vLLA) const [private] [virtual]

If the fire control system's target is a latitude/longitude/altitude, this will return that position. Otherwise the return will be FALSE. (radians/radians/feet)

§ SetSelectedTargetID()

virtual void SetSelectedTargetID (UINT id) [private] [virtual]

Sets the fire control system target by object

§ SetSelectedTargetMissionID()

virtual HRESULT SetSelectedTargetMissionID (__in const GUID & guid) private virtual

Sets the fire control system target by instance ID (Missions only)

§ SetTargetLLA()

virtual void SetTargetLLA (__in const **P3D::DXYZ** & vLLA) private virtual

Sets the fire control system's target to be a latitude/longitude/altitude. (radians/radians/feet)

§ P3D::IGuidanceSystemV01

class P3D::IGuidanceSystemV01

Professional Plus Only

Interface for getting guidance parameters for this object

Inherits IGuidanceSystem.

Private Member Functions

virtual void	**SetTargetObjectID** (UINT targetedObjectID) **PURE**
virtual UINT	**GetTargetObjectID** () const **PURE**
virtual void	**SetTargetLLA** (__in const **P3D::DXYZ** &vLLA) **PURE**
virtual **BOOL**	**GetTargetLLA** (__out **P3D::DXYZ** &vLLA) const **PURE**

Member Function Documentation

§ GetTargetLLA()

virtual **BOOL** GetTargetLLA (__out **P3D::DXYZ** & vLLA) const [private] [virtual]

If the guidance system's target is a latitude/longitude/altitude, this will return that position. Otherwise the return will be FALSE. (radians/radians/feet)

§ GetTargetObjectID()

virtual UINT GetTargetObjectID ( ) const [private] [virtual]

Gets the ID of a target

§ SetTargetLLA()

virtual void SetTargetLLA (__in const **P3D::DXYZ** & vLLA) [private] [virtual]

sets the guidance system's target to be a latitude/longitude/altitude. (radians/radians/feet)

§ SetTargetObjectID()

virtual void SetTargetObjectID (UINT targetedObjectID) [private] [virtual]

Sets the ID of a target

§ P3D::IPylonServiceV01

class P3D::IPylonServiceV01

Professional Plus Only

Interface for getting parameters for a weapon pylon

Inherits IPylonService.

Private Member Functions

virtual HRESULT	**SetOwnerId** (__in UINT uOwnerId) **PURE**
virtual UINT	**GetOwnerId** () const **PURE**
virtual HRESULT	**GetAttachOffsetFeet** (__out **P3D::DXYZ** &vOffset) const **PURE**

Member Function Documentation

§ GetAttachOffsetFeet()

virtual HRESULT GetAttachOffsetFeet (__out **P3D::DXYZ** & vOffset) const [private] [virtual]

Gets the offset on the pylon in which it is attached to the parent

§ GetOwnerId()

virtual UINT GetOwnerId ( ) const [private] [virtual]

Gets the ID of the object that the pylon is attached

§ SetOwnerId()

virtual HRESULT SetOwnerId (__in UINT uOwnerId) [private] [virtual]

Sets the ID of the object that the pylon is attached

§ P3D::ArticulatedPart

class P3D::ArticulatedPart

Class Members		
float	m_fPadding	
float	m_fParameterValue	
unsigned int	m_uiParameterType	
unsigned short	m_usAttachedToId	
unsigned char	m_yChangeIndicator	
unsigned char	m_yRecordType	

§ P3D::ArticulatedParameter

class P3D::ArticulatedParameter

Class Members		
union **ArticulatedParameter**	__unnamed__	

§ P3D::IPduBuilderV440

class P3D::IPduBuilderV440

This interface allows developers to build PDU's on a per-field basis.

Remarks

It is the developer's responsibility to fill in the PDU header as well as any necessary data.

Inherits IUnknown.

Private Member Functions

virtual void	**WriteChar** (__in char c) **PURE**
virtual void	**WriteUChar** (__in unsigned char uc) **PURE**
virtual void	**WriteFloat** (__in float f) **PURE**
virtual void	**WriteDouble** (__in double d) **PURE**
virtual void	**WriteInt** (__in int i) **PURE**
virtual void	**WriteUInt** (__in unsigned int u) **PURE**
virtual void	**WriteLong** (__in long l) **PURE**
virtual void	**WriteULong** (__in unsigned long ul) **PURE**
virtual void	**WriteLongLong** (__in long long ll) **PURE**
virtual void	**WriteUShort** (__in unsigned short us) **PURE**
virtual void	**WriteShort** (__in short s) **PURE**
virtual int	**GetSize** () const **PURE**

Member Function Documentation

§ GetSize()

virtual int GetSize ( ) const — private virtual

Returns the current size of the PDU in bytes.

§ WriteChar()

virtual void WriteChar (__in char c) — private virtual

Adds an 8-bit signed byte to the packet.

§ WriteDouble()

virtual void WriteDouble (__in double d) — private virtual

Adds a 64-bit floating point to the packet.

§ WriteFloat()

virtual void WriteFloat (__in float f) private virtual

Adds a 32-bit floating point to the packet.

§ WriteInt()

virtual void WriteInt (__in int i) private virtual

Adds a 32-bit signed integer to the packet.

§ WriteLong()

virtual void WriteLong (__in long l) private virtual

Adds a 32-bit signed long to the packet.

§ WriteLongLong()

virtual void WriteLongLong (__in long long ll) private virtual

Adds a 64-bit signed long to the packet.

§ WriteShort()

virtual void WriteShort (__in short s) private virtual

Adds a 16-bit signed short to the packet.

§ WriteUChar()

virtual void WriteUChar (__in unsigned char uc) private virtual

Adds an 8-bit unsigned short to the packet.

§ WriteUInt()

virtual void WriteUInt (__in unsigned int u) private virtual

Adds a 32-bit unsigned integer to the packet.

§ WriteULong()

virtual void WriteULong (__in unsigned long ul) private virtual

Adds a 32-bit unsigned long to the packet.

§ WriteUShort()

virtual void WriteUShort (__in unsigned short us) private virtual

Adds a 16-bit unsigned short to the packet.

§ P3D::IPduReaderV440

class P3D::IPduReaderV440

This interface allows developers to read PDU's on a per-field basis.

Inherits IUnknown.

Private Member Functions

virtual char	**ReadChar** () **PURE**
virtual unsigned char	**ReadUChar** () **PURE**
virtual float	**ReadFloat** () **PURE**
virtual double	**ReadDouble** () **PURE**
virtual int	**ReadInt** () **PURE**
virtual unsigned int	**ReadUInt** () **PURE**
virtual long	**ReadLong** () **PURE**
virtual unsigned long	**ReadULong** () **PURE**
virtual long long	**ReadLongLong** () **PURE**
virtual unsigned short	**ReadUShort** () **PURE**
virtual short	**ReadShort** () **PURE**
virtual const char *	**GetRawData** () const **PURE**
virtual UINT	**GetSize** () const **PURE**

Member Function Documentation

§ GetRawData()

virtual const char* GetRawData ( ) const private virtual

Returns the current data inside of the PDU.

§ GetSize()

virtual UINT GetSize ( ) const private virtual

Returns the current size of the PDU in bytes.

§ ReadChar()

virtual char ReadChar ( ) private virtual

Reads an 8-bit signed byte to the packet.

§ ReadDouble()

virtual double ReadDouble ( ) private virtual

Reads a 64-bit floating point to the packet.

§ ReadFloat()

virtual float ReadFloat ( ) private virtual

Reads a 32-bit floating point to the packet.

§ ReadInt()

virtual int ReadInt ( ) private virtual

Reads a 32-bit signed integer to the packet.

§ ReadLong()

virtual long ReadLong ( ) private virtual

Reads a 32-bit signed long to the packet.

§ ReadLongLong()

virtual long long ReadLongLong ( ) private virtual

Reads a 64-bit signed long to the packet.

§ ReadShort()

virtual short ReadShort ( ) private virtual

Reads a 16-bit signed short to the packet.

§ ReadUChar()

virtual unsigned char ReadUChar ( ) private virtual

Reads an 8-bit unsigned short to the packet.

§ ReadUInt()

virtual unsigned int ReadUInt ( ) private virtual

Reads a 32-bit unsigned integer to the packet.

§ ReadULong()

virtual unsigned long ReadULong ( ) private virtual

Reads a 32-bit unsigned long to the packet.

§ ReadUShort()

virtual unsigned short ReadUShort ( ) private virtual

Reads a 16-bit unsigned short to the packet.

§ P3D::IPduCallbackV440

class P3D::IPduCallbackV440

This interface allows developers to create PDU's to be sent or received.

Inherits IUnknown.

Private Member Functions

virtual HRESULT	**OnSend** (__in **IPduReaderV440** *pReader, __in BYTE uPduType) **PURE**
virtual HRESULT	**OnReceive** (__in **IPduReaderV440** *pReader, __in BYTE uPduType) **PURE**

Member Function Documentation

§ OnReceive()

virtual HRESULT OnReceive (__in **IPduReaderV440** * pReader,
__in BYTE uPduType
) private virtual

Plugins should implement this function to receive callbacks when Prepar3D has received a packet.

Remarks

Returning anything other than S_OK will prevent the packet from being processed within Prepar3D.

§ OnSend()

virtual HRESULT OnSend (__in **IPduReaderV440** * pReader,
__in BYTE uPduType
) private virtual

Plugins should implement this function to receive callbacks when Prepar3D is about to send a packet.

Remarks

Returning anything other than S_OK will prevent the packet from being sent over the network.

§ P3D::IDISManagerV450

class P3D::IDISManagerV450

Professional Plus Only

This service allows the developer to interact and retrieve information with a distributed interactive simulation (DIS) session. Developers integrating with this interface should be familiar with and are expected to follow DIS IEEE standards. This service is provided by the **IPdk** interface.

Inherits IDISManagerV440.

Private Member Functions

virtual **BOOL**	**IsConnected** () const **PURE**
virtual HRESULT	**GetEntityTypeById** (__in **UINT32** uID, __out **P3D::EntityType** &**EntityType**) const **PURE**
virtual HRESULT	**NotifyMunitionFired** (__in **UINT32** uAttackerID, __in **UINT32** uTargetID, __in **UINT32** uMunitionID, __in const **P3D::EntityType** &**EntityType**, __in const **P3D::DXYZ** &xyzLonAltLat, __in const **P3D::DXYZ** &xyzLinearVelocity, __in unsigned short usWarheadType, __in unsigned short usFuseType, __in unsigned short usQuantity, __in unsigned short usRate, __in float fRange, __inout unsigned short &usEventID) **PURE**
virtual HRESULT	**NotifyMunitionDetonated** (__in **UINT32** uAttackerID, __in **UINT32** uTargetID, __in **UINT32** uMunitionID, __in const **P3D::EntityType** &**EntityType**, __in unsigned short usEventID, __in const **P3D::DXYZ** &xyzLonAltLat, __in const **P3D::DXYZ** &xyzLinearVelocity, __in unsigned short usWarheadType, __in unsigned short usFuseType, __in unsigned short usQuantity, __in unsigned short usRate, __in unsigned char yDetonationResult) **PURE**
virtual HRESULT	**GetEntityIdByObjectId** (__in **UINT32** uObjectId, __out unsigned short &usSiteId, __out unsigned short &usApplicationId, __out unsigned short &usEntityId) **PURE**
virtual HRESULT	**GetObjectIdByEntityId** (__in unsigned short usSiteId, __in unsigned short usApplicationId, __in unsigned short usEntityId, __out **UINT32** &uObjectId) **PURE**
virtual **P3D::IPduBuilderV440** *	**CreatePdu** () **PURE**
virtual HRESULT	**IssuePdu** (**P3D::IPduBuilderV440** *pPduBuilder) **PURE**
virtual HRESULT	**RegisterPduCallback** (__in BYTE yPduType, __in __notnull **IPduCallbackV440** *pCallback) **PURE**
virtual HRESULT	**UnregisterPduCallback** (__in BYTE yPduType, __in __notnull **IPduCallbackV440** *pCallback) **PURE**
virtual HRESULT	**SetDisableReceive** (__in **BOOL** bDisableReceive) **PURE**
virtual HRESULT	**SetDisableSend** (__in **BOOL** bDisableSend) **PURE**
virtual USHORT	**GetSiteId** () const **PURE**
virtual USHORT	**GetApplicationId** () const **PURE**
virtual BYTE	**GetExerciseId** () const **PURE**
virtual USHORT	**GetEventId** () **PURE**
virtual int	**GetWallClockHour** () const **PURE**
virtual UINT	**GetWallTimestamp** () const **PURE**
virtual int	**GetSimClockHour** () const **PURE**
virtual UINT	**GetSimTimestamp** () const **PURE**

Member Function Documentation

§ CreatePdu()

virtual **P3D::IPduBuilderV440*** CreatePdu ( ) private virtual

Returns an **IPduBuilderV440** interface with a reference count of 1. This interface can be used to build PDU's to be used with the IssuePdu function. Developers should release this object after it has been issued using the IssuePdu function.

Sample implementation:

```
P3D::IPduBuilderV440* pPdu = spDIS->CreatePdu();

// Write PDU header
pPdu->WriteUChar(6);
pPdu->WriteUChar(spDIS->GetExerciseId());
pPdu->WriteUChar(1);
pPdu->WriteUChar(1);
pPdu->WriteUInt(spDIS->GetSimTimestamp());
// ...

// Write remaining PDU specific data
// ...

spDIS->IssuePdu(pPdu);

pPdu->Release();
pPdu = nullptr;
```

Remarks

It is the developer's responsibility to fill in the PDU header as well as any necessary data.

§ GetApplicationId()

virtual USHORT GetApplicationId ( ) const private virtual

Returns the session's current application id.

§ GetEntityIdByObjectId()

virtual HRESULT GetEntityIdByObjectId (__in **UINT32** uObjectId,
__out unsigned short & usSiteId,
__out unsigned short & usApplicationId,
__out unsigned short & usEntityId
) private virtual

Returns the entity identifier for the given object id if successful.

Parameters

uObjectId	The object id of the request
usSiteId	The site id of the entity identifier
usApplicationId	The application id of the entity identifier
usEntityId	The entity/object id of the entity identifier

§ GetEntityTypeById()

virtual HRESULT GetEntityTypeById (__in **UINT32** uID,
__out **P3D::EntityType** & EntityType
) const private virtual

Provides the **EntityType** for the given object ID if successful.

§ GetEventId()

virtual USHORT GetEventId ( ) private virtual

Creates and returns a unique event id for the session. This value should be used when creating PDU's with the **IPduBuilderV440** interface that require an event ID.

§ GetExerciseId()

virtual BYTE GetExerciseId ( ) const private virtual

Returns the session's current exercise id.

§ GetObjectIdByEntityId()

virtual HRESULT GetObjectIdByEntityId (__in unsigned short usSiteId,
__in unsigned short usApplicationId,
__in unsigned short usEntityId,
__out **UINT32** & uObjectId
) private virtual

Returns the object id for the given entity identifier if successful.

Parameters

usSiteId	The site id of the entity identifier
usApplicationId	The application id of the entity identifier
usEntityId	The entity/object id of the entity identifier
uObjectId	The object id of the request

§ GetSimClockHour()

virtual int GetSimClockHour ( ) const private virtual

Returns the session's current simulation clock hour since 0000 hours January 1, 1970 UTC.

§ GetSimTimestamp()

virtual UINT GetSimTimestamp ( ) const private virtual

Returns the session's current simulation timestamp in DIS timestamp format.

§ GetSiteId()

virtual USHORT GetSiteId ( ) const private virtual

Returns the session's current site id.

§ GetWallClockHour()

virtual int GetWallClockHour ( ) const private virtual

Returns the session's current wall clock hour since 0000 hours January 1, 1970 UTC.

§ GetWallTimestamp()

virtual UINT GetWallTimestamp ( ) const

private virtual

Returns the session's current wall timestamp in DIS timestamp format. This value should be used when filling out the PDU header using the **IPduBuilderV440** interface.

§ IsConnected()

virtual **BOOL** IsConnected ( ) const

private virtual

Returns TRUE if a DIS connection is active, FALSE otherwise.

§ IssuePdu()

virtual HRESULT IssuePdu (**P3D::IPduBuilderV440** * pPduBuilder)

private virtual

Informs core **P3D** to queue the given **IPduBuilderV440** interface data to be broadcast across the network.

Remarks

The **IPduBuilderV440** object can be created with a call to CreatePdu.

This function does not add a ref to the given **IPduBuilderV440** interface.

§ NotifyMunitionDetonated()

```
virtual HRESULT NotifyMunitionDetonated ( __in UINT32                 uAttackerID,
                                          __in UINT32                 uTargetID,
                                          __in UINT32                 uMunitionID,
                                          __in const P3D::EntityType & EntityType,
                                          __in unsigned short         usEventID,
                                          __in const P3D::DXYZ &      xyzLonAltLat,
                                          __in const P3D::DXYZ &      xyzLinearVelocity,
                                          __in unsigned short         usWarheadType,
                                          __in unsigned short         usFuseType,
                                          __in unsigned short         usQuantity,
                                          __in unsigned short         usRate,
                                          __in unsigned char          yDetonationResult
                                        )                                                  private virtual
```

Used to issue a Detonation PDU.

Parameters

uAttackerID	The object ID of the firing entity
uTargetID	The object ID of the target entity if available, 0 otherwise
uMunitionID	The object ID of the munition entity if available, 0 otherwise
EntityType	The **EntityType** of the munition
usEventID	The event ID from an associated Fire PDU if available, 0 otherwise
xyzLonAltLat	Radians/feet
xyzLinearVelocity	World/FPS
usWarheadType	The warhead type
usFuseType	The fuse type
usQuantity	The quantity of munitions represented
usRate	The rate of fire in rounds per minute
yDetonationResult	The result of the detonation

§ NotifyMunitionFired()

```
virtual HRESULT NotifyMunitionFired ( __in UINT32                 uAttackerID,
                                      __in UINT32                 uTargetID,
                                      __in UINT32                 uMunitionID,
                                      __in const P3D::EntityType & EntityType,
                                      __in const P3D::DXYZ &      xyzLonAltLat,
                                      __in const P3D::DXYZ &      xyzLinearVelocity,
                                      __in unsigned short         usWarheadType,
                                      __in unsigned short         usFuseType,
                                      __in unsigned short         usQuantity,
                                      __in unsigned short         usRate,
                                      __in float                  fRange,
                                      __inout unsigned short &    usEventID
                                    )
```

private virtual

Used to issue a Fire PDU.

Parameters

uAttackerID	The object ID of the firing entity
uTargetID	The object ID of the target entity if available, 0 otherwise
uMunitionID	The object ID of the munition entity if available, 0 otherwise
EntityType	The **EntityType** of the munition
xyzLonAltLat	World location in radians and feet
xyzLinearVelocity	World velocity in feet per second
usWarheadType	The warhead type
usFuseType	The fuse type
usQuantity	The quantity of munitions represented
usRate	Rounds per minute
fRange	Meters
usEventID	Set to 0 for new Fire PDU or previously returned value to signify continuous firing

§ RegisterPduCallback()

```
virtual HRESULT RegisterPduCallback ( __in BYTE                          yPduType,
                                      __in __notnull IPduCallbackV440 *  pCallback
                                    )
```

private virtual

Registers a PDU callback.

§ SetDisableReceive()

virtual HRESULT SetDisableReceive (__in **BOOL** bDisableReceive) private virtual

Plugins can toggle whether Prepar3D receives any packets.

Parameters

bDisableReceive True to disable receiving, False to enable receiving.

Remarks

By default receiving is enabled.

§ SetDisableSend()

virtual HRESULT SetDisableSend (__in **BOOL** bDisableSend) private virtual

Plugins can toggle whether Prepar3D sends any packets.

Parameters

bDisableSend True to disable sending, False to enable sending.

Remarks

By default sending is enabled.

§ UnregisterPduCallback()

virtual HRESULT UnregisterPduCallback (__in BYTE yPduType,
__in __notnull **IPduCallbackV440** * pCallback
) private virtual

Unregisters a PDU callback.

§ P3D::IDISServiceV400

class P3D::IDISServiceV400

Professional Plus Only

This service allows developers to provide Distributed Interactive Simulation (DIS) information to the core simulation. Developers should implement this service and provide the requested information following DIS IEEE standards.

Inherits IUnknown.

Private Member Functions

virtual HRESULT	**SerializeEntityAppearance** (__inout UINT &iAppearance) **PURE**
virtual HRESULT	**DeserializeEntityAppearance** (__in UINT iAppearance) **PURE**
virtual HRESULT	**GetArticulatedParameterCount** (__inout UINT &iCount) **PURE**
virtual HRESULT	**SerializeArticulatedParameter** (__in UINT iIndex, __inout **ArticulatedParameter** &ArticulatedParam) **PURE**
virtual HRESULT	**DeserializeArticulatedParameter** (__in UINT iIndex, __in const **ArticulatedParameter** &ArticulatedParam) **PURE**

Member Function Documentation

§ DeserializeArticulatedParameter()

virtual HRESULT DeserializeArticulatedParameter (__in UINT iIndex, __in const **ArticulatedParameter** & ArticulatedParam) private virtual

This function is called on remote objects when the given articulated parameter needs to be updated. This function maybe called by the application when an entity state PDU is received or due to the dead reckoning of the parameter. The function should return S_OK if the articulated parameter was correctly deserialized.

§ DeserializeEntityAppearance()

virtual HRESULT DeserializeEntityAppearance (__in UINT iAppearance) private virtual

This function will be called on remote entities when the appearance needs to be updated. The data should be deserialized in the same manner as described above. The function should return S_OK if the entity appearance was correctly deserialized by the ISimObject.

§ GetArticulatedParameterCount()

virtual HRESULT GetArticulatedParameterCount (__inout UINT & iCount) private virtual

This function is called when requesting the articulated parameter count. The function should return S_OK and provide the number of articulated parameters if the ISimObject is providing articulated parameter support.

§ SerializeArticulatedParameter()

virtual HRESULT SerializeArticulatedParameter (__in UINT iIndex, __inout **ArticulatedParameter** & ArticulatedParam) private virtual

This function is called on a given articulated parameter when the application is requesting an update. The **ArticulatedParameter** union class should be filled out in accordance to DIS standards. This function maybe called by the application when an entity state PDU is required due to heartbeat duration or articulated parameter position or rotation threshold values being exceeded. The function should return S_OK if the articulated parameter was correctly serialized.

§ SerializeEntityAppearance()

virtual HRESULT SerializeEntityAppearance (__inout UINT & iAppearance) private virtual

This function will be called when the application requires the object's entity appearance. The entity appearance is a 32-bit unsigned integer. The application expects the data to be packed according to DIS standards. The application expects the appearance type to match that of the entity type and domain (platform, air, land, munition, expendable, etc.). The function should return S_OK if the entity appearance is being provided by the ISimObject.

§ P3D::ArticulatedParameter.__unnamed__

union P3D::ArticulatedParameter.__unnamed__

Class Members	
ArticulatedPart	m_ArticulatedPart

Variables

GUID **IID_IAIBehaviorManagerV01**
GUID **SID_AIBehaviorManager**
GUID **SID_AIBehavior**
GUID **IID_IAIBehaviorWingmanFormationV01**
GUID **SID_AIBehaviorWingmanFormation**
GUID **IID_IAIBehaviorAttackerV400**
GUID **SID_AIBehaviorAttacker**

GUID	**IID_IAIBehaviorPursueV01**
GUID	**SID_AIBehaviorPursue**
GUID	**IID_IAIBehaviorCombatAirPatrolV01**
GUID	**SID_AIBehaviorCombatAirPatrol**
GUID	**IID_IAIBehaviorCloseAirSupportV01**
GUID	**SID_AIBehaviorCloseAirSupport**
GUID	**IID_IAIBehaviorSearchTrackV01**
GUID	**SID_AIBehaviorSearchTrack**
GUID	**IID_ISimObjectAIV02**
GUID	**SID_SimObjectAI**
GUID	**SID_AIService**
GUID	**SID_AircraftAIService**
GUID	**IID_IAirplaneAIServiceV02**
GUID	**SID_AirplaneAIService**
GUID	**IID_IHelicopterAIServiceV420**
GUID	**SID_HelicopterAIService**
GUID	**IID_IGroundVehicleAIServiceV01**
GUID	**SID_GroundVehicleAIService**
GUID	**IID_IWeaponsSystemV440**
GUID	**SID_WeaponsSystem**
GUID	**IID_IWeaponServiceV420**
GUID	**SID_WeaponService**
GUID	**IID_ICountermeasureSystemV01**
GUID	**SID_CountermeasureSystem**
GUID	**IID_ICountermeasureServiceV02**
GUID	**SID_CountermeasureService**
GUID	**IID_IGunSystemV440**
GUID	**SID_GunSystem**
GUID	**IID_IGunV400**
GUID	**SID_Gun**
GUID	**IID_IFireControlSystemV01**
GUID	**SID_FireControlSystem**
GUID	**IID_IGuidanceSystemV01**
GUID	**SID_GuidanceSystem**
GUID	**IID_IPylonServiceV01**
GUID	**SID_PylonService**
GUID	**IID_IPduBuilderV440**
GUID	**IID_IPduReaderV440**
GUID	**IID_IPduCallbackV440**
GUID	**IID_IDISManagerV450**
GUID	**SID_DISManager**
GUID	**IID_IDISServiceV400**
GUID	**SID_DISService**

Variable Documentation

§ IID_IAIBehaviorAttackerV400

GUID IID_IAIBehaviorAttackerV400

§ IID_IAIBehaviorCloseAirSupportV01

GUID IID_IAIBehaviorCloseAirSupportV01

§ IID_IAIBehaviorCombatAirPatrolV01

GUID IID_IAIBehaviorCombatAirPatrolV01

§ IID_IAIBehaviorManagerV01

GUID IID_IAIBehaviorManagerV01

§ IID_IAIBehaviorPursueV01

GUID IID_IAIBehaviorPursueV01

§ IID_IAIBehaviorSearchTrackV01

GUID IID_IAIBehaviorSearchTrackV01

§ IID_IAIBehaviorWingmanFormationV01

GUID IID_IAIBehaviorWingmanFormationV01

§ IID_IAirplaneAIServiceV02

GUID IID_IAirplaneAIServiceV02

§ IID_ICountermeasureServiceV02

GUID IID_ICountermeasureServiceV02

§ IID_ICountermeasureSystemV01

GUID IID_ICountermeasureSystemV01

§ IID_IDISManagerV450

GUID IID_IDISManagerV450

§ IID_IDISServiceV400

GUID IID_IDISServiceV400

§ IID_IFireControlSystemV01

GUID IID_IFireControlSystemV01

§ IID_IGroundVehicleAIServiceV01

GUID IID_IGroundVehicleAIServiceV01

§ IID_IGuidanceSystemV01

GUID IID_IGuidanceSystemV01

§ IID_IGunSystemV440

GUID IID_IGunSystemV440

§ IID_IGunV400

GUID IID_IGunV400

§ IID_IHelicopterAIServiceV420

GUID IID_IHelicopterAIServiceV420

§ IID_IPduBuilderV440

GUID IID_IPduBuilderV440

§ IID_IPduCallbackV440

GUID IID_IPduCallbackV440

§ IID_IPduReaderV440

GUID IID_IPduReaderV440

§ IID_IPylonServiceV01

GUID IID_IPylonServiceV01

§ IID_ISimObjectAIV02

GUID IID_ISimObjectAIV02

§ IID_IWeaponServiceV420

GUID IID_IWeaponServiceV420

§ IID_IWeaponsSystemV440

GUID IID_IWeaponsSystemV440

§ SID_AIBehavior

GUID SID_AIBehavior

§ SID_AIBehaviorAttacker

GUID SID_AIBehaviorAttacker

§ SID_AIBehaviorCloseAirSupport

GUID SID_AIBehaviorCloseAirSupport

§ SID_AIBehaviorCombatAirPatrol

GUID SID_AIBehaviorCombatAirPatrol

§ SID_AIBehaviorManager

GUID SID_AIBehaviorManager

§ SID_AIBehaviorPursue

GUID SID_AIBehaviorPursue

§ SID_AIBehaviorSearchTrack

GUID SID_AIBehaviorSearchTrack

§ SID_AIBehaviorWingmanFormation

GUID SID_AIBehaviorWingmanFormation

§ SID_AircraftAIService

GUID SID_AircraftAIService

§ SID_AirplaneAIService

GUID SID_AirplaneAIService

§ SID_AIService

GUID SID_AIService

§ SID_CountermeasureService

GUID SID_CountermeasureService

§ SID_CountermeasureSystem

GUID SID_CountermeasureSystem

§ SID_DISManager

GUID SID_DISManager

§ SID_DISService

GUID SID_DISService

§ SID_FireControlSystem

GUID SID_FireControlSystem

§ SID_GroundVehicleAIService

GUID SID_GroundVehicleAIService

§ SID_GuidanceSystem

GUID SID_GuidanceSystem

§ SID_Gun

GUID SID_Gun

§ SID_GunSystem

GUID SID_GunSystem

§ SID_HelicopterAIService

GUID SID_HelicopterAIService

§ SID_PylonService

GUID SID_PylonService

§ SID_SimObjectAI

GUID SID_SimObjectAI

§ SID_WeaponService

GUID SID_WeaponService

§ SID_WeaponsSystem

GUID SID_WeaponsSystem

- top -