Plan View Display 1.2 Documentation for Plug-in Classes

This file documents the classes used to create plug-ins. It is based on the header files and includes many of the comments from them. It provides an alternative to looking through the individual header files when you need information about constructors or other aspects of a class's functioning. We hope you find this format helpful.

Classes are listed alphabetically and in a class hierarchy. The hypertext links will help you quickly jump to related information.

Notes

- If you find a difference between the information in this file and the original header file shipped with your version of the Plan View Display, rely on the header file.
- The top line in each class description lists the header file in which it is defined. However, the capitalization of the header file is incorrect. The actual header files use mixed capitalization, not all lowercase.
- Some links point to header files. If you click one of these links, you will get a message indicating that the file could not be found. The file name will be included in the message, so if you want to, you can look at the file in the Plan View Display User's Guide or the delivered files.

MÄK Technologies, Inc.

185 Alewife Brook Parkway

Cambridge, MA 02138

Phone: 617.876.8085

Fax: 617.876.9208

www.mak.com

support@mak.com

Document ID: PVD-1.2-5-990504

Plan View Display 1.2 Documentation for Plug-in Classes

Hierarchy of classes

Table of Contents

Classes

- PvPluginAppInterface *class PvPluginAppInterface: Instances of PvPluginAppInterface are the interface between the plugin and the application*
- PvPluginCanvasInterface *class PvPluginCanvasInterface: Instances of PvPluginCanvasInterface are the interface between the plugin and a specific plan view window*
- PvPluginCircleObjectInterface *class PvPluginCircleObjectInterface: Instances of PvPluginCircleObjectInterface are the interface between the plugin and a circle sim object*
- PvPluginEntityObjectInterface *class PvPluginEntityObjectInterface: Instances of PvPluginEntityObjectInterface are the interface between the plugin and a single entity sim object*
- PvPluginInteractionObjectInterface *class PvPluginInteractionObjectInterface: Instances of PvPluginInteractionObjectInterface are the interface between the plugin and a base interaction*
- PvPluginLineObjectInterface *class PvPluginLineObjectInterface: Instances of PvPluginLineObjectInterface are the interface between the plugin and a line sim object*
- PvPluginObjectInterface *class PvPluginObjectInterface: Instances of PvPluginObjectInterface are the interface between the plugin and an object (sim object) in the PVD*
- PvPluginObjectListInterface *class PvPluginObjectListInterface: Instances of PvPluginObjectListInterface are the interface between the plugin and the sim object list*
- PvPluginPointObjectInterface *class PvPluginPointObjectInterface: Instances of PvPluginPointObjectInterface are the interface between the plugin and a point sim object*
- PvPluginPolygonObjectInterface *class PvPluginPolygonObjectInterface: Instances of PvPluginPolygonObjectInterface are the interface between the plugin and a polygon sim object*
- PvPluginTextObjectInterface

Functions

- (*PvPluginMenuCbFunc)
- (*PvPluginObjectCbFunc)
- (*PvPluginObjectListCbFunc)
- (*PvPluginObjectMenuCbFunc)
- (*PvPluginPvCbFunc)

- (*PvPluginTickCbFunc)
- (*PvPluginTrackedObjCbFunc)
- (*PvPluginUserEventCbFunc)
- DtDECLARE_DEFINE_RIGHT_END_CLASS
- DtDECLARE_DEFINE_RIGHT_END_CLASS
- DtDECLARE_DEFINE_RIGHT_END_CLASS
- DtDECLARE_DEFINE_RIGHT_END_CLASS
- DtDECLARE_DEFINE_RIGHT_END_CLASS_WITH_MOD
- DtRIGHT_END_NOTIFIER
- DtRIGHT_END_NOTIFIER
- DtRIGHT_END_NOTIFIER
- DtRIGHT_END_NOTIFIER
- DtRIGHT_END_NOTIFIER_WITH_MOD
- PvPluginAppInterface *copy constructor*
- PvPluginCanvasInterface *constructor*
- PvPluginCircleObjectInterface *constructor*
- PvPluginEntityObjectInterface *constructor*
- PvPluginInteractionObjectInterface *copy constructor*
- PvPluginLineObjectInterface *constructor*
- PvPluginObjectInterface *constructor*
- PvPluginObjectListInterface *constructor*
- PvPluginPointObjectInterface *constructor*
- PvPluginPolygonObjectInterface *copy constructor*
- PvPluginTextObjectInterface *copy constructor*
- accessInternals *Get at the internals - requires additional header files and libraries*
- accessInternals *Get at the internals - requires additional header files and libraries*
- accessInternals *Get at the internals - requires additional header files and libraries*
- accessInternals *Get at the internals - requires additional header files and libraries*
- addCallbackForPlanViewWindow *Add/Remove a callback on plan view window creation and destruction*
- addCallbackForTrackedObjects *Add/Remove a callback that is triggered when the tracked object(s) change*
- addCallbackForUserEvents *Add/Remove a callback that is triggered for a user event*
- addCallbackOnObjectList *Add/Remove a callback on the sim object list*
- addCallbackOnObject *Add/Remove callback on the object*
- addCircleObject
- addEntityObject *Add objects of various types*
- addInteractionObject
- addItemToMenu *Add an item to a top-level menu on the plan view menubar*
- addLineObject
- addListObject *Adds/Removes an object interface from the list only*
- addMenuToMenu *Add a menu to a top-level menu on the plan view menubar*
- addPointObject

- addPolygonObject
- addRightClickMenuItem *Add a right click menu item*
- addTextObject
- addTickCallback *Add/Remove a function from the tick callback list*
- addVertexAfterVertex *Adds a vertex after another vertex*
- addVertexAfterVertex *Adds a vertex after another vertex*
- addVertexToEnd
- addVertexToEnd
- addVertexToStart *Adds a vertex to the start/end*
- addVertexToStart *Adds a vertex to the start/end*
- annotation
- appInterface *Get the app interface*
- appInterface *Get the plugin app interface*
- canvasInterfaceList *Get the list of canvas interfaces*
- centerGeocLocation
- circumferenceGeocLocation
- databaseName *Get the filename of the database*
- deleteMarkedObjectCallbacks *The remove function above merely mark the callbacks for deletion*
- deleteMarkedObjectListCallbacks *The remove function above merely mark the callbacks for deletion*
- deleteMarkedPlanViewCallbacks
- deleteMarkedTickCallbacks *The remove functions above merely mark the callbacks for deletion*
- deleteMarkedTrackedObjectCallbacks
- deleteMarkedUserEventCallbacks *The remove functions above merely mark the callbacks for deletion*
- deleteVertex *Removes a vertex*
- deleteVertex *Removes a vertex*
- drawingStyle
- drawingWidth
- esr
- exerciseConn *Returns the exercise connection*
- fillStyle
- fillStyle
- fontFamily
- fontSize
- fontStyle
- fontWeight
- getCanvasCoordFromDatabase
- getCanvasCoordFromGeoc *Convert geocentric world/database coordinates from canvas coordinates*
- getCanvasExtentsInDatabaseCoord
- getColor
- getDatabaseCoordFromCanvas

- getDatabaseCoordFromGeoc *Convert geocentric world/canvas coordinates to database coordinates*
- getFillColor
- getFillColor
- getGeocCoordFromCanvas
- getGeocCoordFromDatabase *Convert database/canvas coordinates to geocentric world coordinates*
- getSelectedObjectInterfaces *Get a list of interfaces to the currently selected objects*
- getSelectedPoint *Gets the currently selected point in database coordinates*
- getTerrainInformation *Get the height of terrain, soil type (if any), and ground normal at a particular*
- globalId *Get the entity's object's global object designator*
- hidden
- interaction *Get the interaction*
- interfaceFactory *Get the interface factory*
- listOfVertices *Get the list of vertices*
- listOfVertices *Get the list of vertices*
- lookup *Lookup an object interface*
- menuCb *Handle the menubar menu callbacks*
- numberOfVertices *Gets the number of vertices*
- numberOfVertices *Gets the number of vertices*
- objectGeocLocation
- objectId *Get the object id*
- objectInterfaceList *Get the list of object interfaces*
- objectListInterface *Get the object list interface*
- objectListInterface *Get the object list interface*
- operator= *assignment operator*
- operator= *assignment operator*
- operator= *assignment operator*
- operator= *assignment operator*
- operator= *assignment operator*
- operator= *assignment operator*
- operator= *assignment operator*
- operator= *assignment operator*
- operator= *assignment operator*
- operator= *assignment operator*
- operator= *assignment operator*
- processMessage *Processes a message from the main frame*
- processMessage *Processes a message from the main frame*
- processMessage *Processes a message from the main frame*
- processMessage *Processes a message from the main frame*
- removeAllCallbacksForPlanViewWindow
- removeAllCallbacksForTrackedObjects
- removeAllCallbacksForUserEvents

- removeAllCallbacksOnObjectList
- removeAllCallbacksOnObject
- removeAllTickCallbacks
- removeCallbackForPlanViewWindow
- removeCallbackForTrackedObjects
- removeCallbackForUserEvents
- removeCallbackOnObjectList
- removeCallbackOnObject
- removeListObject
- removeObject
- removeTickCallback
- rightClickCb *Handle the right click menu callbacks*
- selection
- setAnnotation *Set/Get the object's annotation*
- setCanvasExtentsInDatabaseCoord *Set/Get the rectangle that the canvas is viewing*
- setCenterGeocLocation *Set/Get the circle's center location in geocentric world coordinates*
- setCircumferenceGeocLocation *Set/Get the circle's circumference location in geocentric world coordinates*
- setColor *Set/Get the object's color on a per canvas basis*
- setDrawingStyle *Set/Get the object's drawing style on a per canvas basis*
- setDrawingWidth *Set/Get the object's pen width on a per canvas basis*
- setEsr *Set/Get the entity state repository*
- setFillColor *Set/Get the circle's fill color on a per canvas basis*
- setFillColor *Set/Get the polygon's fill color on a per canvas basis*
- setFillStyle *Set/Get the circle's fill style on a per canvas basis*
- setFillStyle *Set/Get the polygon's fill style on a per canvas basis*
- setFontFamily *Set/Get the object's font information*
- setFontSize
- setFontStyle
- setFontWeight
- setHidden *Set/Get the object's hidden status on a per canvas basis*
- setObjectGeocLocation *Set/Get the point location*
- setSelection *Set/Get the object's selection status on a per canvas basis*
- setString *Set/Get the object's text string*
- setTimeoutValue *Set/Get the object's timeout value on a per canvas basis*
- setVertex
- setVertex
- string
- tick *Tick function, executed every cycle by the pvd app*
- timeoutValue
- update *Force an update*

- version *Returns the version info, so that the plugin can match with the app*
- vertex *Set/Get a particular vertex*
- vertex *Set/Get a particular vertex*
- ~PvPluginAppInterface *destructor*
- ~PvPluginCanvasInterface *destructor*
- ~PvPluginCircleObjectInterface *destructor*
- ~PvPluginEntityObjectInterface *destructor*
- ~PvPluginInteractionObjectInterface *destructor*
- ~PvPluginLineObjectInterface *destructor*
- ~PvPluginObjectInterface *destructor*
- ~PvPluginObjectListInterface *destructor*
- ~PvPluginPointObjectInterface *destructor*
- ~PvPluginPolygonObjectInterface *destructor*
- ~PvPluginTextObjectInterface *destructor*

Variables

- PvPluginDIS3
- PvPluginDIS5
- PvPluginDrawingStyleDash
- PvPluginDrawingStyleDot
- PvPluginDrawingStyleSolid
- PvPluginDrawingStyleTransparent
- PvPluginFillStyleCrossHatch
- PvPluginFillStyleCrossHatch
- PvPluginFillStyleDiagonalBackwardHatch
- PvPluginFillStyleDiagonalBackwardHatch
- PvPluginFillStyleDiagonalCrossHatch
- PvPluginFillStyleDiagonalCrossHatch
- PvPluginFillStyleDiagonalForwardHatch
- PvPluginFillStyleDiagonalForwardHatch
- PvPluginFillStyleHorizontalHatch
- PvPluginFillStyleHorizontalHatch
- PvPluginFillStyleSolid
- PvPluginFillStyleSolid
- PvPluginFillStyleTransparent
- PvPluginFillStyleTransparent
- PvPluginFillStyleVerticalHatch
- PvPluginFillStyleVerticalHatch
- PvPluginFontFamilyDefault
- PvPluginFontFamilyModern

- PvPluginFontFamilyRoman
- PvPluginFontStyleItalic
- PvPluginFontStyleNormal
- PvPluginFontWeightBold
- PvPluginFontWeightNormal
- PvPluginGenericAddedMessage
- PvPluginGenericChangedMessage
- PvPluginGenericDeletedMessage
- PvPluginGenericOtherMessage
- PvPluginGroundSelection
- PvPluginHLA
- PvPluginObjectSelectionChanged
- PvSurfaceOther
- PvSurfacePaved
- PvSurfacePavedRoad
- PvSurfaceUnpavedNoGo
- PvSurfaceUnpaved
- PvSurfaceUnpavedSlowGo
- PvSurfaceUnspecified
- PvSurfaceWetLake
- PvSurfaceWetMuck
- PvSurfaceWetOcean
- PvSurfaceWet
- PvSurfaceWetRiver
- PvSurfaceWetSwamp
- myAppInterface
- myAppInterface
- myApp
- myCanvasInterfaceList
- myCircleSimObject
- myEntitySimObject
- myIdForMenuCreation
- myInteractionSimObject
- myInterfaceFactory
- myLineSimObject
- myObjectCbFunctions
- myObjectInterfaceList
- myObjectListCallbacks
- myObjectListInterface
- myObjectListInterface
- myObject

- myPlanView
- myPointSimObject
- myPolygonSimObject
- myPvCbFunctions
- mySimObjectList
- myTextSimObject
- myTickCbFunctions
- myTrackingCbFunctions
- myUserEventCbFunctions

Macros

- EXPORT
- PvPluginAppInterface_H_
- PvPluginCanvasInterface_H_
- PvPluginCircleObjectInterface_H_
- PvPluginEntityObjectInterface_H_
- PvPluginInteractionObjectInterface_H_
- PvPluginLineObjectInterface_H_
- PvPluginObjectInterface_H_
- PvPluginObjectListInterface_H_
- PvPluginPointObjectInterface_H_
- PvPluginPolygonObjectInterface_H_
- PvPluginTextObjectInterface_H_

Enums, Unions, Structs

- PvPluginDrawingStyle
- PvPluginFillStyle
- PvPluginFillStyle
- PvPluginFontFamilyType
- PvPluginFontStyleType
- PvPluginFontWeightType
- PvPluginGenericMessageType
- PvPluginSurfaceType
- PvPluginUserEventType
- PvPluginVersionInfo

Hierarchy of classes

Hierarchy of Classes

- PvPluginAppInterface
- PvPluginCanvasInterface
- PvPluginObjectInterface
    - PvPluginPolygonObjectInterface
    - PvPluginPointObjectInterface
        - PvPluginTextObjectInterface
    - PvPluginLineObjectInterface
    - PvPluginInteractionObjectInterface
    - PvPluginEntityObjectInterface
    - PvPluginCircleObjectInterface
- PvPluginObjectListInterface

In file plugappinter.h:

class PvPluginAppInterface

class PvPluginAppInterface: Instances of PvPluginAppInterface are the interface between the plugin and the application

---

Public Classes

- enum **PvPluginVersionInfo**
  - **PvPluginDIS3**
  - **PvPluginDIS5**
  - **PvPluginHLA**

Public Methods

	PvPluginAppInterface (PvPvdApp *app) *default constructor*
virtual	**~PvPluginAppInterface** () *destructor*
virtual DtExerciseConn*	**exerciseConn** () *Returns the exercise connection*
virtual void	**addCallbackForPlanViewWindow** (PvPluginPvCbFunc cb, void* usr) *Add/Remove a callback on plan view window creation and destruction*
virtual DtBoolean	**removeCallbackForPlanViewWindow** (PvPluginPvCbFunc cb)
virtual void	**removeAllCallbacksForPlanViewWindow** ()
virtual void	**addTickCallback** (PvPluginTickCbFunc cb, void* usr) *Add/Remove a function from the tick callback list*
virtual DtBoolean	**removeTickCallback** (PvPluginTickCbFunc cb)
virtual void	**removeAllTickCallbacks** ()
virtual PvPluginObjectListInterface*	**objectListInterface** () *Get the object list interface*
virtual DtList*	**canvasInterfaceList** () *Get the list of canvas interfaces*
virtual PvPvdApp*	**accessInternals** () *Get at the internals - requires additional header files and libraries*
virtual void	**tick** () *Tick function, executed every cycle by the pvd app*
virtual PvPluginVersionInfo	**version** () *Returns the version info, so that the plugin can match with the app*
virtual DtBoolean	**processMessage** (const PvCanvasWindowConduitMessage* message) *Processes a message from the main frame*

Protected Fields

- PvPvdApp* **myApp**
- PvPluginObjectListInterface* **myObjectListInterface**
- DtList* **myCanvasInterfaceList**
- DtList* **myTickCbFunctions**
- DtList* **myPvCbFunctions**

Protected Methods

- **PvPluginAppInterface** (const PvPluginAppInterface& orig)
  copy constructor
- PvPluginAppInterface& **operator=** (const PvPluginAppInterface& orig)
  assignment operator
- virtual void **deleteMarkedTickCallbacks** ()
  The remove functions above merely mark the callbacks for deletion
- virtual void **deleteMarkedPlanViewCallbacks** ()
- **DtRIGHT_END_NOTIFIER** (PvPluginAppInterface)

Documentation

class PvPluginAppInterface: Instances of PvPluginAppInterface are the interface between the plugin and the application

- **`enum PvPluginVersionInfo`**
- **`PvPluginDIS3`**
- **`PvPluginDIS5`**
- **`PvPluginHLA`**
- **`PvPluginAppInterface(PvPvdApp *app)`**
  default constructor
- **`virtual ~PvPluginAppInterface()`**
  destructor
- **`virtual DtExerciseConn* exerciseConn()`**
  Returns the exercise connection
- **`virtual void addCallbackForPlanViewWindow(PvPluginPvCbFunc cb, void* usr)`**
  Add/Remove a callback on plan view window creation and destruction
- **`virtual DtBoolean removeCallbackForPlanViewWindow( PvPluginPvCbFunc cb)`**
- **`virtual void removeAllCallbacksForPlanViewWindow()`**
- **`virtual void addTickCallback(PvPluginTickCbFunc cb, void* usr)`**
  Add/Remove a function from the tick callback list

- **virtual DtBoolean removeTickCallback(PvPluginTickCbFunc cb)**
- **virtual void removeAllTickCallbacks()**
- **virtual PvPluginObjectListInterface* objectListInterface()**

  Get the object list interface
- **virtual DtList* canvasInterfaceList()**

  Get the list of canvas interfaces
- **virtual PvPvdApp* accessInternals()**

  Get at the internals - requires additional header files and libraries
- **virtual void tick()**

  Tick function, executed every cycle by the pvd app. Not intended as part of the interface for the plugin creator.
- **virtual PvPluginVersionInfo version()**

  Returns the version info, so that the plugin can match with the app
- **virtual DtBoolean processMessage(const PvCanvasWindowConduitMessage* message)**

  Processes a message from the main frame. Not part of the interface for the plugin creator.
- **PvPluginAppInterface(const PvPluginAppInterface& orig)**

  copy constructor
- **PvPluginAppInterface& operator=(const PvPluginAppInterface& orig)**

  assignment operator
- **virtual void deleteMarkedTickCallbacks()**

  The remove functions above merely mark the callbacks for deletion. These functions actually delete them. This is done to prevent someone from corrupting the list while the list is executing its callbacks.
- **virtual void deleteMarkedPlanViewCallbacks()**
- **PvPvdApp* myApp**
- **PvPluginObjectListInterface* myObjectListInterface**
- **DtList* myCanvasInterfaceList**
- **DtList* myTickCbFunctions**
- **DtList* myPvCbFunctions**
- **DtRIGHT_END_NOTIFIER(PvPluginAppInterface)**

This class has no child classes.

Alphabetic index *Hierarchy of classes*

In file plugobjlinter.h:

class PvPluginObjectListInterface

class PvPluginObjectListInterface: Instances of PvPluginObjectListInterface are the interface between the plugin and the sim object list

---

Public Methods

	PvPluginObjectListInterface (PvSimObjectList* objList, PvPluginAppInterface* appInterface) *constructor*
virtual	**~PvPluginObjectListInterface** () *destructor*
virtual PvObjectId	**addEntityObject** (DtEntityStateRepository* esr, DtGlobalObjectDesignator globalId) *Add objects of various types*
virtual PvObjectId	**addInteractionObject** (DtInteraction* interaction)
virtual PvObjectId	**addPointObject** (DtConstVector geocCoord)
virtual PvObjectId	**addLineObject** (DtList* listOfGeocCoords)
virtual PvObjectId	**addPolygonObject** (DtList* listOfGeocCoords)
virtual PvObjectId	**addCircleObject** (DtConstVector centerGeocCoord, DtConstVector circumferenceGeocCoord)
virtual PvObjectId	**addTextObject** (DtConstVector geocCoord, const DtString& text)
virtual DtBoolean	**removeObject** (PvPluginObjectInterface* objInterface) *Remove an object*
virtual DtBoolean	**removeObject** (PvObjectId id)
virtual const DtList*	**objectInterfaceList** () *Get the list of object interfaces*
virtual PvPluginObjectInterface*	**lookup** (PvObjectId id) *Lookup an object interface*
virtual void	**addCallbackOnObjectList** (PvPluginObjectListCbFunc cb, void* usr) *Add/Remove a callback on the sim object list*
virtual DtBoolean	**removeCallbackOnObjectList** (PvPluginObjectListCbFunc cb)
virtual void	**removeAllCallbacksOnObjectList** ()
virtual PvPluginAppInterface*	**appInterface** () *Get the plugin app interface*
virtual PvPluginObjectInterfaceFactory*	**interfaceFactory** () *Get the interface factory*
virtual PvSimObjectList*	**accessInternals** () *Get at the internals - requires additional header files and libraries*
virtual DtBoolean	**processMessage** (const PvBaseObjectConduitMessage* message) *Processes a message from the main frame*

Protected Fields

PvSimObjectList*	**mySimObjectList**
PvPluginAppInterface*	**myAppInterface**
DtHashlist*	**myObjectInterfaceList**
DtList*	**myObjectListCallbacks**
PvPluginObjectInterfaceFactory*	**myInterfaceFactory**

Protected Methods

	PvPluginObjectListInterface (const PvPluginObjectListInterface& orig) *copy constructor*
PvPluginObjectListInterface&	**operator=** (const PvPluginObjectListInterface& orig) *assignment operator*
virtual PvPluginObjectInterface*	**addListObject** (PvObjectId id) *Adds/Removes an object interface from the list only*
virtual void	**removeListObject** (PvObjectId id)
virtual void	**deleteMarkedObjectListCallbacks** () *The remove function above merely mark the callbacks for deletion*
	DtRIGHT_END_NOTIFIER (PvPluginObjectListInterface)

Documentation

class PvPluginObjectListInterface: Instances of PvPluginObjectListInterface are the interface between the plugin and the sim object list

`PvPluginObjectListInterface(PvSimObjectList* objList, PvPluginAppInterface* appInterface)`

constructor

`virtual ~PvPluginObjectListInterface()`

destructor

`virtual PvObjectId addEntityObject(DtEntityStateRepository* esr, DtGlobalObjectDesignator globalId)`

Add objects of various types

`virtual PvObjectId addInteractionObject(DtInteraction* interaction)`

`virtual PvObjectId addPointObject(DtConstVector geocCoord)`

`virtual PvObjectId addLineObject(DtList* listOfGeocCoords)`

`virtual PvObjectId addPolygonObject(DtList* listOfGeocCoords)`

`virtual PvObjectId addCircleObject(DtConstVector centerGeocCoord, DtConstVector circumferenceGeocCoord)`

`virtual PvObjectId addTextObject(DtConstVector geocCoord, const DtString& text)`

`virtual DtBoolean removeObject(PvPluginObjectInterface* objInterface)`

Remove an object

- **virtual DtBoolean removeObject(PvObjectId id)**
- **virtual const DtList* objectInterfaceList()**

  Get the list of object interfaces
- **virtual PvPluginObjectInterface* lookup(PvObjectId id)**

  Lookup an object interface
- **virtual void addCallbackOnObjectList(PvPluginObjectListCbFunc cb, void* usr)**

  Add/Remove a callback on the sim object list
- **virtual DtBoolean removeCallbackOnObjectList(PvPluginObjectListCbFunc cb)**
- **virtual void removeAllCallbacksOnObjectList()**
- **virtual PvPluginAppInterface* appInterface()**

  Get the plugin app interface
- **virtual PvPluginObjectInterfaceFactory* interfaceFactory()**

  Get the interface factory
- **virtual PvSimObjectList* accessInternals()**

  Get at the internals - requires additional header files and libraries
- **virtual DtBoolean processMessage(const PvBaseObjectConduitMessage* message)**

  Processes a message from the main frame. Not part of the interface for the plugin creator.
- **PvPluginObjectListInterface(const PvPluginObjectListInterface& orig)**

  copy constructor
- **PvPluginObjectListInterface& operator=(const PvPluginObjectListInterface& orig)**

  assignment operator
- **virtual PvPluginObjectInterface* addListObject(PvObjectId id)**

  Adds/Removes an object interface from the list only
- **virtual void removeListObject(PvObjectId id)**
- **virtual void deleteMarkedObjectListCallbacks()**

  The remove function above merely mark the callbacks for deletion. This function actually deletes them. This is done to prevent someone from corrupting the list while the list is executing its callbacks.
- **PvSimObjectList* mySimObjectList**
- **PvPluginAppInterface* myAppInterface**
- **DtHashlist* myObjectInterfaceList**
- **DtList* myObjectListCallbacks**
- **DtRIGHT_END_NOTIFIER(PvPluginObjectListInterface)**
- **PvPluginObjectInterfaceFactory* myInterfaceFactory**

This class has no child classes.

In file plugobjinter.h:

class PvPluginObjectInterface: public DtListedObj

class PvPluginObjectInterface: Instances of PvPluginObjectInterface are the interface between the plugin and an object (sim object) in the PVD

Inheritance:

PvPluginObjectInterface

Public Classes

- enum **PvPluginDrawingStyle**
  - **PvPluginDrawingStyleSolid**
  - **PvPluginDrawingStyleDot**
  - **PvPluginDrawingStyleDash**
  - **PvPluginDrawingStyleTransparent**

Public Methods

	PvPluginObjectInterface (PvSimObject* object, PvPluginObjectListInterface* listInterface) *constructor*
virtual	**~PvPluginObjectInterface** () *destructor*
virtual void	**addCallbackOnObject** (PvPluginObjectCbFunc cb, void* usr) *Add/Remove callback on the object*
virtual DtBoolean	**removeCallbackOnObject** (PvPluginObjectCbFunc cb)
virtual void	**removeAllCallbacksOnObject** ()
virtual PvObjectId	**objectId** () *Get the object id*
virtual PvPluginObjectListInterface*	**objectListInterface** () *Get the object list interface*
virtual void	**update** () *Force an update*
virtual void	**setAnnotation** (const DtString& str) *Set/Get the object's annotation*
virtual const DtString&	**annotation** () const
virtual void	**setSelection** (DtBoolean value, PvPluginCanvasInterface* canvas = NULL) *Set/Get the object's selection status on a per canvas basis*
virtual DtBoolean	**selection** (PvPluginCanvasInterface* canvas)

virtual void	**setHidden** (DtBoolean value, PvPluginCanvasInterface* canvas = NULL) *Set/Get the object's hidden status on a per canvas basis*
virtual DtBoolean	**hidden** (PvPluginCanvasInterface* canvas)
virtual void	**setTimeoutValue** (DtTime time, PvPluginCanvasInterface* canvas = NULL) *Set/Get the object's timeout value on a per canvas basis*
virtual DtTime	**timeoutValue** (PvPluginCanvasInterface* canvas = NULL)
virtual void	**setColor** (unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL) *Set/Get the object's color on a per canvas basis*
virtual void	**getColor** (unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)
virtual void	**setDrawingStyle** (PvPluginDrawingStyle style, PvPluginCanvasInterface* canvas = NULL) *Set/Get the object's drawing style on a per canvas basis*
virtual PvPluginDrawingStyle	**drawingStyle** (PvPluginCanvasInterface* canvas)
virtual void	**setDrawingWidth** (int width, PvPluginCanvasInterface* canvas = NULL) *Set/Get the object's pen width on a per canvas basis*
virtual int	**drawingWidth** (PvPluginCanvasInterface* canvas)
virtual PvSimObject*	**accessInternals** () *Get at the internals - requires additional header files and libraries*
virtual DtBoolean	**processMessage** (const PvBaseObjectConduitMessage* message) *Processes a message from the main frame*

Protected Fields

PvSimObject*	**myObject**
PvPluginObjectListInterface*	**myObjectListInterface**
DtList*	**myObjectCbFunctions**

Protected Methods

	PvPluginObjectInterface (const PvPluginObjectInterface& orig) *copy constructor*
PvPluginObjectInterface&	**operator=** (const PvPluginObjectInterface& orig) *assignment operator*
virtual void	**deleteMarkedObjectCallbacks** () *The remove function above merely mark the callbacks for deletion*
	DtRIGHT_END_NOTIFIER (PvPluginObjectInterface)

Documentation

class PvPluginObjectInterface: Instances of PvPluginObjectInterface are the interface between the plugin and an object (sim object) in the PVD

- **`enum PvPluginDrawingStyle`**
  - **`PvPluginDrawingStyleSolid`**
  - **`PvPluginDrawingStyleDot`**

- **PvPluginDrawingStyleDash**
- **PvPluginDrawingStyleTransparent**
- **PvPluginObjectInterface(PvSimObject* object, PvPluginObjectListInterface* listInterface)**

  constructor
- **virtual ~PvPluginObjectInterface()**

  destructor
- **virtual void addCallbackOnObject(PvPluginObjectCbFunc cb, void* usr)**

  Add/Remove callback on the object. It is especially easy to create a loop which can corrupt the callback list and crash the program. For example, do not, in your callback, delete a callback and do anything which causes the entity to update.
- **virtual DtBoolean removeCallbackOnObject(PvPluginObjectCbFunc cb)**
- **virtual void removeAllCallbacksOnObject()**
- **virtual PvObjectId objectId()**

  Get the object id
- **virtual PvPluginObjectListInterface* objectListInterface()**

  Get the object list interface
- **virtual void update()**

  Force an update
- **virtual void setAnnotation(const DtString& str)**

  Set/Get the object's annotation
- **virtual const DtString& annotation() const**
- **virtual void setSelection(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)**

  Set/Get the object's selection status on a per canvas basis. A NULL value for the canvas sets it application wide.
- **virtual DtBoolean selection(PvPluginCanvasInterface* canvas)**
- **virtual void setHidden(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)**

  Set/Get the object's hidden status on a per canvas basis. A NULL value for the canvas sets it application wide.
- **virtual DtBoolean hidden(PvPluginCanvasInterface* canvas)**
- **virtual void setTimeoutValue(DtTime time, PvPluginCanvasInterface* canvas = NULL)**

  Set/Get the object's timeout value on a per canvas basis. A zero value for the time will disable timeout processing. A NULL value for the canvas sets or gets it application wide.
- **virtual DtTime timeoutValue(PvPluginCanvasInterface* canvas = NULL)**
- **virtual void setColor(unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL)**

  Set/Get the object's color on a per canvas basis. If the object does not have settable colors, these do nothing. A NULL value for the canvas sets it application wide.
- **virtual void getColor(unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)**
- **virtual void setDrawingStyle(PvPluginDrawingStyle style, PvPluginCanvasInterface* canvas = NULL)**

Set/Get the object's drawing style on a per canvas basis. If the object does not have settable drawing style, these do nothing. A NULL value for the canvas sets it application wide.

- **virtual PvPluginDrawingStyle drawingStyle(PvPluginCanvasInterface* canvas)**

- **virtual void setDrawingWidth(int width, PvPluginCanvasInterface* canvas = NULL)**

  Set/Get the object's pen width on a per canvas basis. If the object does not have settable pen width, these do nothing. A NULL value for the canvas sets it application wide.

- **virtual int drawingWidth(PvPluginCanvasInterface* canvas)**

- **virtual PvSimObject* accessInternals()**

  Get at the internals - requires additional header files and libraries

- **virtual DtBoolean processMessage(const PvBaseObjectConduitMessage* message)**

  Processes a message from the main frame. Not part of the interface for the plugin creator.

- **PvPluginObjectInterface(const PvPluginObjectInterface& orig)**

  copy constructor

- **PvPluginObjectInterface& operator=(const PvPluginObjectInterface& orig)**

  assignment operator

- **virtual void deleteMarkedObjectCallbacks()**

  The remove function above merely mark the callbacks for deletion. This function actually deletes them. This is done to prevent someone from corrupting the list while the list is executing its callbacks.

- **PvSimObject* myObject**

- **PvPluginObjectListInterface* myObjectListInterface**

- **DtList* myObjectCbFunctions**

- **DtRIGHT_END_NOTIFIER(PvPluginObjectInterface)**

Direct child classes:

PvPluginPolygonObjectInterface
PvPluginPointObjectInterface
PvPluginLineObjectInterface
PvPluginInteractionObjectInterface
PvPluginEntityObjectInterface
PvPluginCircleObjectInterface

Alphabetic index *Hierarchy of classes*

In file plugcanvinter.h:

class PvPluginCanvasInterface

class PvPluginCanvasInterface: Instances of PvPluginCanvasInterface are the interface between the plugin and a specific plan view window

---

Public Methods

	PvPluginCanvasInterface (PvPlanView* planView, PvPluginAppInterface* appInterface) *constructor*
virtual	**~PvPluginCanvasInterface** () *destructor*
virtual void	**getCanvasCoordFromGeoc** (DtConstVector geocCoord, DtVector& canvasCoord) *Convert geocentric world/database coordinates from canvas coordinates*
virtual void	**getCanvasCoordFromDatabase** (DtConstVector databaseCoord, DtVector& canvasCoord)
virtual void	**getDatabaseCoordFromGeoc** (DtConstVector geocCoord, DtVector& databaseCoord) *Convert geocentric world/canvas coordinates to database coordinates*
virtual void	**getDatabaseCoordFromCanvas** (DtConstVector canvasCoord, DtVector& databaseCoord)
virtual void	**getGeocCoordFromDatabase** (DtConstVector databaseCoord, DtVector& geocCoord) *Convert database/canvas coordinates to geocentric world coordinates*
virtual void	**getGeocCoordFromCanvas** (DtConstVector canvasCoord, DtVector& geocCoord)
virtual void	**setCanvasExtentsInDatabaseCoord** (DtConstVector upperLeft, DtConstVector lowerRight) *Set/Get the rectangle that the canvas is viewing*
virtual void	**getCanvasExtentsInDatabaseCoord** (DtVector& upperLeft, DtVector& lowerRight)
virtual void	**getTerrainInformation** (DtConstVector databaseCoord, DtVector& intersectionCoord, PvPluginSurfaceType& surfaceType, DtVector& intersectNormal) *Get the height of terrain, soil type (if any), and ground normal at a particular*
virtual DtString	**databaseName** () *Get the filename of the database*
virtual void	**addItemToMenu** (DtString menu, DtString itemName, DtString itemHelp, PvPluginMenuCbFunc cb, void* usr) *Add an item to a top-level menu on the plan view menubar*
virtual void	**addMenuToMenu** (DtString menu, DtString newMenuName, DtString menuHelp) *Add a menu to a top-level menu on the plan view menubar*
virtual void	**addCallbackForTrackedObjects** (PvPluginTrackedObjCbFunc cb, void* usr) *Add/Remove a callback that is triggered when the tracked object(s) change*
virtual DtBoolean	**removeCallbackForTrackedObjects** (PvPluginTrackedObjCbFunc cb)
virtual void	**removeAllCallbacksForTrackedObjects** ()
virtual void	**addCallbackForUserEvents** (PvPluginUserEventCbFunc cb, void* usr) *Add/Remove a callback that is triggered for a user event*
virtual DtBoolean	**removeCallbackForUserEvents** (PvPluginUserEventCbFunc cb)
virtual void	**removeAllCallbacksForUserEvents** ()
virtual void	**addRightClickMenuItem** (DtString name, PvPluginObjectMenuCbFunc cb, void* usr) *Add a right click menu item*
virtual void	**getSelectedObjectInterfaces** (DtList* selectedList) *Get a list of interfaces to the currently selected objects*

virtual DtBoolean	**getSelectedPoint** (DtVector& databaseCoord) *Gets the currently selected point in database coordinates*
virtual PvPluginAppInterface*	**appInterface** () *Get the app interface*
virtual PvPlanView*	**accessInternals** () *Get at the internals - requires additional header files and libraries*
virtual DtBoolean	**processMessage** (const PvEntityGroupListConduitMessage* message) *Processes a message from the main frame*
virtual DtBoolean	**processMessage** (const PvCanvasSelectionConduitMessage* message) *Processes a message from the canvas*

Protected Fields

int	**myIdForMenuCreation**
PvPlanView*	**myPlanView**
PvPluginAppInterface*	**myAppInterface**
DtList*	**myTrackingCbFunctions**
DtList*	**myUserEventCbFunctions**

Protected Methods

	PvPluginCanvasInterface (const PvPluginCanvasInterface& orig) *copy constructor*
PvPluginCanvasInterface&	**operator=** (const PvPluginCanvasInterface& orig) *assignment operator*
virtual void	**deleteMarkedUserEventCallbacks** () *The remove functions above merely mark the callbacks for deletion*
virtual void	**deleteMarkedTrackedObjectCallbacks** ()
static void	**rightClickCb** (void* info, void* planViewCanvas) *Handle the right click menu callbacks*
static void	**menuCb** (void* info, void* planView) *Handle the menubar menu callbacks*
	DtRIGHT_END_NOTIFIER (PvPluginCanvasInterface)
	DtRIGHT_END_NOTIFIER_WITH_MOD (PvPluginCanvasInterface, Selection)

Documentation

class PvPluginCanvasInterface: Instances of PvPluginCanvasInterface are the interface between the plugin and a specific plan view window

`PvPluginCanvasInterface(PvPlanView* planView, PvPluginAppInterface* appInterface)`

constructor

`virtual ~PvPluginCanvasInterface()`

destructor

`virtual void getCanvasCoordFromGeoc(DtConstVector geocCoord, DtVector& canvasCoord)`

Convert geocentric world/database coordinates from canvas coordinates

- **virtual void getCanvasCoordFromDatabase(DtConstVector databaseCoord, DtVector& canvasCoord)**
- **virtual void getDatabaseCoordFromGeoc(DtConstVector geocCoord, DtVector& databaseCoord)**

  Convert geocentric world/canvas coordinates to database coordinates
- **virtual void getDatabaseCoordFromCanvas(DtConstVector canvasCoord, DtVector& databaseCoord)**
- **virtual void getGeocCoordFromDatabase(DtConstVector databaseCoord, DtVector& geocCoord)**

  Convert database/canvas coordinates to geocentric world coordinates
- **virtual void getGeocCoordFromCanvas(DtConstVector canvasCoord, DtVector& geocCoord)**
- **virtual void setCanvasExtentsInDatabaseCoord(DtConstVector upperLeft, DtConstVector lowerRight)**

  Set/Get the rectangle that the canvas is viewing
- **virtual void getCanvasExtentsInDatabaseCoord(DtVector& upperLeft, DtVector& lowerRight)**
- **virtual void getTerrainInformation(DtConstVector databaseCoord, DtVector& intersectionCoord, PvPluginSurfaceType& surfaceType, DtVector& intersectNormal)**

  Get the height of terrain, soil type (if any), and ground normal at a particular
- **virtual DtString databaseName()**

  Get the filename of the database
- **virtual void addItemToMenu(DtString menu, DtString itemName, DtString itemHelp, PvPluginMenuCbFunc cb, void* usr)**

  Add an item to a top-level menu on the plan view menubar
- **virtual void addMenuToMenu(DtString menu, DtString newMenuName, DtString menuHelp)**

  Add a menu to a top-level menu on the plan view menubar
- **virtual void addCallbackForTrackedObjects(PvPluginTrackedObjCbFunc cb, void* usr)**

  Add/Remove a callback that is triggered when the tracked object(s) change
- **virtual DtBoolean removeCallbackForTrackedObjects(PvPluginTrackedObjCbFunc cb)**
- **virtual void removeAllCallbacksForTrackedObjects()**
- **virtual void addCallbackForUserEvents(PvPluginUserEventCbFunc cb, void* usr)**

  Add/Remove a callback that is triggered for a user event
- **virtual DtBoolean removeCallbackForUserEvents(PvPluginUserEventCbFunc cb)**
- **virtual void removeAllCallbacksForUserEvents()**
- **virtual void addRightClickMenuItem(DtString name, PvPluginObjectMenuCbFunc cb, void* usr)**

  Add a right click menu item
- **virtual void getSelectedObjectInterfaces(DtList* selectedList)**

  Get a list of interfaces to the currently selected objects. SelectedList should be an empty list which will be filled with object interfaces.
- **virtual DtBoolean getSelectedPoint(DtVector& databaseCoord)**

  Gets the currently selected point in database coordinates. Returns DtFalse if no point is selected.
- **virtual PvPluginAppInterface* appInterface()**

  Get the app interface

- **virtual PvPlanView* accessInternals()**

  Get at the internals - requires additional header files and libraries

- **virtual DtBoolean processMessage(const PvEntityGroupListConduitMessage* message)**

  Processes a message from the main frame. Not part of the interface for the plugin creator.

- **virtual DtBoolean processMessage(const PvCanvasSelectionConduitMessage* message)**

  Processes a message from the canvas. Not part of the interface for the plugin creator.

- **PvPluginCanvasInterface(const PvPluginCanvasInterface& orig)**

  copy constructor

- **PvPluginCanvasInterface& operator=(const PvPluginCanvasInterface& orig)**

  assignment operator

- **virtual void deleteMarkedUserEventCallbacks()**

  The remove functions above merely mark the callbacks for deletion. These functions actually delete them. This is done to prevent someone from corrupting the list while the list is executing its callbacks.

- **virtual void deleteMarkedTrackedObjectCallbacks()**

- **static void rightClickCb(void* info, void* planViewCanvas)**

  Handle the right click menu callbacks

- **static void menuCb(void* info, void* planView)**

  Handle the menubar menu callbacks

- **int myIdForMenuCreation**

- **PvPlanView* myPlanView**

- **PvPluginAppInterface* myAppInterface**

- **DtList* myTrackingCbFunctions**

- **DtList* myUserEventCbFunctions**

- **DtRIGHT_END_NOTIFIER(PvPluginCanvasInterface)**

- **DtRIGHT_END_NOTIFIER_WITH_MOD(PvPluginCanvasInterface, Selection)**

This class has no child classes.

Alphabetic index *Hierarchy of classes*

In file plugpolyobjif.h:

class PvPluginPolygonObjectInterface: public PvPluginObjectInterface

class PvPluginPolygonObjectInterface: Instances of PvPluginPolygonObjectInterface are the interface between the plugin and a polygon sim object

---

Inheritance:

PvPluginPolygonObjectInterface - PvPluginObjectInterface

Public Classes

- enum **PvPluginFillStyle**
  - **PvPluginFillStyleSolid**
  - **PvPluginFillStyleHorizontalHatch**
  - **PvPluginFillStyleVerticalHatch**
  - **PvPluginFillStyleCrossHatch**
  - **PvPluginFillStyleDiagonalCrossHatch**
  - **PvPluginFillStyleDiagonalForwardHatch**
  - **PvPluginFillStyleDiagonalBackwardHatch**
  - **PvPluginFillStyleTransparent**

Public Methods

	PvPluginPolygonObjectInterface (PvPolygonSimObject* simObject, PvPluginObjectListInterface* interfaceList) *constructor*
virtual	**~PvPluginPolygonObjectInterface** () *destructor*
virtual DtConstVector	**vertex** (unsigned int vertexNumber) *Set/Get a particular vertex*
virtual void	**setVertex** (unsigned int vertexNumber, DtConstVector geocCoord)
virtual DtBoolean	**addVertexAfterVertex** (DtConstVector geocCoord, unsigned int vertexNumber) *Adds a vertex after another vertex*
virtual void	**addVertexToStart** (DtConstVector geocCoord) *Adds a vertex to the start/end*
virtual void	**addVertexToEnd** (DtConstVector geocCoord)

virtual DtBoolean	**deleteVertex** (unsigned int vertexNumber) *Removes a vertex*
virtual unsigned int	**numberOfVertices** () *Gets the number of vertices*
virtual DtList*	**listOfVertices** () *Get the list of vertices*
virtual void	**setFillColor** (unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL) *Set/Get the polygon's fill color on a per canvas basis*
virtual void	**getFillColor** (unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)
virtual void	**setFillStyle** (PvPluginFillStyle style, PvPluginCanvasInterface* canvas = NULL) *Set/Get the polygon's fill style on a per canvas basis*
virtual PvPluginFillStyle	**fillStyle** (PvPluginCanvasInterface* canvas)

Protected Fields

PvPolygonSimObject*	**myPolygonSimObject**

Protected Methods

	PvPluginPolygonObjectInterface (const PvPluginPolygonObjectInterface& orig) *copy constructor*
PvPluginPolygonObjectInterface&	**operator=** (const PvPluginPolygonObjectInterface& orig) *assignment operator*

Inherited from PvPluginObjectInterface:

Public Classes

- enum**PvPluginDrawingStyle**
- **PvPluginDrawingStyleSolid**
- **PvPluginDrawingStyleDot**
- **PvPluginDrawingStyleDash**
- **PvPluginDrawingStyleTransparent**

Public Methods

- virtual void **addCallbackOnObject**(PvPluginObjectCbFunc cb, void* usr)
- virtual DtBoolean **removeCallbackOnObject**(PvPluginObjectCbFunc cb)
- virtual void **removeAllCallbacksOnObject**()
- virtual PvObjectId **objectId**()
- virtual PvPluginObjectListInterface* **objectListInterface**()

- virtual void **update**()
- virtual void **setAnnotation**(const DtString& str)
- virtual const DtString& **annotation**() const
- virtual void **setSelection**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **selection**(PvPluginCanvasInterface* canvas)
- virtual void **setHidden**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **hidden**(PvPluginCanvasInterface* canvas)
- virtual void **setTimeoutValue**(DtTime time, PvPluginCanvasInterface* canvas = NULL)
- virtual DtTime **timeoutValue**(PvPluginCanvasInterface* canvas = NULL)
- virtual void **setColor**(unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL)
- virtual void **getColor**(unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)
- virtual void **setDrawingStyle**(PvPluginDrawingStyle style, PvPluginCanvasInterface* canvas = NULL)
- virtual PvPluginDrawingStyle **drawingStyle**(PvPluginCanvasInterface* canvas)
- virtual void **setDrawingWidth**(int width, PvPluginCanvasInterface* canvas = NULL)
- virtual int **drawingWidth**(PvPluginCanvasInterface* canvas)
- virtual PvSimObject* **accessInternals**()
- virtual DtBoolean **processMessage**(const PvBaseObjectConduitMessage* message)

Protected Fields

- PvSimObject* **myObject**
- PvPluginObjectListInterface* **myObjectListInterface**
- DtList* **myObjectCbFunctions**

Protected Methods

- virtual void **deleteMarkedObjectCallbacks**()
- **DtRIGHT_END_NOTIFIER**(PvPluginObjectInterface)

Documentation

class PvPluginPolygonObjectInterface:

Instances of PvPluginPolygonObjectInterface are the interface between the plugin and a polygon sim object. Vertices are represented by DtVectors in geocentric world coordinates. The last vertex is connected to the first.

- **enum PvPluginFillStyle**
- **PvPluginFillStyleSolid**
- **PvPluginFillStyleHorizontalHatch**
- **PvPluginFillStyleVerticalHatch**
- **PvPluginFillStyleCrossHatch**
- **PvPluginFillStyleDiagonalCrossHatch**
- **PvPluginFillStyleDiagonalForwardHatch**
- **PvPluginFillStyleDiagonalBackwardHatch**
- **PvPluginFillStyleTransparent**
- **PvPluginPolygonObjectInterface(PvPolygonSimObject* simObject, PvPluginObjectListInterface* interfaceList)**

  constructor
- **virtual ~PvPluginPolygonObjectInterface()**

  destructor
- **virtual DtConstVector vertex(unsigned int vertexNumber)**

  Set/Get a particular vertex. Vertices are numbered starting at zero. Returns a zero vector if the index is not found.
- **virtual void setVertex(unsigned int vertexNumber, DtConstVector geocCoord)**
- **virtual DtBoolean addVertexAfterVertex(DtConstVector geocCoord, unsigned int vertexNumber)**

  Adds a vertex after another vertex. Returns DtTrue if successful.
- **virtual void addVertexToStart(DtConstVector geocCoord)**

  Adds a vertex to the start/end
- **virtual void addVertexToEnd(DtConstVector geocCoord)**
- **virtual DtBoolean deleteVertex(unsigned int vertexNumber)**

  Removes a vertex. Returns DtTrue if successful.
- **virtual unsigned int numberOfVertices()**

  Gets the number of vertices
- **virtual DtList* listOfVertices()**

  Get the list of vertices. The list items contain pointers to DtVectors in geocentric world coordinates. It is recommended that you use the other functions to access or modify the list. If you do use this function to modify the list, be sure to call update() afterwards.
- **virtual void setFillColor(unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL)**

  Set/Get the polygon's fill color on a per canvas basis. A NULL value for the canvas sets it application wide.
- **virtual void getFillColor(unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)**
- **virtual void setFillStyle(PvPluginFillStyle style, PvPluginCanvasInterface* canvas = NULL)**

  Set/Get the polygon's fill style on a per canvas basis. A NULL value for the canvas sets it application wide.

- **virtual PvPluginFillStyle fillStyle(PvPluginCanvasInterface* canvas)**
- **PvPluginPolygonObjectInterface(const PvPluginPolygonObjectInterface& orig)**

  copy constructor
- **PvPluginPolygonObjectInterface& operator=(const PvPluginPolygonObjectInterface& orig)**

  assignment operator
- **PvPolygonSimObject* myPolygonSimObject**

This class has no child classes.

Alphabetic index *Hierarchy of classes*

In file plugpntobjif.h:

class PvPluginPointObjectInterface: public PvPluginObjectInterface

class PvPluginPointObjectInterface: Instances of PvPluginPointObjectInterface are the interface between the plugin and a point sim object

---

Inheritance:

PvPluginPointObjectInterface - PvPluginObjectInterface

Public Methods

	PvPluginPointObjectInterface (PvPointSimObject* simObject, PvPluginObjectListInterface* interfaceList) *constructor*
virtual	**~PvPluginPointObjectInterface** () *destructor*
virtual void	**setObjectGeocLocation** (DtConstVector geocCoord) *Set/Get the point location*
virtual DtConstVector	**objectGeocLocation** ()

Protected Fields

PvPointSimObject*	**myPointSimObject**

Protected Methods

	PvPluginPointObjectInterface (const PvPluginPointObjectInterface& orig) *copy constructor*
PvPluginPointObjectInterface&	**operator=** (const PvPluginPointObjectInterface& orig) *assignment operator*

Inherited from PvPluginObjectInterface:

Public Classes

- enum **PvPluginDrawingStyle**
- **PvPluginDrawingStyleSolid**
- **PvPluginDrawingStyleDot**
- **PvPluginDrawingStyleDash**
- **PvPluginDrawingStyleTransparent**

Public Methods

- virtual void **addCallbackOnObject**(PvPluginObjectCbFunc cb, void* usr)
- virtual DtBoolean **removeCallbackOnObject**(PvPluginObjectCbFunc cb)
- virtual void **removeAllCallbacksOnObject**()
- virtual PvObjectId **objectId**()
- virtual PvPluginObjectListInterface* **objectListInterface**()
- virtual void **update**()
- virtual void **setAnnotation**(const DtString& str)
- virtual const DtString& **annotation**() const
- virtual void **setSelection**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **selection**(PvPluginCanvasInterface* canvas)
- virtual void **setHidden**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **hidden**(PvPluginCanvasInterface* canvas)
- virtual void **setTimeoutValue**(DtTime time, PvPluginCanvasInterface* canvas = NULL)
- virtual DtTime **timeoutValue**(PvPluginCanvasInterface* canvas = NULL)
- virtual void **setColor**(unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL)
- virtual void **getColor**(unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)
- virtual void **setDrawingStyle**(PvPluginDrawingStyle style, PvPluginCanvasInterface* canvas = NULL)
- virtual PvPluginDrawingStyle **drawingStyle**(PvPluginCanvasInterface* canvas)
- virtual void **setDrawingWidth**(int width, PvPluginCanvasInterface* canvas = NULL)
- virtual int **drawingWidth**(PvPluginCanvasInterface* canvas)
- virtual PvSimObject* **accessInternals**()
- virtual DtBoolean **processMessage**(const PvBaseObjectConduitMessage* message)

Protected Fields

- PvSimObject* **myObject**
- PvPluginObjectListInterface* **myObjectListInterface**
- DtList* **myObjectCbFunctions**

Protected Methods

- virtual void **deleteMarkedObjectCallbacks**()
- **DtRIGHT_END_NOTIFIER**(PvPluginObjectInterface)

Documentation

class PvPluginPointObjectInterface: Instances of PvPluginPointObjectInterface are the interface between the plugin and a point sim object

- **`PvPluginPointObjectInterface(PvPointSimObject* simObject, PvPluginObjectListInterface* interfaceList)`**

  constructor

- **`virtual ~PvPluginPointObjectInterface()`**

  destructor

- **`virtual void setObjectGeocLocation(DtConstVector geocCoord)`**

  Set/Get the point location

- **`virtual DtConstVector objectGeocLocation()`**
- **`PvPluginPointObjectInterface(const PvPluginPointObjectInterface& orig)`**

  copy constructor

- **`PvPluginPointObjectInterface& operator=(const PvPluginPointObjectInterface& orig)`**

  assignment operator

- **`PvPointSimObject* myPointSimObject`**

Direct child classes:

PvPluginTextObjectInterface

Alphabetic index *Hierarchy of classes*

In file pluglineobjif.h:

class PvPluginLineObjectInterface: public PvPluginObjectInterface

class PvPluginLineObjectInterface: Instances of PvPluginLineObjectInterface are the interface between the plugin and a line sim object

---

Inheritance:

PvPluginLineObjectInterface - PvPluginObjectInterface

Public Methods

	PvPluginLineObjectInterface (PvLineSimObject* simObject, PvPluginObjectListInterface* interfaceList) *constructor*
virtual	**~PvPluginLineObjectInterface** () *destructor*
virtual DtConstVector	**vertex** (unsigned int vertexNumber) *Set/Get a particular vertex*
virtual void	**setVertex** (unsigned int vertexNumber, DtConstVector geocCoord)
virtual DtBoolean	**addVertexAfterVertex** (DtConstVector geocCoord, unsigned int vertexNumber) *Adds a vertex after another vertex*
virtual void	**addVertexToStart** (DtConstVector geocCoord) *Adds a vertex to the start/end*
virtual void	**addVertexToEnd** (DtConstVector geocCoord)
virtual DtBoolean	**deleteVertex** (unsigned int vertexNumber) *Removes a vertex*
virtual unsigned int	**numberOfVertices** () *Gets the number of vertices*
virtual DtList*	**listOfVertices** () *Get the list of vertices*

Protected Fields

PvLineSimObject*	**myLineSimObject**

Protected Methods

- **PvPluginLineObjectInterface** (const PvPluginLineObjectInterface& orig)
  copy constructor
- PvPluginLineObjectInterface& **operator=** (const PvPluginLineObjectInterface& orig)
  assignment operator

Inherited from PvPluginObjectInterface:

Public Classes

- enum**PvPluginDrawingStyle**
- **PvPluginDrawingStyleSolid**
- **PvPluginDrawingStyleDot**
- **PvPluginDrawingStyleDash**
- **PvPluginDrawingStyleTransparent**

Public Methods

- virtual void **addCallbackOnObject**(PvPluginObjectCbFunc cb, void* usr)
- virtual DtBoolean **removeCallbackOnObject**(PvPluginObjectCbFunc cb)
- virtual void **removeAllCallbacksOnObject**()
- virtual PvObjectId **objectId**()
- virtual PvPluginObjectListInterface* **objectListInterface**()
- virtual void **update**()
- virtual void **setAnnotation**(const DtString& str)
- virtual const DtString& **annotation**() const
- virtual void **setSelection**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **selection**(PvPluginCanvasInterface* canvas)
- virtual void **setHidden**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **hidden**(PvPluginCanvasInterface* canvas)
- virtual void **setTimeoutValue**(DtTime time, PvPluginCanvasInterface* canvas = NULL)
- virtual DtTime **timeoutValue**(PvPluginCanvasInterface* canvas = NULL)
- virtual void **setColor**(unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL)
- virtual void **getColor**(unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)

- virtual void **setDrawingStyle**(PvPluginDrawingStyle style, PvPluginCanvasInterface* canvas = NULL)
- virtual PvPluginDrawingStyle **drawingStyle**(PvPluginCanvasInterface* canvas)
- virtual void **setDrawingWidth**(int width, PvPluginCanvasInterface* canvas = NULL)
- virtual int **drawingWidth**(PvPluginCanvasInterface* canvas)
- virtual PvSimObject* **accessInternals**()
- virtual DtBoolean **processMessage**(const PvBaseObjectConduitMessage* message)

Protected Fields

- PvSimObject* **myObject**
- PvPluginObjectListInterface* **myObjectListInterface**
- DtList* **myObjectCbFunctions**

Protected Methods

- virtual void **deleteMarkedObjectCallbacks**()
- **DtRIGHT_END_NOTIFIER**(PvPluginObjectInterface)

Documentation

class PvPluginLineObjectInterface:

Instances of PvPluginLineObjectInterface are the interface between the plugin and a line sim object. Vertices are represented by DtVectors in geocentric world coordinates.

- **`PvPluginLineObjectInterface(PvLineSimObject* simObject, PvPluginObjectListInterface* interfaceList)`**

  constructor

- **`virtual ~PvPluginLineObjectInterface()`**

  destructor

- **`virtual DtConstVector vertex(unsigned int vertexNumber)`**

  Set/Get a particular vertex. Vertices are numbered starting at zero. Returns a zero vector if the index is not found.

- **`virtual void setVertex(unsigned int vertexNumber, DtConstVector geocCoord)`**
- **`virtual DtBoolean addVertexAfterVertex(DtConstVector geocCoord, unsigned int vertexNumber)`**

  Adds a vertex after another vertex. Returns DtTrue if successful.

- **`virtual void addVertexToStart(DtConstVector geocCoord)`**

  Adds a vertex to the start/end

- **`virtual void addVertexToEnd(DtConstVector geocCoord)`**
- **`virtual DtBoolean deleteVertex(unsigned int vertexNumber)`**

Removes a vertex. Returns DtTrue if successful.

- **`virtual unsigned int numberOfVertices()`**

  Gets the number of vertices

- **`virtual DtList* listOfVertices()`**

  Get the list of vertices. The list items contain pointers to DtVectors in geocentric world coordinates. It is recommended that you use the other functions to access or modify the list. If you do use this function to modify the list, be sure to call update() afterwards.

- **`PvPluginLineObjectInterface(const PvPluginLineObjectInterface& orig)`**

  copy constructor

- **`PvPluginLineObjectInterface& operator=(const PvPluginLineObjectInterface& orig)`**

  assignment operator

- **`PvLineSimObject* myLineSimObject`**

This class has no child classes.

Alphabetic index *Hierarchy of classes*

In file plugintobjif.h:

class PvPluginInteractionObjectInterface: public PvPluginObjectInterface

class PvPluginInteractionObjectInterface: Instances of PvPluginInteractionObjectInterface are the interface between the plugin and a base interaction

---

Inheritance:

PvPluginInteractionObjectInterface - PvPluginObjectInterface

Public Methods

	PvPluginInteractionObjectInterface (PvBaseInteractionSimObject* simObject, PvPluginObjectListInterface* interfaceList) *constructor*
virtual	**~PvPluginInteractionObjectInterface** () *destructor*
virtual DtInteraction*	**interaction** () *Get the interaction*

Protected Fields

PvBaseInteractionSimObject*	**myInteractionSimObject**

Protected Methods

	PvPluginInteractionObjectInterface (const PvPluginInteractionObjectInterface& orig) *copy constructor*
PvPluginInteractionObjectInterface&	**operator=** (const PvPluginInteractionObjectInterface& orig) *assignment operator*

Inherited from PvPluginObjectInterface:

Public Classes

- enum **PvPluginDrawingStyle**
- **PvPluginDrawingStyleSolid**
- **PvPluginDrawingStyleDot**
- **PvPluginDrawingStyleDash**
- **PvPluginDrawingStyleTransparent**

Public Methods

- virtual void **addCallbackOnObject**(PvPluginObjectCbFunc cb, void* usr)
- virtual DtBoolean **removeCallbackOnObject**(PvPluginObjectCbFunc cb)
- virtual void **removeAllCallbacksOnObject**()
- virtual PvObjectId **objectId**()
- virtual PvPluginObjectListInterface* **objectListInterface**()
- virtual void **update**()
- virtual void **setAnnotation**(const DtString& str)
- virtual const DtString& **annotation**() const
- virtual void **setSelection**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **selection**(PvPluginCanvasInterface* canvas)
- virtual void **setHidden**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **hidden**(PvPluginCanvasInterface* canvas)
- virtual void **setTimeoutValue**(DtTime time, PvPluginCanvasInterface* canvas = NULL)
- virtual DtTime **timeoutValue**(PvPluginCanvasInterface* canvas = NULL)
- virtual void **setColor**(unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL)
- virtual void **getColor**(unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)
- virtual void **setDrawingStyle**(PvPluginDrawingStyle style, PvPluginCanvasInterface* canvas = NULL)
- virtual PvPluginDrawingStyle **drawingStyle**(PvPluginCanvasInterface* canvas)
- virtual void **setDrawingWidth**(int width, PvPluginCanvasInterface* canvas = NULL)
- virtual int **drawingWidth**(PvPluginCanvasInterface* canvas)
- virtual PvSimObject* **accessInternals**()
- virtual DtBoolean **processMessage**(const PvBaseObjectConduitMessage* message)

Protected Fields

- PvSimObject* **myObject**
- PvPluginObjectListInterface* **myObjectListInterface**
- DtList* **myObjectCbFunctions**

Protected Methods

- virtual void **deleteMarkedObjectCallbacks**()
- **DtRIGHT_END_NOTIFIER**(PvPluginObjectInterface)

Documentation

class PvPluginInteractionObjectInterface: Instances of PvPluginInteractionObjectInterface are the interface between the plugin and a base interaction

- **`PvPluginInteractionObjectInterface(PvBaseInteractionSimObject* simObject, PvPluginObjectListInterface* interfaceList)`**

  constructor

- **`virtual ~PvPluginInteractionObjectInterface()`**

  destructor

- **`virtual DtInteraction* interaction()`**

  Get the interaction

- **`PvPluginInteractionObjectInterface(const PvPluginInteractionObjectInterface& orig)`**

  copy constructor

- **`PvPluginInteractionObjectInterface& operator=(const PvPluginInteractionObjectInterface& orig)`**

  assignment operator

- **`PvBaseInteractionSimObject* myInteractionSimObject`**

This class has no child classes.

Alphabetic index *Hierarchy of classes*

In file plugentobjif.h:

class PvPluginEntityObjectInterface: public PvPluginObjectInterface

class PvPluginEntityObjectInterface: Instances of PvPluginEntityObjectInterface are the interface between the plugin and a single entity sim object

---

Inheritance:

PvPluginEntityObjectInterface - PvPluginObjectInterface

Public Methods

	PvPluginEntityObjectInterface (PvSingleEntitySimObject* simObject, PvPluginObjectListInterface* interfaceList) *constructor*
virtual	**~PvPluginEntityObjectInterface** () *destructor*
virtual void	**setEsr** (DtEntityStateRepository* esr) *Set/Get the entity state repository*
virtual DtEntityStateRepository*	**esr** ()
virtual const DtGlobalObjectDesignator&	**globalId** () const *Get the entity's object's global object designator*

Protected Fields

PvSingleEntitySimObject*	**myEntitySimObject**

Protected Methods

	PvPluginEntityObjectInterface (const PvPluginEntityObjectInterface& orig) *copy constructor*
PvPluginEntityObjectInterface&	**operator=** (const PvPluginEntityObjectInterface& orig) *assignment operator*

Inherited from PvPluginObjectInterface:

Public Classes

- enum **PvPluginDrawingStyle**
- **PvPluginDrawingStyleSolid**
- **PvPluginDrawingStyleDot**
- **PvPluginDrawingStyleDash**
- **PvPluginDrawingStyleTransparent**

Public Methods

- virtual void **addCallbackOnObject**(PvPluginObjectCbFunc cb, void* usr)
- virtual DtBoolean **removeCallbackOnObject**(PvPluginObjectCbFunc cb)
- virtual void **removeAllCallbacksOnObject**()
- virtual PvObjectId **objectId**()
- virtual PvPluginObjectListInterface* **objectListInterface**()
- virtual void **update**()
- virtual void **setAnnotation**(const DtString& str)
- virtual const DtString& **annotation**() const
- virtual void **setSelection**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **selection**(PvPluginCanvasInterface* canvas)
- virtual void **setHidden**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **hidden**(PvPluginCanvasInterface* canvas)
- virtual void **setTimeoutValue**(DtTime time, PvPluginCanvasInterface* canvas = NULL)
- virtual DtTime **timeoutValue**(PvPluginCanvasInterface* canvas = NULL)
- virtual void **setColor**(unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL)
- virtual void **getColor**(unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)
- virtual void **setDrawingStyle**(PvPluginDrawingStyle style, PvPluginCanvasInterface* canvas = NULL)
- virtual PvPluginDrawingStyle **drawingStyle**(PvPluginCanvasInterface* canvas)
- virtual void **setDrawingWidth**(int width, PvPluginCanvasInterface* canvas = NULL)
- virtual int **drawingWidth**(PvPluginCanvasInterface* canvas)
- virtual PvSimObject* **accessInternals**()

- virtual DtBoolean **processMessage**(const PvBaseObjectConduitMessage* message)

Protected Fields

- PvSimObject* **myObject**
- PvPluginObjectListInterface* **myObjectListInterface**
- DtList* **myObjectCbFunctions**

Protected Methods

- virtual void **deleteMarkedObjectCallbacks**()
- **DtRIGHT_END_NOTIFIER**(PvPluginObjectInterface)

Documentation

class PvPluginEntityObjectInterface: Instances of PvPluginEntityObjectInterface are the interface between the plugin and a single entity sim object

- **`PvPluginEntityObjectInterface(PvSingleEntitySimObject* simObject, PvPluginObjectListInterface* interfaceList)`**

  constructor

- **`virtual ~PvPluginEntityObjectInterface()`**

  destructor

- **`virtual void setEsr(DtEntityStateRepository* esr)`**

  Set/Get the entity state repository

- **`virtual DtEntityStateRepository* esr()`**
- **`virtual const DtGlobalObjectDesignator& globalId() const`**

  Get the entity's object's global object designator

- **`PvPluginEntityObjectInterface(const PvPluginEntityObjectInterface& orig)`**

  copy constructor

- **`PvPluginEntityObjectInterface& operator=(const PvPluginEntityObjectInterface& orig)`**

  assignment operator

- **`PvSingleEntitySimObject* myEntitySimObject`**

This class has no child classes.

Alphabetic index *Hierarchy of classes*

In file plugcircobjif.h:

class PvPluginCircleObjectInterface: public PvPluginObjectInterface

class PvPluginCircleObjectInterface: Instances of PvPluginCircleObjectInterface are the interface between the plugin and a circle sim object

---

Inheritance:

PvPluginCircleObjectInterface - PvPluginObjectInterface

Public Classes

- enum **PvPluginFillStyle**
  - **PvPluginFillStyleSolid**
  - **PvPluginFillStyleHorizontalHatch**
  - **PvPluginFillStyleVerticalHatch**
  - **PvPluginFillStyleCrossHatch**
  - **PvPluginFillStyleDiagonalCrossHatch**
  - **PvPluginFillStyleDiagonalForwardHatch**
  - **PvPluginFillStyleDiagonalBackwardHatch**
  - **PvPluginFillStyleTransparent**

Public Methods

	PvPluginCircleObjectInterface (PvCircleSimObject* simObject, PvPluginObjectListInterface* interfaceList) *constructor*
virtual	**~PvPluginCircleObjectInterface** () *destructor*
virtual void	**setCenterGeocLocation** (DtConstVector geocCoord) *Set/Get the circle's center location in geocentric world coordinates*
virtual DtConstVector	**centerGeocLocation** ()
virtual void	**setCircumferenceGeocLocation** (DtConstVector geocCoord) *Set/Get the circle's circumference location in geocentric world coordinates*
virtual DtConstVector	**circumferenceGeocLocation** ()
virtual void	**setFillColor** (unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL) *Set/Get the circle's fill color on a per canvas basis*

- virtual void **getFillColor** (unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)
- virtual void **setFillStyle** (PvPluginFillStyle style, PvPluginCanvasInterface* canvas = NULL)
  Set/Get the circle's fill style on a per canvas basis
- virtual PvPluginFillStyle **fillStyle** (PvPluginCanvasInterface* canvas)

Protected Fields

- PvCircleSimObject* **myCircleSimObject**

Protected Methods

- **PvPluginCircleObjectInterface** (const PvPluginCircleObjectInterface& orig)
  copy constructor
- PvPluginCircleObjectInterface& **operator=** (const PvPluginCircleObjectInterface& orig)
  assignment operator

Inherited from PvPluginObjectInterface:

Public Classes

- enum**PvPluginDrawingStyle**
- **PvPluginDrawingStyleSolid**
- **PvPluginDrawingStyleDot**
- **PvPluginDrawingStyleDash**
- **PvPluginDrawingStyleTransparent**

Public Methods

- virtual void **addCallbackOnObject**(PvPluginObjectCbFunc cb, void* usr)
- virtual DtBoolean **removeCallbackOnObject**(PvPluginObjectCbFunc cb)
- virtual void **removeAllCallbacksOnObject**()
- virtual PvObjectId **objectId**()
- virtual PvPluginObjectListInterface* **objectListInterface**()
- virtual void **update**()
- virtual void **setAnnotation**(const DtString& str)
- virtual const DtString& **annotation**() const
- virtual void **setSelection**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)

- virtual DtBoolean **selection**(PvPluginCanvasInterface* canvas)
- virtual void **setHidden**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **hidden**(PvPluginCanvasInterface* canvas)
- virtual void **setTimeoutValue**(DtTime time, PvPluginCanvasInterface* canvas = NULL)
- virtual DtTime **timeoutValue**(PvPluginCanvasInterface* canvas = NULL)
- virtual void **setColor**(unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL)
- virtual void **getColor**(unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)
- virtual void **setDrawingStyle**(PvPluginDrawingStyle style, PvPluginCanvasInterface* canvas = NULL)
- virtual PvPluginDrawingStyle **drawingStyle**(PvPluginCanvasInterface* canvas)
- virtual void **setDrawingWidth**(int width, PvPluginCanvasInterface* canvas = NULL)
- virtual int **drawingWidth**(PvPluginCanvasInterface* canvas)
- virtual PvSimObject* **accessInternals**()
- virtual DtBoolean **processMessage**(const PvBaseObjectConduitMessage* message)

Protected Fields

- PvSimObject* **myObject**
- PvPluginObjectListInterface* **myObjectListInterface**
- DtList* **myObjectCbFunctions**

Protected Methods

- virtual void **deleteMarkedObjectCallbacks**()
- **DtRIGHT_END_NOTIFIER**(PvPluginObjectInterface)

Documentation

class PvPluginCircleObjectInterface: Instances of PvPluginCircleObjectInterface are the interface between the plugin and a circle sim object

- **`enum PvPluginFillStyle`**
- **`PvPluginFillStyleSolid`**
- **`PvPluginFillStyleHorizontalHatch`**
- **`PvPluginFillStyleVerticalHatch`**
- **`PvPluginFillStyleCrossHatch`**

- **PvPluginFillStyleDiagonalCrossHatch**
- **PvPluginFillStyleDiagonalForwardHatch**
- **PvPluginFillStyleDiagonalBackwardHatch**
- **PvPluginFillStyleTransparent**
- **PvPluginCircleObjectInterface(PvCircleSimObject* simObject, PvPluginObjectListInterface* interfaceList)**

  constructor
- **virtual ~PvPluginCircleObjectInterface()**

  destructor
- **virtual void setCenterGeocLocation(DtConstVector geocCoord)**

  Set/Get the circle's center location in geocentric world coordinates
- **virtual DtConstVector centerGeocLocation()**
- **virtual void setCircumferenceGeocLocation(DtConstVector geocCoord)**

  Set/Get the circle's circumference location in geocentric world coordinates
- **virtual DtConstVector circumferenceGeocLocation()**
- **virtual void setFillColor(unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL)**

  Set/Get the circle's fill color on a per canvas basis. A NULL value for the canvas sets it application wide.
- **virtual void getFillColor(unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)**
- **virtual void setFillStyle(PvPluginFillStyle style, PvPluginCanvasInterface* canvas = NULL)**

  Set/Get the circle's fill style on a per canvas basis. A NULL value for the canvas sets it application wide.
- **virtual PvPluginFillStyle fillStyle(PvPluginCanvasInterface* canvas)**
- **PvPluginCircleObjectInterface(const PvPluginCircleObjectInterface& orig)**

  copy constructor
- **PvPluginCircleObjectInterface& operator=(const PvPluginCircleObjectInterface& orig)**

  assignment operator
- **PvCircleSimObject* myCircleSimObject**

This class has no child classes.

Alphabetic index *Hierarchy of classes*

In file plugcanvinter.h:

enum PvPluginSurfaceType

- **PvSurfaceWet**
- **PvSurfaceWetOcean**
- **PvSurfaceWetLake**
- **PvSurfaceWetRiver**
- **PvSurfaceWetSwamp**
- **PvSurfaceWetMuck**
- **PvSurfacePaved**
- **PvSurfacePavedRoad**
- **PvSurfaceUnpaved**
- **PvSurfaceUnpavedSlowGo**
- **PvSurfaceUnpavedNoGo**
- **PvSurfaceOther**
- **PvSurfaceUnspecified**

Documentation

- `PvSurfaceWet`
- `PvSurfaceWetOcean`
- `PvSurfaceWetLake`
- `PvSurfaceWetRiver`
- `PvSurfaceWetSwamp`
- `PvSurfaceWetMuck`
- `PvSurfacePaved`
- `PvSurfacePavedRoad`
- `PvSurfaceUnpaved`
- `PvSurfaceUnpavedSlowGo`
- `PvSurfaceUnpavedNoGo`

- **PvSurfaceOther**
- **PvSurfaceUnspecified**

Alphabetic index *Hierarchy of classes*

In file plugtextobjif.h:

class PvPluginTextObjectInterface: public PvPluginPointObjectInterface

---

Inheritance:

PvPluginTextObjectInterface - PvPluginPointObjectInterface - PvPluginObjectInterface

---

Public Classes

- enum **PvPluginFontFamilyType**
  - **PvPluginFontFamilyRoman**
  - **PvPluginFontFamilyModern**
  - **PvPluginFontFamilyDefault**
- enum **PvPluginFontStyleType**
  - **PvPluginFontStyleNormal**
  - **PvPluginFontStyleItalic**
- enum **PvPluginFontWeightType**
  - **PvPluginFontWeightNormal**
  - **PvPluginFontWeightBold**

Public Methods

	PvPluginTextObjectInterface (PvTextSimObject* simObject, PvPluginObjectListInterface* interfaceList) *constructor*
virtual	**~PvPluginTextObjectInterface** () *destructor*
virtual void	**setString** (const DtString& str) *Set/Get the object's text string*
virtual const DtString&	**string** () const
virtual void	**setFontFamily** (PvPluginFontFamilyType family, PvPluginCanvasInterface* canvas = NULL) *Set/Get the object's font information*
virtual PvPluginFontFamilyType	**fontFamily** (PvPluginCanvasInterface* canvas)
virtual void	**setFontStyle** (PvPluginFontStyleType style, PvPluginCanvasInterface* canvas = NULL)

- virtual PvPluginFontStyleType **fontStyle** (PvPluginCanvasInterface* canvas)
- virtual void **setFontWeight** (PvPluginFontWeightType weight, PvPluginCanvasInterface* canvas = NULL)
- virtual PvPluginFontWeightType **fontWeight** (PvPluginCanvasInterface* canvas)
- virtual void **setFontSize** (int pointSize, PvPluginCanvasInterface* canvas = NULL)
- virtual int **fontSize** (PvPluginCanvasInterface* canvas)

Protected Fields

- PvTextSimObject* **myTextSimObject**

Protected Methods

- **PvPluginTextObjectInterface** (const PvPluginTextObjectInterface& orig)
  copy constructor
- PvPluginTextObjectInterface& **operator=** (const PvPluginTextObjectInterface& orig)
  assignment operator

Inherited from PvPluginPointObjectInterface:

Public Methods

- virtual void **setObjectGeocLocation**(DtConstVector geocCoord)
- virtual DtConstVector **objectGeocLocation**()

Protected Fields

- PvPointSimObject* **myPointSimObject**

Inherited from PvPluginObjectInterface:

Public Classes

- enum**PvPluginDrawingStyle**
- **PvPluginDrawingStyleSolid**
- **PvPluginDrawingStyleDot**
- **PvPluginDrawingStyleDash**
- **PvPluginDrawingStyleTransparent**

Public Methods

- virtual void **addCallbackOnObject**(PvPluginObjectCbFunc cb, void* usr)
- virtual DtBoolean **removeCallbackOnObject**(PvPluginObjectCbFunc cb)
- virtual void **removeAllCallbacksOnObject**()
- virtual PvObjectId **objectId**()
- virtual PvPluginObjectListInterface* **objectListInterface**()
- virtual void **update**()
- virtual void **setAnnotation**(const DtString& str)
- virtual const DtString& **annotation**() const
- virtual void **setSelection**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **selection**(PvPluginCanvasInterface* canvas)
- virtual void **setHidden**(DtBoolean value, PvPluginCanvasInterface* canvas = NULL)
- virtual DtBoolean **hidden**(PvPluginCanvasInterface* canvas)
- virtual void **setTimeoutValue**(DtTime time, PvPluginCanvasInterface* canvas = NULL)
- virtual DtTime **timeoutValue**(PvPluginCanvasInterface* canvas = NULL)
- virtual void **setColor**(unsigned char red, unsigned char green, unsigned char blue, PvPluginCanvasInterface* canvas = NULL)
- virtual void **getColor**(unsigned char* red, unsigned char* green, unsigned char* blue, PvPluginCanvasInterface* canvas)
- virtual void **setDrawingStyle**(PvPluginDrawingStyle style, PvPluginCanvasInterface* canvas = NULL)
- virtual PvPluginDrawingStyle **drawingStyle**(PvPluginCanvasInterface* canvas)
- virtual void **setDrawingWidth**(int width, PvPluginCanvasInterface* canvas = NULL)
- virtual int **drawingWidth**(PvPluginCanvasInterface* canvas)
- virtual PvSimObject* **accessInternals**()
- virtual DtBoolean **processMessage**(const PvBaseObjectConduitMessage* message)

Protected Fields

- PvSimObject* **myObject**
- PvPluginObjectListInterface* **myObjectListInterface**
- DtList* **myObjectCbFunctions**

Protected Methods

- virtual void **deleteMarkedObjectCallbacks**()
- **DtRIGHT_END_NOTIFIER**(PvPluginObjectInterface)

Documentation

- **enum PvPluginFontFamilyType**
- **PvPluginFontFamilyRoman**
- **PvPluginFontFamilyModern**
- **PvPluginFontFamilyDefault**
- **enum PvPluginFontStyleType**
- **PvPluginFontStyleNormal**
- **PvPluginFontStyleItalic**
- **enum PvPluginFontWeightType**
- **PvPluginFontWeightNormal**
- **PvPluginFontWeightBold**
- **PvPluginTextObjectInterface(PvTextSimObject* simObject, PvPluginObjectListInterface* interfaceList)**

  constructor
- **virtual ~PvPluginTextObjectInterface()**

  destructor
- **virtual void setString(const DtString& str)**

  Set/Get the object's text string
- **virtual const DtString& string() const**
- **virtual void setFontFamily(PvPluginFontFamilyType family, PvPluginCanvasInterface* canvas = NULL)**

  Set/Get the object's font information. The get routines requires a canvas. The set routines will apply the change to all canvases if none is provided. For now, these routines cannot be called from within an object added callback because the canvas may not have added the object yet.
- **virtual PvPluginFontFamilyType fontFamily(PvPluginCanvasInterface* canvas)**
- **virtual void setFontStyle(PvPluginFontStyleType style, PvPluginCanvasInterface* canvas = NULL)**
- **virtual PvPluginFontStyleType fontStyle(PvPluginCanvasInterface* canvas)**
- **virtual void setFontWeight(PvPluginFontWeightType weight, PvPluginCanvasInterface* canvas = NULL)**
- **virtual PvPluginFontWeightType fontWeight(PvPluginCanvasInterface* canvas)**

- **virtual void setFontSize(int pointSize, PvPluginCanvasInterface* canvas = NULL)**
- **virtual int fontSize(PvPluginCanvasInterface* canvas)**
- **PvPluginTextObjectInterface(const PvPluginTextObjectInterface& orig)**

  copy constructor
- **PvPluginTextObjectInterface& operator=(const PvPluginTextObjectInterface& orig)**

  assignment operator
- **PvTextSimObject* myTextSimObject**

This class has no child classes.

Alphabetic index *Hierarchy of classes*

Plan View Display 1.2 Documentation for Plug-in Classes
Alphabetic index *Hierarchy of classes*

- #define **PvPluginAppInterface_H_**
- #define **EXPORT**(dllexport)
- #define **EXPORT**
- typedef void **(*PvPluginPvCbFunc)**(PvPluginCanvasInterface* source, PvPluginGenericMessageType type, void* usr, PvPluginAppInterface* myAppInterface)
- typedef void **(*PvPluginTickCbFunc)**(void* usr, PvPluginAppInterface* myAppInterface)
- **DtDECLARE_DEFINE_RIGHT_END_CLASS**(PvPluginAppInterface, const PvCanvasWindowConduitMessage*)
- #define **PvPluginCanvasInterface_H_**
- typedef void **(*PvPluginMenuCbFunc)**(void* usr, void* canvasInterface)
- typedef void **(*PvPluginObjectMenuCbFunc)**(void* usr, void* canvasInterface)
- typedef void **(*PvPluginTrackedObjCbFunc)**(DtList* trackedObjectInterfaces, void* usr, PvPluginCanvasInterface* myCanvasInterface)
- typedef void **(*PvPluginUserEventCbFunc)**(PvPluginUserEventType type, DtConstVector geocCoord, DtList* objectInterfaces, void* usr, PvPluginCanvasInterface* myCanvasInterface)
- **DtDECLARE_DEFINE_RIGHT_END_CLASS**(PvPluginCanvasInterface, const PvEntityGroupListConduitMessage*)
- **DtDECLARE_DEFINE_RIGHT_END_CLASS_WITH_MOD**(PvPluginCanvasInterface, Selection, const PvCanvasSelectionConduitMessage*)
- #define **PvPluginCircleObjectInterface_H_**
- #define **PvPluginEntityObjectInterface_H_**
- #define **PvPluginInteractionObjectInterface_H_**
- #define **PvPluginLineObjectInterface_H_**
- #define **PvPluginObjectInterface_H_**
- typedef void **(*PvPluginObjectCbFunc)**(PvPluginGenericMessageType type, void* usr, PvPluginObjectInterface* myObjInterface)
- **DtDECLARE_DEFINE_RIGHT_END_CLASS**(PvPluginObjectInterface, const PvBaseObjectConduitMessage*)
- #define **PvPluginObjectListInterface_H_**
- typedef void **(*PvPluginObjectListCbFunc)**(PvPluginGenericMessageType type, PvObjectId id, void* usr, PvPluginObjectListInterface* myObjListInterface)
- **DtDECLARE_DEFINE_RIGHT_END_CLASS**(PvPluginObjectListInterface, const PvBaseObjectConduitMessage*)
- #define **PvPluginPointObjectInterface_H_**
- #define **PvPluginPolygonObjectInterface_H_**
- #define **PvPluginTextObjectInterface_H_**

Alphabetic index *Hierarchy of classes*

In file plugappinter.h:

enum PvPluginGenericMessageType

- **PvPluginGenericAddedMessage**
- **PvPluginGenericChangedMessage**
- **PvPluginGenericDeletedMessage**
- **PvPluginGenericOtherMessage**

Documentation

- `PvPluginGenericAddedMessage`
- `PvPluginGenericChangedMessage`
- `PvPluginGenericDeletedMessage`
- `PvPluginGenericOtherMessage`

In file plugcanvinter.h:

enum PvPluginUserEventType

- **PvPluginObjectSelectionChanged**
- **PvPluginGroundSelection**

Documentation

- `PvPluginObjectSelectionChanged`
- `PvPluginGroundSelection`
