Autodesk Navisworks COM Interface

User Guide

March 2015

Legal Notices

Trademarks

The following are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and other countries: 123D, 3ds Max, Algor, Alias, Alias (swirl design/logo), AliasStudio, ATC, AUGI, AutoCAD, AutoCAD Learning Assistance, AutoCAD LT, AutoCAD Simulator, AutoCAD SQL Extension, AutoCAD SQL Interface, Autodesk, Autodesk Homestyler, Autodesk Intent, Autodesk Inventor, Autodesk MapGuide, Autodesk Streamline, AutoLISP, AutoSketch, AutoSnap, AutoTrack, Backburner, Backdraft, Beast, Beast (design/logo) Built with ObjectARX (design/logo), Burn, Buzzsaw, CAiCE, CFdesign, Civil 3D, Cleaner, Cleaner Central, ClearScale, Colour Warper, Combustion, Communication Specification, Constructware, Content Explorer, Creative Bridge, Dancing Baby (image), DesignCenter, Design Doctor, Designer's Toolkit, DesignKids, DesignProf, DesignServer, DesignStudio, Design Web Format, Discreet, DWF, DWG, DWG (design/logo), DWG Extreme, DWG TrueConvert, DWG TrueView, DWFX, DXF, Ecotect, Evolver, Exposure, Extending the Design Team, Face Robot, FBX, Fempro, Fire, Flame, Flare, Flint, FMDesktop, Freewheel, GDX Driver, Green Building Studio, Heads-up Design, Heidi, Homestyler, HumanIK, IDEA Server, i-drop, Illuminate Labs AB (design/logo), ImageModeler, iMOUT, Incinerator, Inferno, Instructables, Instructables (stylized robot design/logo),Inventor, Inventor LT, Kynapse, Kynogon, LandXplorer, LiquidLight, LiquidLight (design/logo), Lustre, MatchMover, Maya, Mechanical Desktop, Moldflow, Moldflow Plastics Advisers, Moldflow Plastics Insight, Moldflow Plastics Xpert, Moondust, MotionBuilder, Movimento, MPA, MPA (design/logo), MPI, MPI (design/logo), MPX, MPX (design/logo), Mudbox, Multi-Master Editing, Navisworks, ObjectARX, ObjectDBX, Opticore, Pipeplus, Pixlr, Pixlr-o-matic, PolarSnap, PortfolioWall, Powered with Autodesk Technology, Productstream, ProMaterials, RasterDWG, RealDWG, Real-time Roto, Recognize, Render Queue, Retimer, Reveal, Revit, RiverCAD, Robot, Scaleform, Scaleform GFx, Showcase, Show Me, ShowMotion, SketchBook, Smoke, Softimage, Softimage|XSI (design/logo), Sparks, SteeringWheels, Stitcher, Stone, StormNET, Tinkerbox, ToolClip, Topobase, Toxik, TrustedDWG, U-Vis, ViewCube, Visual, Visual LISP, Voice Reality, Volo, Vtour, WaterNetworks, Wire, Wiretap, WiretapCentral, XSI.

Lightworks, the Lightworks logo, LWA and LWA-Enabled are registered trademarks of Lightwork Design Ltd. The LWA-Enabled logo, Interactive Image Regeneration, IIR, A-Cubed, Feature-Following Anti-Aliasing and FFAA are all trademarks of Lightwork Design Ltd. All other trademarks, images and logos remain the property of their respective owners. Copyright of Lightwork Design Ltd. 1990-2007, 2008-2012.

This software is based in part on the work of the Independent JPEG Group.

Disclaimer

THIS PUBLICATION AND THE INFORMATION CONTAINED HEREIN IS MADE AVAILABLE BY AUTODESK, INC. "AS IS." AUTODESK, INC. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE REGARDING THESE MATERIALS.

Contents

Introduction to the COM API in Navisworks

1

The Navisworks COM (Component Object Model) Interface allows Navisworks users a range of possibilities for customizing and extending its functionality.

The Component Object Model (COM) is a methodology that allows one discrete piece of software to communicate data to another component via an application programming interface (API).

This means that:

Components can be written in most languages that support COM, e.g. C, C++, Visual Basic, Visual Basic Script (VBS), JAVA, Delphi.

Components can be modified without directly affecting each other (they are independent pieces of software).

Where does the COM interface fit into Navisworks?

The COM interface's relative simplicity and the ability to use many programming languages mean that it is aimed at a broader base of end users. Non expert C++ users can code and implement simple changes to the interface themselves.

Users can write relatively simple scripts to achieve many aims, for example;

1. Link objects to Excel spreadsheet or Access database from Navisworks and pull up information in the object property area.
2. Construction sequencing link to MS Project: link selection set as timeline in Project and attach a hide/remove override to them. Then be able to take several viewpoints of these stages and stick them together as an animation.
3. Open Navisworks at a set viewpoint/animation (and start animation).
4. Set up AutoCAD viewpoint from within AutoCAD, based on a Navisworks view.
5. Report on clashes - scan the clash results and be able to write an html file of the clash information returned, including a bitmap of the clash viewpoint.
6. Container application with a Navisworks ActiveX control inside, scanning objects in the partition and filtering based upon some query (such as name, object info etc.).

What comprises the COM interface?

There are three main parts to the full COM interface product.

1. The COM interface itself – which allows users to write software plug-ins for Navisworks.
2. COM automation – conceptually 'super macro's' that enables users to control Navisworks programmatically e.g. an application can start up Navisworks and open a specific file with a specific viewpoint.
3. An ActiveX control – which allows Navisworks functionality to be embedded inside ActiveX 'containers' e.g. PowerPoint, IE, Word. An ActiveX control could be used to run Navisworks in a web browser and jump to pre-defined viewpoints using VBScript.

Requirements

For development, Visual Basic 6.0 is needed to get the most out of the COM API. However, once developed, the code can be distributed to other computers as Dynamic Link Library (DLL) or executable (EXE) files.

Navisworks Concepts

2

Introduction

Navisworks is a technology primarily concerned with supporting the interactive display of large models.

We define large models as those which take so long to display that interactive frame rates cannot be achieved. Whether a frame rate is interactive is very much dependent on the user. A good rule of thumb is that anything less than 10 frames a second is unacceptable for a novice user. Experienced users may be able to cope with frame rates as low as 1 or 2 frames a second.

Whether a model is large or not depends on the hardware available to display it. For a typical PC, models start to become large at around 10,000 triangles. For a PC with a good hardware accelerator, models start to become large at around 100,000 triangles. Indeed, one solution to the problem of displaying large models is to upgrade the available hardware.

Conventions in this section

Throughout this section italics are used to denote a concept that corresponds directly to a class of object within Navisworks. Keywords in bold denote properties or members of objects.

The Traditional Approach

If the hardware cannot be improved, the only solution is to display less of the model. The traditional approach involves an authoring process that modifies the model. The model is enhanced to include simplified versions of model parts (commonly called “level of detail”, or LODs). At runtime a decision is made to either display a simplified version of each part, or to use the original part. The decision is usually based on static rules (distance from viewer), sometimes with feedback (if last frame was too low reduce distance at which simpler part is used), sometimes with prediction (try to work out in advance how long displaying will take and calculate best distance).

The authoring process is tricky. Toolkits are available which will automatically generate a simplified version of a part. However, a decision is still needed as to what parts to simplify, to what extent and how many simplified versions of a part to include. Typically there is an author/test cycle until the model achieves adequate performance on the available hardware. There is a need to re-author if the model changes or is moved to less capable hardware.

Frame rates are still model and hardware dependent and there is a minimum authored complexity below which the model cannot drop. In addition, this approach requires the entire model to be examined for every frame displayed.

The authoring process takes time. It also requires more memory to store the multiple versions of each part.

How is Navisworks different?

Navisworks takes a different approach. Traversal of the model is ordered so that the most important parts of the model are displayed first. Guaranteed minimum frame rates are easily achieved by stopping when out of time. The prioritized traversal ensures that whenever dynamic displaying stops, the best image possible is available.

The viewing system drops detail while the user is interacting with the model. As soon as the interaction is over, the system starts to fill in the missing detail in order of importance. To the user it looks a little like a photograph developing. The user can start interacting again at any time, as soon as they've seen enough detail.

Navisworks works automatically from the original model data. No authoring step is required. No LODs are needed. Navisworks automatically scales with the model size and available hardware. The same model can be used on a graphics workstation and a laptop PC.

Architecture

Navisworks represents a model using separate logical and spatial data structures. The logical data structure corresponds to how a user would think about a model. So, an AutoCAD model would be divided up into layers, with inserts of blocks providing a hierarchical structure. A MicroStation model is divided into levels with elements grouped into cells providing a hierarchical structure.

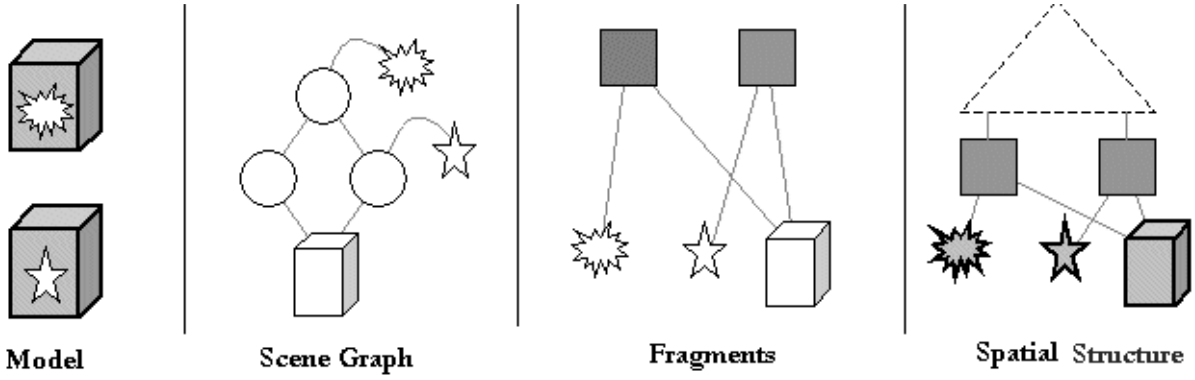
The logical structure of the model is usually not the best arrangement for fast displaying. So, Navisworks maintains a separate spatially organized data structure for optimal display performance. The spatial data structure is automatically rebuilt as the logical structure changes.

The Navisworks logical structure is a scene graph. A scene graph describes hierarchically structured data with arbitrary multiple instancing. Topologically a scene graph is a directed acyclic graph (DAG). Navisworks supports scene graphs with Attributes (Materials, Transforms, etc.) attached to Nodes in the graph. The leaf nodes of the graph represent Geometry. The combined set of attributes that apply to each geometric object is inherited from parent nodes in the graph.

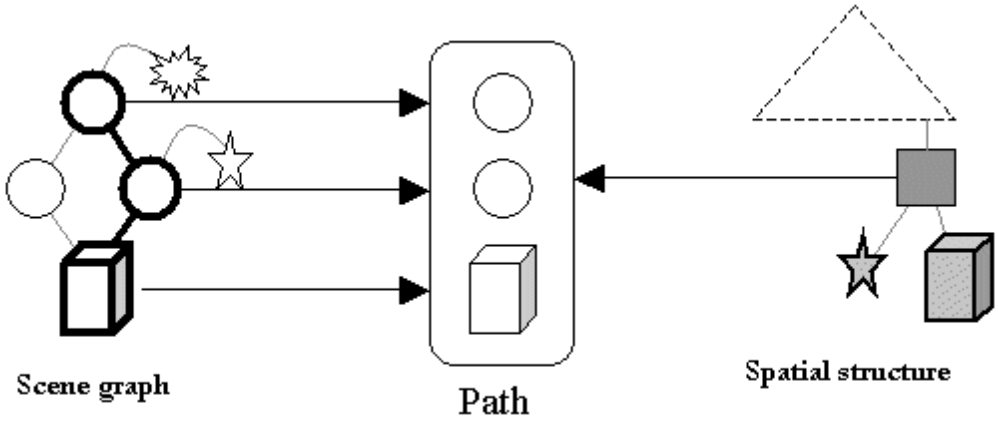
The logical structure of the source model is converted to the equivalent Navisworks scene graph. Each SceneBase (that is, Node or Attribute) has a name and type defined during the conversion of the model. This allows the logical structure in Navisworks to match that of the source CAD system as closely as possible. Two strings define the Type of a SceneBase. The first is ClassUserName, which is the string

displayed to the Navisworks user to describe the type. This string may be localized and may vary between different versions of Navisworks. It should not be used to identify any particular type of SceneBase when programming. The second is ClassName, which is never shown to the user, and is the same in all versions of Navisworks. This is safe to use to identify a particular type of SceneBase.

During conversion to the spatial data structure, the hierarchical scene graph of the logical structure is flattened, while still preserving instancing of geometry. Each instance of a Geometry node is represented by one or more Fragments in the spatial data structure. Each Fragment contains epitomized versions of the Geometry, Transform and Appearance attributes from the original scene graph. To save memory the original geometric data is discarded.



In a scene graph with arbitrary instancing the only way to identify a specific instance is by the Path through the scene graph from the root node (Partition) to the Geometry node. A Path can be thought of as a collection of Nodes where each Node is the child of the previous node in the collection. Paths are used to maintain the links between the logical and spatial data structures. Each Node contains a list of all Paths in which it is used. Each Fragment maintains a reference to the Path which it represents. Paths are used throughout Navisworks to identify objects within the model. For instance, a Selection is a collection of paths, as shown below.



Data Model

A complete Navisworks model is accessed via a State object.

State

A State corresponds to a document in Windows terminology. The State contains:

- A Partition forming the root of the model scene graph. If multiple files have been appended together the root partition will have child partitions, with each child representing one of the loaded files.
- The current view (AnonView).
- The current plan view (AnonView).
- The current section view (AnonView).
- The current Animation.
- The current Selection.
- A collection of SavedViews.
- A collection of scene Lights.
- A collection of SelectionSets
- Various other scene properties.
- A collection of Plug-ins, which extends and customizes Navisworks.

Partition

The root Node of a scene graph which is created by conversion of a single source file or model. A partition is used for the root of the current Navisworks model, for each file appended into the model and for external reference files (xrefs) referred to by the explicitly loaded files. The original source Filename (if available) is stored in the partition.

Node

The base class for the different types of node in the scene graph. There are three types of node:

- Partition: Root of a scene graph
- Group: Grouping and instancing node. Use IsLayer and IsInstance to determine intended interpretation.
- Geometry: Leaf of a scene graph. Represents a geometric object.

Attribute

Each Node can have an arbitrary number of attributes attached to it. Attributes are used to control the appearance of a Node or to just associate additional information with a node.

All appearance attributes apart from Transform are inherited from parent Nodes with the attribute furthest down the path being used. Appearance attributes are:

- Material: Color, transparency, shininess. If the node that the material is attached to has IsOverrideMaterial set, then the material will be used in preference to any further down the path.
- SemanticPriority: displaying priority. Objects with a higher priority are less likely to be dropped as details.
- Transform: Transform that positions object. Transforms accumulate along a path.

Information attributes are:

- NameAttribute: An arbitrary name.
- Nat64Attribute: A 64 bit unsigned integer.
- PropertyAttribute: A list of Property values.
- TextAttribute: A named block of text.
- URLAttribute: A named set of URLs linking to external data.

Property

A property is a named value of a particular type. The name is in two parts. A ‘UserName’, for display to the end user, and a fixed Name for use in programming. The value can be one of five types:

- Double precision floating point
- 32 bit integer

- Boolean
- Wide string
- Date/Time

Path

A Path is the sequence of nodes from the Partition at the root of the scene graph to a particular node within it that uniquely specifies a particular instance.

Selection and SelectionSet

A Selection is a set of paths that specify a subset of the complete scene graph. A SelectionSet is a named Selection.

Fragment

A Fragment is the representation of (possibly part of) a particular instance of a Geometry node within the scene graph. Large Geometry may be broken into multiple Fragments for more efficient displaying. Each Fragment provides access to:

- The complete Transform3f from local to world space.
- The Path that the fragment represents.
- The geometry used to display the fragment.
- The appearance used to display the fragment.

AnonView

An AnonView contains the complete specification of a view. It consists of:

- A ViewPoint
- A collection of section planes (Plane3f)

ViewPoint

A ViewPoint defines a camera position and properties which control how the viewpoint is displayed and modified. A ViewPoint includes:

- Camera: Defines what you're looking at.
- WorldUp vector: The preferred up direction when walking.

- FocalDistance: Distance from camera to center of attention when examining or orbiting.
- LinearSpeed: Defines speed of movement forwards and backwards.
- AngularSpeed: Defines speed of movement when rotating.
- Paradigm: Defines navigation mode (Walk, Orbit, Examine, etc.)
- Lighting: Off, Headlight or Scene Lights.
- Display Style: Full, Shaded, Wireframe or Hidden Line.

Camera

A 3D mathematical model of a camera. Defines what you're looking at. Includes:

- Position: Vec3f
- Orientation: Rotation3f
- Projection: Orthographic or Perspective
- HeightField: Height if Orthographic, Vertical field of view if Perspective.

SavedView

The base class for the different types of saved view within a model. There are four types of SavedView:

- View: A named AnonView.
- Folder: A collection of SavedViews.
- Animation: An animation sequence.
- Cut: A break in an animation sequence (jump cut).

Animation

Animations are represented as sequences of Views and Cuts. Each View acts as a key frame in the animation. When the animation is played, Navisworks interpolates between key frames. Cuts provide an optional pause and then jump to the next key frame. An animation has two properties:

- IsLoop: Does the animation loop indefinitely when played?
- Smoothing: What sort of interpolation should the animation use?

Linear Algebra (XXX3f)

Navisworks uses several objects to represent concepts in linear algebra.

- Vec3f: An arbitrary vector in 3D space. Specified a direction and length in 3D space. Represented by 3 floating point values.
- UnitVec3f: A direction in 3D space. Represented by a unit length vector.
- Pos3f. A position in 3D space. Represented by 3 floating point values.
- Rotation3f: A rotation in 3D space. Represented by an Axis UnitVec3f and an Angle in radians. Stored internally as a Quaternion.
- Plane3f: A plane in 3D space. Represented by a Normal UnitVec3f and a Distance from the origin.
- Box3f: An axis aligned bounding box. Represented by Min and Max Pos3fs.
- Transform3f: An arbitrary 3D transformation. Represented by a 4x4 matrix. All transforms should be affine.

Plugin

Navisworks supports plug-in objects that extend basic functionality. There are four types of plug-in:

- Export: Specifies a new command to be added to the Navisworks export, tools or context menus.
- Property: Modifies the way Attributes are displayed and searched in the Properties and Find dialogs.
- Ignore: Adds extra rules to the Navisworks ignore options.
- Presenter: Adds extra rules to the Presenter tab. These are used to apply textures to the model.

COM Technical Notes

3

Overview

COM support has been added to the Navisworks product in the following areas.

- Navisworks ActiveX controls.
- Navisworks Automation.
- COM Plug-ins for Navisworks.
- COM 'State' API for Navisworks and ActiveX controls.

Each of these areas is discussed in their separate sections.

Requirements

For development, Visual Basic 6.0 is needed to get the most out of the COM API. However, once developed, the code can be distributed to other computers as Dynamic Link Library (DLL) or executable (EXE) files.

Navisworks ActiveX Controls

4

Overview

These controls allow Navisworks style navigation of files embedded within hosts such as web pages of end user applications. . The aim is to allow users to embed Navisworks capabilities inside Web Pages, applications and other containers. The user can also access and manipulate the controls internal state. The controls can open files using absolute, URL or relative paths. File download is done in a separate thread to prevent 'hanging' the containing program.

ActiveX DLLs

Two Main types of ActiveX DLL are available:

'Navisworks Redistributable API Library 14' lcodieDX.dll

- This control is supplied in a Windows Installer merge module and also a self-installing executable.
- For control and internal state access.
- Only Navisworks files may be opened.
- 32bit and 64bit versions available.
- Only certain methods in the API will works as the control runs in ‘unlicensed’ mode unless a special runtime license is embedded in the container application.

'Navisworks Integrated API Library 14' lcodieD.dll

- Built in to the Navisworks main product install.
- For control and internal state access.
- 32bit only on 32bit OS, 64bit only on 64bit OS.

- The control can open various none Navisworks CAD file formats.
- Whole API is available if the main product is licensed.
- Note: Microsoft Visual Studio itself is a 32bit product, even on 64bit OS. This causes problems trying to develop applications as it cannot load the 64bit Navisworks Integrated ActiveX controls.
    - To get round this we have installed a dummy 32bit control in C:\Program files (x86)\Autodesk\Navisworks Manage 2017". You must manually register this using "regsvr32.exe lcodied.dll". This control is only for development purposes and won't be required once you have built your application. The supplied ActiveX samples will not compile unless this registration has been performed.

MDI / SDI Support

For each ActiveX DLL, two controls are included:

MDI: Multiple Document Support

- Multiple controls can be used in a process.
- Plugins are not supported. Thus textured materials and extra File readers are not available.

SDI: Single Document Support

- Only a single control can be used in a process.
- Plugins are supported. Thus textured materials and extra file readers may be supported.

HTML Usage

Use the 'OBJECT' Tag with the correct CLSID inside.

The Integrated ActiveX control is provided as a self-installing .exe file. This file allows the use of the 'CODEBASE' parameter for automatic control installation.

COM 'State' API for Navisworks and ActiveX

5

The Navisworks and ActiveX controls share a common API on their internal state. Although similar, some interfaces and co-classes may only appear in one dll.

Interface Names

Each object has only one relevant interface containing all methods. The interface name for a particular object is similar to the object name. E.g. object ‘nwOpState’ has interface ‘InwOpState’.

Top Level Object

The ‘nwOpState’ object is the ‘state’ top-level object. Below this is a hierarchy of other objects exposed via properties, methods and collections. The state is a powerful object. It provides COM events when various things happen in the GUI.

Object / Interface Derivation

Each object’s interface derives either directly or indirectly from ‘InwBase’. This contains standard methods. e.g. ‘InwOpState’ derives from ‘InwBase’. It thus contains all the ‘InwBase’ methods and properties.

Collections

For the purpose of the COM interface all indexes are 1 based. All collections have a standard set of methods but some may not be implemented for some collections. One collection may contain different object types. Normally these will all derive from one common object type.

Object Identification

InwBase contains a read only property ‘ ObjectName’. This can be used to identify unknown objects instead of having to do a ‘QueryInterface’ or Visual Basic ‘Set’ operation. This is the only way to identify objects when using untyped scripting languages.

New Object Creation

You cannot create Navisworks objects using ‘new’. Instead you must use the ‘ ObjectFactory’ method on the state.

Enums

In VBScript you cannot use the predefined Navisworks enum values. The state however has a method ‘GetEnum’ to return the value of an enum from its string form. This makes coding more readable.

Comparing Objects

If you want to know if two VB references refer to the same Navisworks object do not compare them by reference or use the VB ‘Is’ method. Instead compare the values using ‘State.PtrEquals’ method.

Navisworks Automation

6

Overview

Basic automation support has been added to Navisworks. The aim is to allow users to programmatically start up and manipulate the product using Visual Basic, VBA or VBScript.

The user can also access and manipulate the applications internal state.

Side-by-Side Installations

With Navisworks 2017 product releases, each different product (Manage and Simulate) can be installed side-by-side. For COM automation purposes, they are identical, and in fact the GUID for each product is the same.

When started via COM automation, the product that was last installed or last run is the one that will be started. So, for example, if you install Navisworks Manage 2017 and then install Navisworks Simulate 2017, then when the product is started via automation, Navisworks Simulate will start. But then if Navisworks Manage is subsequently run by the user, then it is this that will start next time the product is automated.

Navisworks Licensing

Currently automation will be available for licensed versions of Navisworks.

VBScript

ProgId = 'Navisworks.Document' for Navisworks.

VB Usage - References

'Navisworks Automation Library 14' - for Navisworks automation.

'Navisworks Integrated API Library 14' - for internal state access

'OpenFile' method

This method opens the required Navisworks files.

'State' property

Gets access to the controls internal state.

'Visible' property

Controls the application / document visibility.

'StayOpen' property

Controls if the application / document will die when all COM references are released.

Quit

Closes application.

COM Plug-ins for Navisworks

7

Overview

COM plug-ins can now be written for Navisworks. This allows users to extend the standard GUI functionality using Visual Basic. Currently ‘Property’, ‘Presenter’, ‘Export’ and ‘Navisworks’ plug-ins can be written. Once loaded these plug-ins become part of the standard GUI and the ‘Tools\Options’ tabbed dialog will provide an option page for each plug-in.

Navisworks Licensing

Currently plug-ins can be written for all licensed versions of Navisworks.

VB Usage - References

‘NavisworksAPI’ - for internal state access

Interfaces

InwPlugin, InwExportPlugin, InwClashPlugin, InwPropertyPlugin

All Plug-ins

All plug-ins must implement the InwPlugin Interface. This does basic work like providing the plug-in name and serializing any user definable properties to and from Navisworks. Plug-ins can be built as Dll’s or Exe’s. Executables have advantages that they can display modeless forms.

During initialization you supply a set of flags to the plug-in. This affects the overall behavior.

An options page may be added each plug-in. This can display custom properties of Variant types CLng, CDbl, CBool, CDate and CStr.

Export Plug-ins

Export plug-ins must implement the InwExportPlugin interface. Export plug-ins allow for a complete custom export of data by adding to the Navisworks menus. The main method of note is ‘iExport(…)’, this

will be called to actually do the export. The plug-in author can use the application's state and the full power of VB to do whatever they want here. Flags allow for menu items to be added to the Context, Export and Tools menus.

It is also possible to have no added menu items. Instead you may just be using nwOpState's Events to look for things happening.

Clash Detective Plug-ins

Clash ignore plug-ins must implement the LnwClashPlugin interface. COM Interface plug-ins allow specific clashes to be ignored. As each clash is found the 'iIgnore(…)' method of the plug-in is called.

Property Plug-ins

Property plug-ins must implement the InwPropertyPlugin interface. Property plug-ins allow overriding of what is displayed when a user right clicks on an object. The plug-in can supply a block of text or list of properties to be displayed.

Presenter Plug-ins

Presenter Plug-ins must implement the InwPresenterPlugin interface.

Presenter plug-ins are used to apply textures to the model. The loaded Presenter plug-ins are listed on the Presenters 'Rules' Tab.

Loading Plug-ins

Currently the only way to make Navisworks load COM plug-ins is to add their 'ProgId' string to the registry. No GUI is supplied to do this. Navisworks must be restarted to load the plug-ins.

Because each Navisworks 2017 product can be installed side-by-side, they each have a different set of registry entries. This means that you must register your plug-in with each version of the product that it is compatible with. The registry keys for each plug-in are shown in the table below.

Product	Registry Key
Navisworks Manage 2017	Software\Autodesk\Navisworks Manage\14.0\COM Plugins
Navisworks Simulate 2017	Software\Autodesk\Navisworks Simulate\14.0\COM Plugins

Add string values corresponding to the version independent progIds for your plug-in to the product-specific registry key.

Optionally a version-specific progId may be given as the string value. For example:

Name	Value
mycustom.mypluginA	mycustom.mypluginA.1
mycustom.mypluginB	mycustom.myPluginB

Microsoft .Net support in Navisworks

8

Garbage collection issues

The COM API originates before the advent of .NET languages and their garbage collection design. Thus it is important to ensure that all related properties are kept from being disposed by the Garbage Collector until the process being worked on is complete. For example, let us look at the following code excerpt.

```
foreach (InwOaPath path in sel.Paths())
{
    InwGUIPropertyNode2 propertyNode;
    List<InwOaProperty> properties = new List<InwOaProperty>();
    propertyNode = (InwGUIPropertyNode2)nwOpState.GetGUIPropertyNode(path, true);
    foreach (NavisWorksAPI6.InwGUIAttribute2 attribute in propertyNode.GUIAttributes())
    {
        try
        {
            string classUsername = attribute.ClassUserName;
            if (classUsername.Equals(attributeName))
            {
                foreach (InwOaProperty p in attribute.Properties())
                {
                    InwOaProperty property =
(InwOaProperty)nwOpState.ObjectFactory(nwEObjectType.eObjectType_nwOaProperty, null, null);
                    property.UserName = p.UserName;
                    property.value = p.value;
                    properties.Add(property);
                }
                break;
            }
        }
        catch (Exception e)
        {
            System.Windows.Forms.MessageBox.Show(e.Message+e.StackTrace);
        }
    }
}
```

In this example, it could be assumed that everything would work as expected. Indeed, debugging this section of code, where the Garbage collector is less optimized, all could work as expected. However, 'propertyNode' is last used with the call to GUIAttributes(). Once this call has completed, the .NET Garbage Collector would be allowed to delete propertyNode as it is no longer used. In the case of the InwGUIPropertyNode2 class, the removal of propertyNode also deletes the attributes, because of the underlying functionality of the COM API, making calls such as 'attribute.Properties()' throw exceptions.

This problem is not uncommon with .NET programs using COM components and there are a number of solutions to the problem. The simplest solution here is to call GC.KeepAlive(propertyNode) after the loop on GUIAttributes(). KeepAlive, a function of the Garbage Collector class (GC), lets the Garbage

Collector know that propertyNode is ineligable for Garbage Collection from the point of declaration to the point where KeepAlive is called.

The fixed code could therefor look like this:

```
foreach (InwOaPath path in sel.Paths())
{
    InwGUIPropertyNode2 propertyNode;
    List<InwOaProperty> properties = new List<InwOaProperty>();
    propertyNode = (InwGUIPropertyNode2)nwOpState.GetGUIPropertyNode(path, true);
    foreach (NavisWorksAPI6.InwGUIAttribute2 attribute in propertyNode.GUIAttributes())
    {
        try
        {
            string classUsername = attribute.ClassUserName;
            if (classUsername.Equals(attributeName))
            {
                foreach (InwOaProperty p in attribute.Properties())
                {
                    InwOaProperty property =
(InwOaProperty)nwOpState.ObjectFactory(nwEObjectType.eObjectType_nwOaProperty, null, null);
                    property.UserName = p.UserName;
                    property.value = p.value;
                    properties.Add(property);
                }
                break;
            }
        }
        catch (Exception e)
        {
            System.Windows.Forms.MessageBox.Show(e.Message+e.StackTrace);
        }
    }
    GC.KeepAlive(propertyNode); //Call here to keep propertyNode 'alive' through the inner loop
}
```

Known compatibility problems and workarounds

Most Navisworks API interfaces have an .Xtension method to allow extension. Flags may be set using this method to provide VB.Net compatibility. Note: These compatibility flags are for development and may not be supported in future releases.

API: Error handling: Invalid descriptions

Exceptions returned from interfaces may have invalid / misleading descriptions. The actual error number however should be correct. (After subtracting 'vbObjectError' the error should be in the range 100 to 114).

API: Returned Safe Arrays are 1 based and not supported by VB.Net

Certain methods (such as path.ArrayData) return 1 based Safe Arrays. This can cause issues as .Net only really supports 0 based arrays.

With .NET 4.0 dynamics you may see exceptions similar to:

```
Unable to cast object of type 'System.Double[*]' to type 'System.Double[]'.* exception:
```

You can get around this by appropriate casting, and then using the ‘GetValue’ accessor:

```
ComApi.InwOaPath oPath = state.CurrentSelection.Paths()[1];
ComApi.InwOaFragment frag = oPath.Fragments()[1];
ComApi.InwLTransform3f transform = frag.GetLocalToWorldMatrix();

Array a=(Array)(object)transform.Matrix;
float f = (float)(double)(a.GetValue(1));
```

There is also a method that will force all subsequently returned arrays to be 0 based for the whole API. However this should only be called as a last resort, and then only within your own ActiveX based project. Calling in the main products may cause side effects in other areas.

```
AxnwControl1.Xtension('Dev_DotNet_ArrayBase0')
or:-
state.Xtension('Dev_DotNet_ArrayBase0').
```

ActiveX Control: 'State' is a reserved object name

Accessing the controls state is difficult as 'State' is a reserved object name in the VB.Net wrapper around an ActiveX control. It is possible to call the Navisworks State method via automation. A new method APIState has been added to InwControl2 to avoid this clash. Alternately use the following code to get the state:-

```
Dim state as InwOpState
state=AxnwControl1.Xtension('Dev_DotNet_State')
OR
state=AxnwControl1.APIState()
```

ActiveX Control: Setting SRC causes an exception

The SRC property may not work properly with VB.Net. This is due to internal problems involving relative path calculation. To get around this before setting the SRC property call:-

```
AxnwControl1.Xtension('Dev_DotNet_AbsoluteSRC')
```

Alternately prefix the filename passed to SRC with a '@'. Note: Relative paths will not be supported in these cases.

Navisworks Plug-ins: .Net plug-ins not supported

Navisworks can only call COM plug-ins.Net plug-ins are not supported. It is however possible to get a .Net Class library to behave as a COM plug-in. The implementation should begin something like:-

```
<ComClass(Class1.MyClassId, Class1.MyInterfaceId, Class1.MyEventsId)>
_
Public Class Class1
Implements NavisworksAPI.InwPlugin
Implements NavisworksAPI.InwExportPlugin
#Region "GUIDs"
Friend Const MyClassId As String = "E569C0D2-2839-XXc2-98D6-4170F6A84A0F"
Friend Const MyInterfaceId As String = "A8AEA57C-0389-XX06-BCD0-E394327E881C"
Friend Const MyEventsId As String = "75E5292E-BE47-XXac-962E-6AD9DD0AEDD0"
```

```
#End Region
```

The project 'Configuration Properties\Build' settings must have the option checked for 'Register for COM Interop'.