Exposing Hidden Functionality for AutoCAD .NET Application Development

James E. Johnson

Sr. Application Developer

Twitter: @jamescad

Class summary

This class is an introduction for AutoCAD .NET API software developers to learn how to extend the managed ObjectARX libraries to use hidden functionality. In this class we show ways to find hidden functionality in AutoCAD software files and ObjectARX libraries. The class will show how add that functionality to managed .NET applications.

Key learning objectives

At the end of this class, you will be able to:

- Learn how to extend the AutoCAD .NET managed capability.
- Discover exported functions in AutoCAD files that can be imported to managed code.
- Learn how to get the structure of the exported functions and import them into managed code.
- Discover Platform Invoke (P/Invoke).

Introduction

Introduction

- ObjectARX® programming environment libraries provides functionality for AutoCAD® application programming.
- ObjectARX® provides direct access to the AutoCAD® database, graphics system, and native command definition.
- Some native capabilities are not exposed with .NET ObjectARX® but can be exposed using Platform Invoke (P/Invoke).

Learn how to extend AutoCAD® .NET managed capability

Learn how to extend AutoCAD® .NET managed capability.

- Classes and methods
- Extension methods
- Referenced DLL’s
- Platform Invoke (P/Invoke)

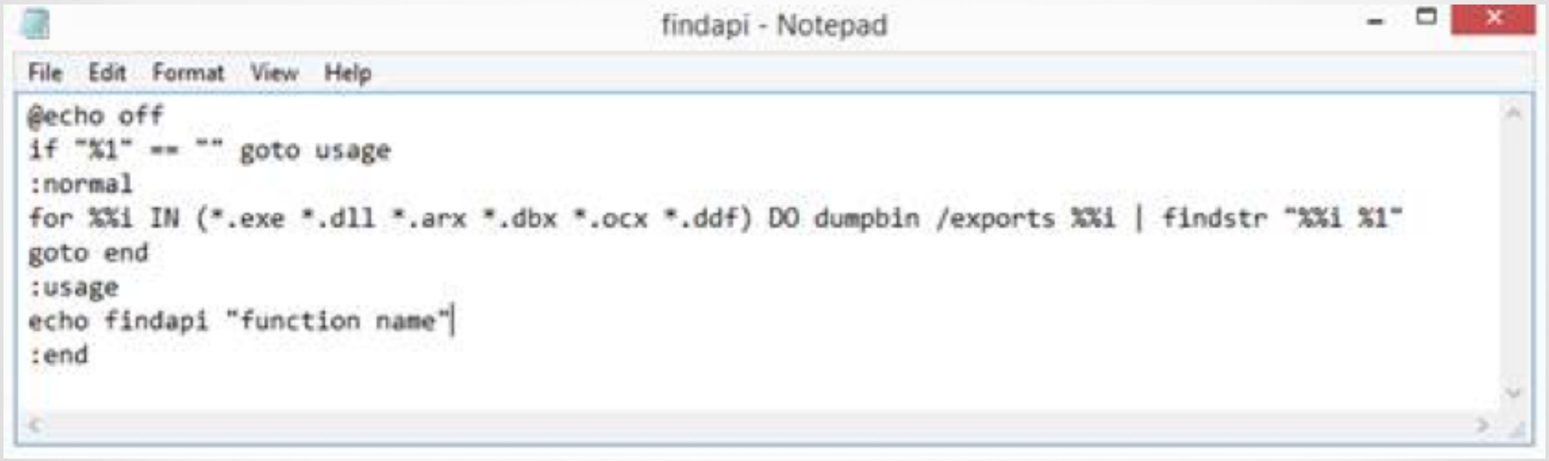
Learn how to extend AutoCAD® .NET managed capability.

Code examples…

Discover exported functions in AutoCAD® files that can be imported to managed code.

Discover exported functions in AutoCAD® files that can be imported to managed code.

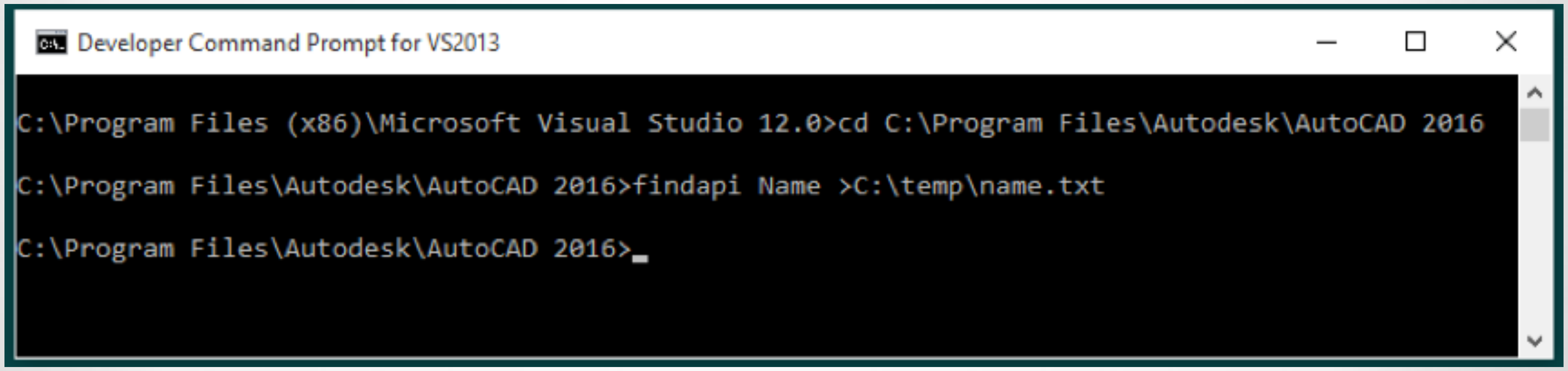
Create a .bat file using “dumpbin.exe” :



```
@echo off
if "%1" == "" goto usage
:normal
for %%i IN (*.exe *.dll *.arx *.dbx *.ocx *.ddf) DO dumpbin /exports %%i | findstr "%%i %1"
goto end
:usage
echo findapi "function name"
:end
```

Discover exported functions in AutoCAD® files that can be imported to managed code.

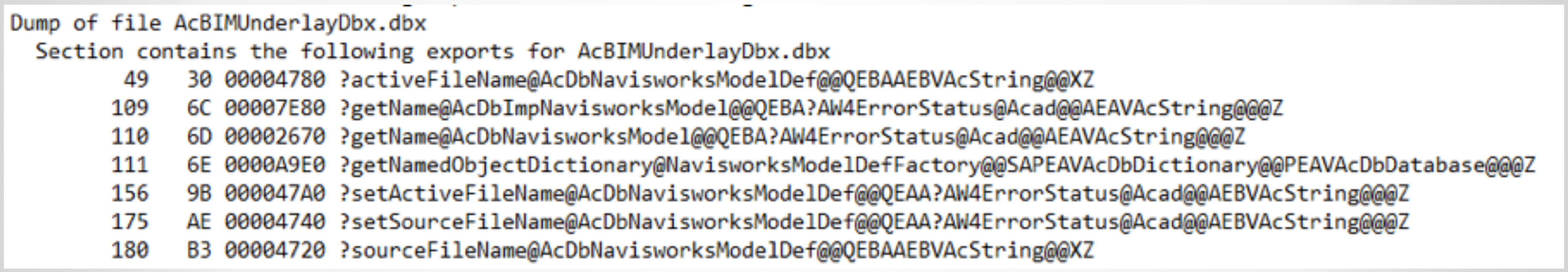
Copied the above findapi.bat to the AutoCAD folder and ran in a command window :



```
C:\Program Files (x86)\Microsoft Visual Studio 12.0>cd C:\Program Files\Autodesk\AutoCAD 2016

C:\Program Files\Autodesk\AutoCAD 2016>findapi Name >C:\temp\name.txt

C:\Program Files\Autodesk\AutoCAD 2016>_
```

Discover exported functions in AutoCAD® files that can be imported to managed code.

In the Name.txt file created from running findapi.txt, found the following :



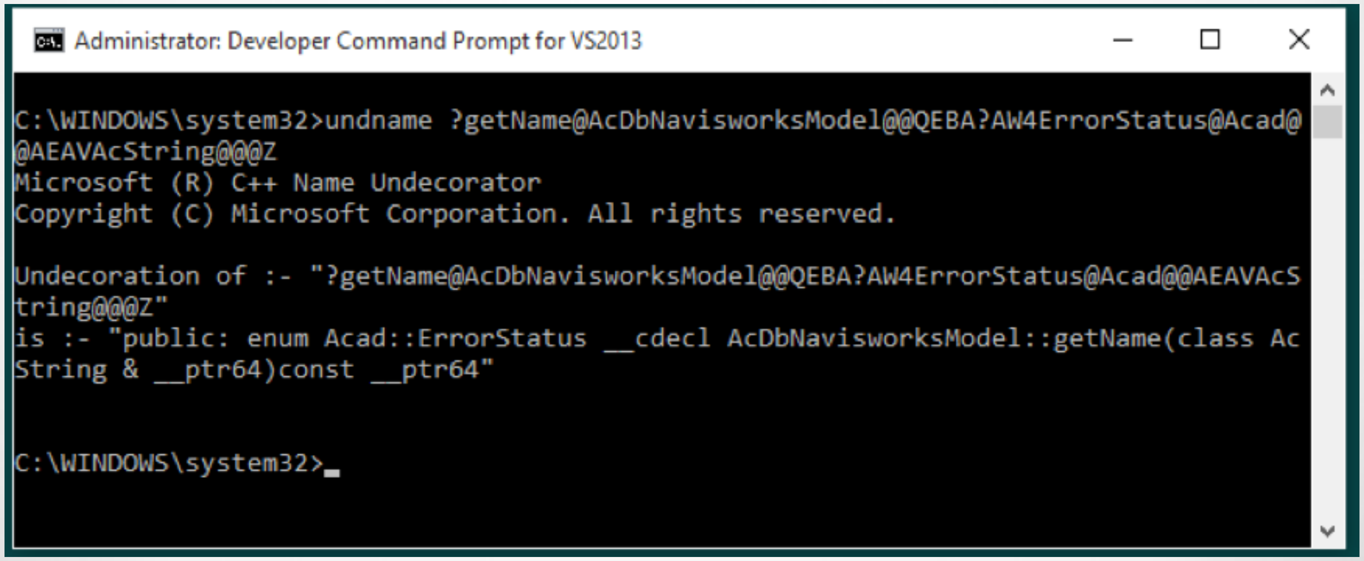
```
Dump of file AcBIMUnderlayDbx.dbx
  Section contains the following exports for AcBIMUnderlayDbx.dbx
         49    30 00004780 ?activeFileName@AcDbNavisworksModelDef@@QEBAAEBVAcString@@XZ
        109    6C 00007E80 ?getName@AcDbImpNavisworksModel@@QEBA?AW4ErrorStatus@Acad@@AEAVAcString@@@Z
        110    6D 00002670 ?getName@AcDbNavisworksModel@@QEBA?AW4ErrorStatus@Acad@@AEAVAcString@@@Z
        111    6E 0000A9E0 ?getNamedObjectDictionary@NavisworksModelDefFactory@@SAPEAVAcDbDictionary@@PEAVAcDbDatabase@@@Z
        156    9B 000047A0 ?setActiveFileName@AcDbNavisworksModelDef@@QEAA?AW4ErrorStatus@Acad@@AEBVAcString@@@Z
        175    AE 00004740 ?setSourceFileName@AcDbNavisworksModelDef@@QEAA?AW4ErrorStatus@Acad@@AEBVAcString@@@Z
        180    B3 00004720 ?sourceFileName@AcDbNavisworksModelDef@@QEBAAEBVAcString@@XZ
```

?getName@AcDbNavisworksModel@@QEBA?AW4ErrorStatus@Acad@@AEAVAcString@@@Z

Learn how to get the structure of the exported functions and import them into managed code.

Learn how to get the structure of the exported functions and import them into managed code.

- Then took the mangled string and used "**undname.exe**" to get the methods structure :



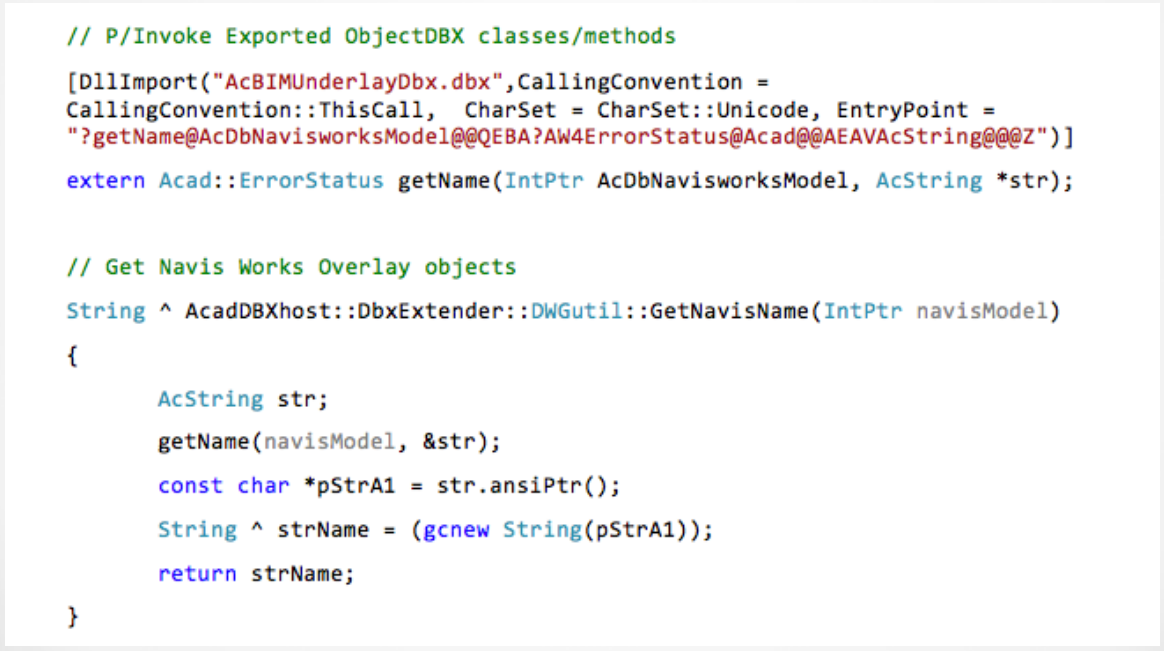
Administrator: Developer Command Prompt for VS2013

```
C:\WINDOWS\system32>undname ?getName@AcDbNavisworksModel@@QEBA?AW4ErrorStatus@Acad@
@AEAVAcString@@@Z
Microsoft (R) C++ Name Undecorator
Copyright (C) Microsoft Corporation. All rights reserved.

Undecoration of :- "?getName@AcDbNavisworksModel@@QEBA?AW4ErrorStatus@Acad@@AEAVAcS
tring@@@Z"
is :- "public: enum Acad::ErrorStatus __cdecl AcDbNavisworksModel::getName(class Ac
String & __ptr64)const __ptr64"


C:\WINDOWS\system32>_
```

Discover Platform Invoke (P/Invoke)

Discover Platform Invoke (P/Invoke)

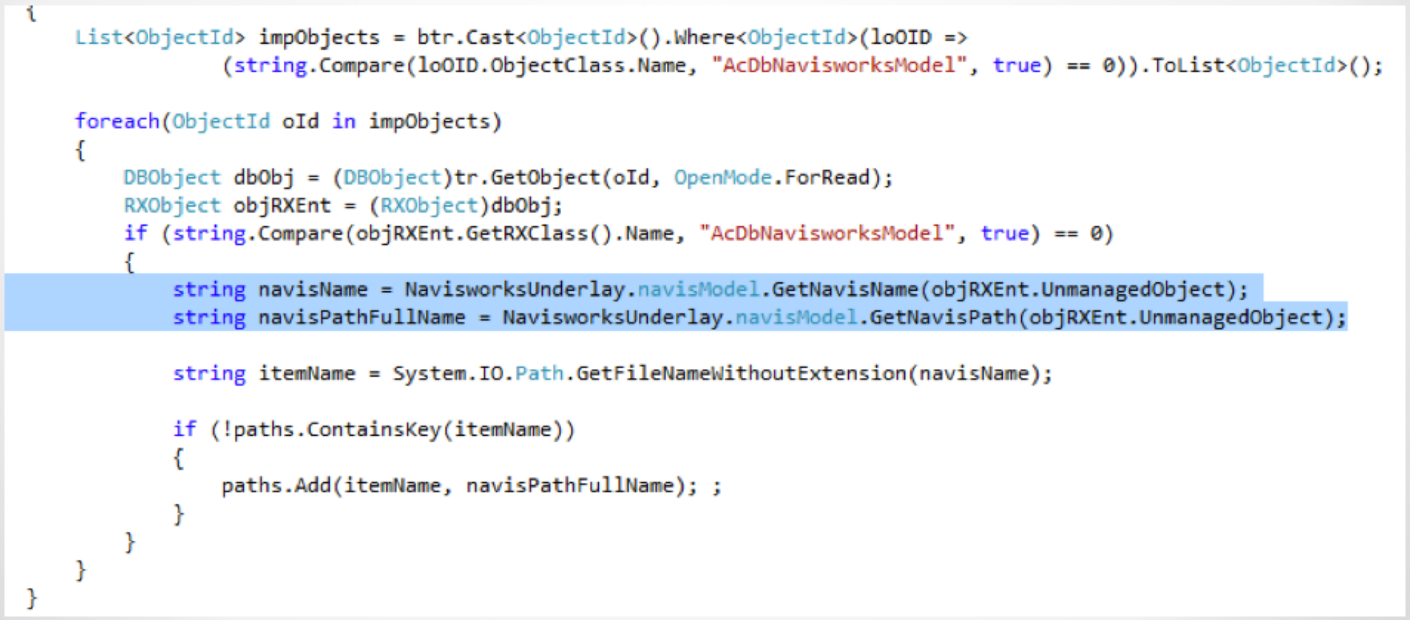
Then added it to the code to a C++/CLI ObjectARX project :



```
// P/Invoke Exported ObjectDBX classes/methods
[DllImport("AcBIMUnderlayDbx.dbx",CallingConvention =
CallingConvention::ThisCall,  CharSet = CharSet::Unicode, EntryPoint =
"?getName@AcDbNavisworksModel@@QEBA?AW4ErrorStatus@Acad@@AEAVAcString@@@Z")]
extern Acad::ErrorStatus getName(IntPtr AcDbNavisworksModel, AcString *str);


// Get Navis Works Overlay objects
String ^ AcadDBXhost::DbxExtender::DWGutil::GetNavisName(IntPtr navisModel)
{
        AcString str;
        getName(navisModel, &str);
        const char *pStrA1 = str.ansiPtr();
        String ^ strName = (gcnew String(pStrA1));
        return strName;
}
```

Discover Platform Invoke (P/Invoke)

Then called from a C# assembly that references the C++/CLI DLL :



```
{
    List<ObjectId> impObjects = btr.Cast<ObjectId>().Where<ObjectId>(loOID =>
            (string.Compare(loOID.ObjectClass.Name, "AcDbNavisworksModel", true) == 0)).ToList<ObjectId>();

    foreach(ObjectId oId in impObjects)
    {
        DBObject dbObj = (DBObject)tr.GetObject(oId, OpenMode.ForRead);
        RXObject objRXEnt = (RXObject)dbObj;
        if (string.Compare(objRXEnt.GetRXClass().Name, "AcDbNavisworksModel", true) == 0)
        {
            string navisName = NavisworksUnderlay.navisModel.GetNavisName(objRXEnt.UnmanagedObject);
            string navisPathFullName = NavisworksUnderlay.navisModel.GetNavisPath(objRXEnt.UnmanagedObject);

            string itemName = System.IO.Path.GetFileNameWithoutExtension(navisName);

            if (!paths.ContainsKey(itemName))
            {
                paths.Add(itemName, navisPathFullName); ;
            }
        }
    }
}
```

Discover Platform Invoke (P/Invoke)

- **Additional uses of Platform Invoke (P/Invoke)**
  - Import other ObjectARX functionality (many websites and forum threads exist on this topic).
  - Read write INI files
  - Import other application exported functions to make them available to .NET application.

Thanks for Attending…

Be heard! Provide AU session feedback.

- Via the Survey Stations, email or mobile device.
- AU 2016 passes awarded daily!
- Give your feedback after each session.
- Give instructors feedback in real-time.

Forget to take notes? No problem!

After AU visit:
AutodeskUniversity.com

Click on **My AU** to find:

- Class Recordings
- Presentations
- Handouts

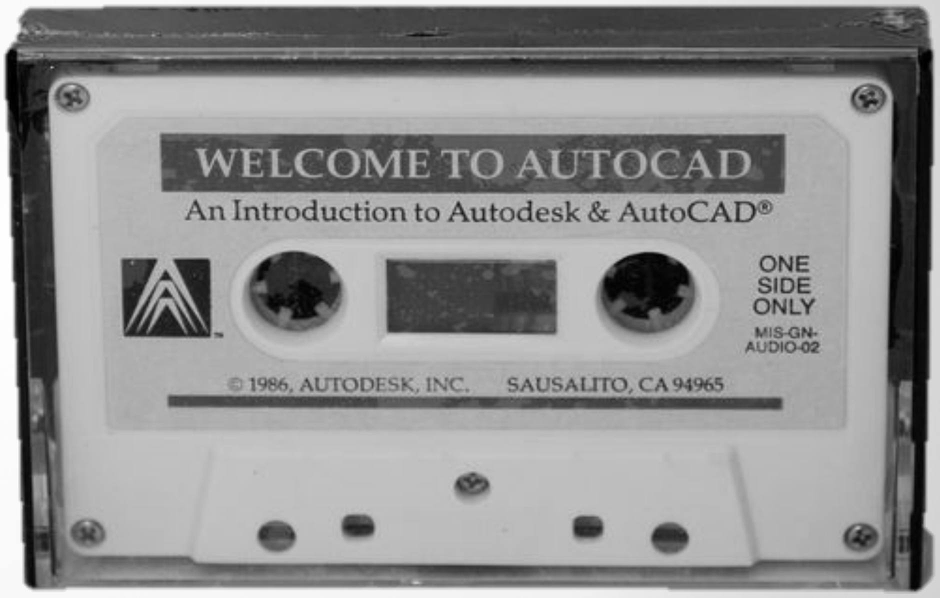
All of your sessions will be there to enjoy again and again.

Instruction Manuals Outdated?

Visit:

AutodeskUniversity.com

Click on **My AU** to start building your own desk reference (with materials from this decade).



Learn something worth sharing?

After AU visit:

AutodeskUniversity.com

Click on **My AU** to share your AU experience with:

- Colleagues
- Peers
- Professionals

Save hundreds of sessions worth sharing.

Too many sessions, too little time?

After AU visit:

AutodeskUniversity.com

- Recorded sessions
- Presentations and handouts
- Key learnings

Don’t miss a second! Find hundreds of sessions waiting for you.

More Questions? Visit the AU Answer Bar

- Seek answers to all of your technical product questions by visiting the **Answer Bar**.
- Open daily 8am-10am and Noon-6pm and located just outside of Hall C on Level 2.
- Staffed by Autodesk developers, QA, & support engineers ready to help you through your most challenging technical questions.

AUTODESK®