AS322865 - Managing CAD Standards with VB.NET

Lee Ambrosius

Principal Content Experience Designer

AUTODESK. UNIVERSITY

Who Am I?

My name is Lee Ambrosius:

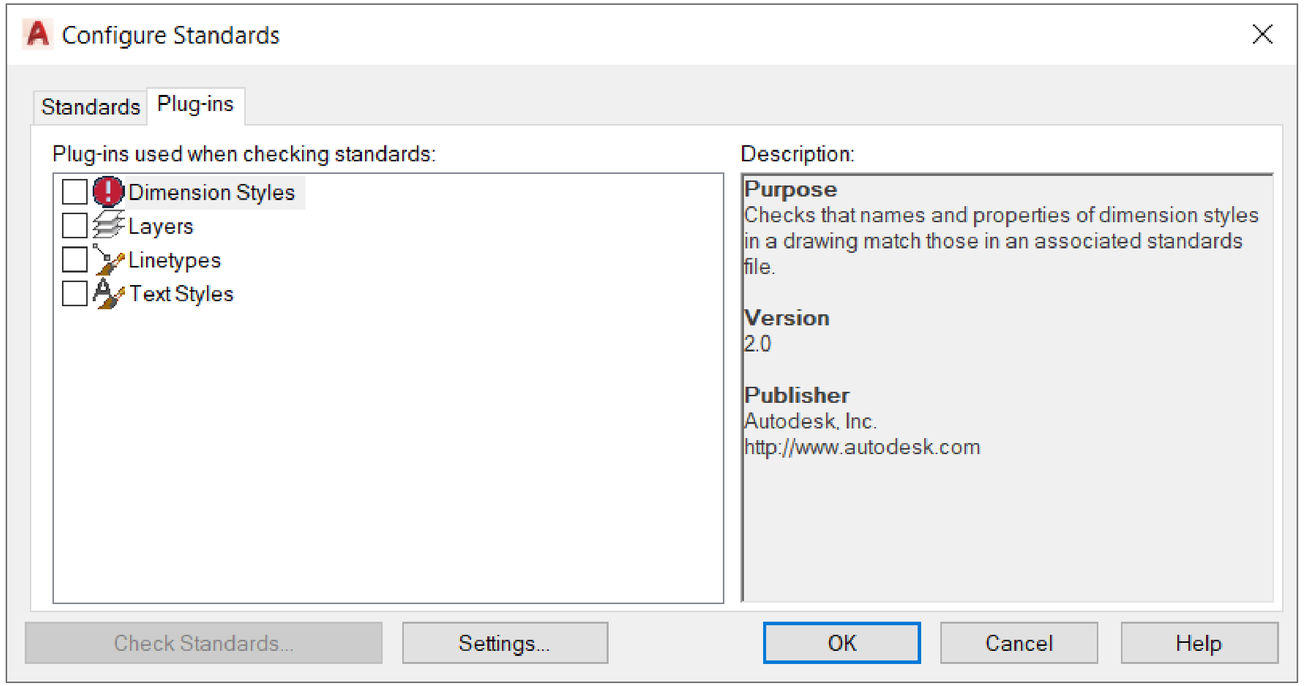
- Principal Learning Experience Designer at Autodesk, Inc.
  - Technical writer and data analyst
  - Customization, Developer, and CAD Administration documentation
- Over 20+ years of AutoCAD customization and programming experience
- Author of the AutoCAD Customization Platform book series published by Wiley & Sons

What is Being Covered

What is Being Covered

Extending the CAD Standards framework with a custom plug-in

- What is needed to get started
- Anatomy of a plug-in
- Define a plug-in
- Load and use a plug-in
- Debug a plug-in
- Look at a graphical plug-in



What is Being Covered

Extending the CAD Standards framework with a custom plug-in

Beyond the CAD Standards API

- Compare object properties (current drawing against DWS file)
- Set a layer current when a command starts/ends
- Import objects from a DWS file into the current drawing
- Parse a CHX file created by the Batch Standards Checker
  - Display property differences found
  - Create a SCR file to fix standards

What is Being Covered

Extending the CAD Standards framework with a custom plug-in

- What is needed to get started
- Anatomy of a plug-in
- Define a plug-in
- Load and use a plug-in
- Debug a plug-in
- Look at a graphical plug-in

Beyond the CAD Standards API

What is Needed to Get Started

What is Needed to Get Started

- AutoCAD 2020 (or earlier)
- Microsoft Visual Studio 2017 (Professional, Enterprise, or Community)
- ObjectARX Software Development Kit (SDK) for 2020 (or earlier)
- AutoCAD Developer Documentation
  - AutoCAD ActiveX Reference
  - CAD Standards Plug-in Reference
  - AutoCAD Managed .NET Developer’s Guide and Reference

What is Needed to Get Started

Interop assembly of the CAD Standards Type Library

1. Launch the Visual Studio 2017 Command Prompt.
2. Type cd "c:\ObjectARX <release>\inc-x64" and press Enter.
3. Type `tlbimp acstmgr.tlb /machine:Agnostic` and press Enter.

   The file *AcStMgr.dll* is created in the *inc-x64* folder; only needs to be done once.
4. Type `tlbimp axdb<version>enu.tlb /machine:Agnostic` and press Enter.

   The file *AXDBLib.dll* is created in the *inc-x64* folder; only needs to be done once.

Note: *cadStdsInterop.bat* in the additional files ZIP file shows how to automate this process.

What is Being Covered

Extending the CAD Standards framework with a custom plug-in

- What is needed to get started
- Anatomy of a plug-in
- Define a plug-in
- Load and use a plug-in
- Debug a plug-in
- Look at a graphical plug-in

Beyond the CAD Standards API

Anatomy of a Plug-in

Anatomy of a Plug-in

`IAcStPlugin2` is the main interface

Interface implements functions to:

- Identify a plug-in
- Initialization a plug-in within the CAD Standards framework
- Retrieve errors and fixes
- Iterate errors
- Apply a fix to an error
- Report errors

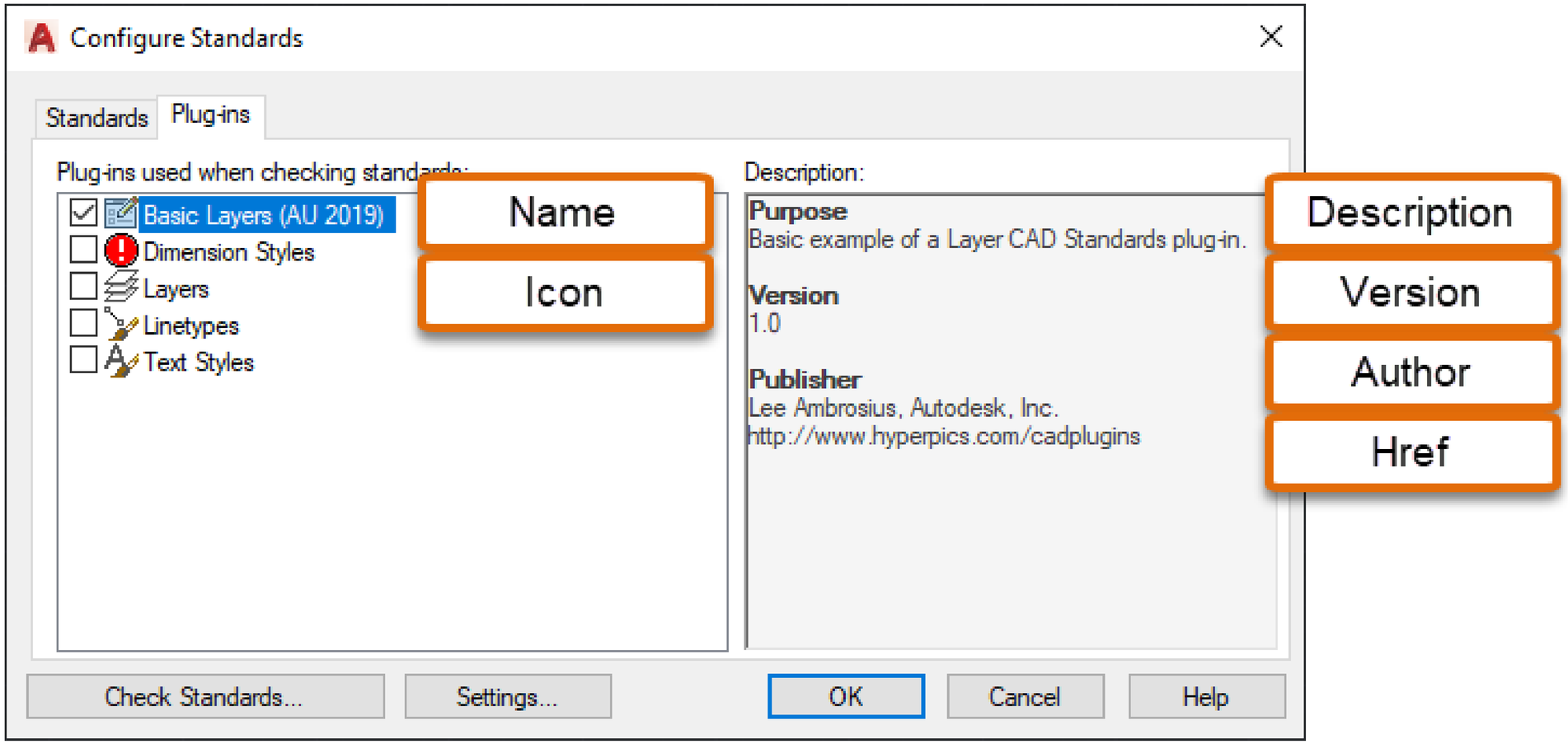
Identify a plug-in

Each plug-in must identify itself when being loaded

Identified by these properties:

- `Author`
- `Description`
- `HRef`
- `Icon`
- `Name`
- `Version`

Identify a plug-in



Initialization of a plug-in

After identify itself, the plug-in is initialized when:

- Opening of a drawing with an associated DWS file, and real-time checking is enabled
- Enabling a plug-in in the CAD Standards dialog or the Batch Standards Checker

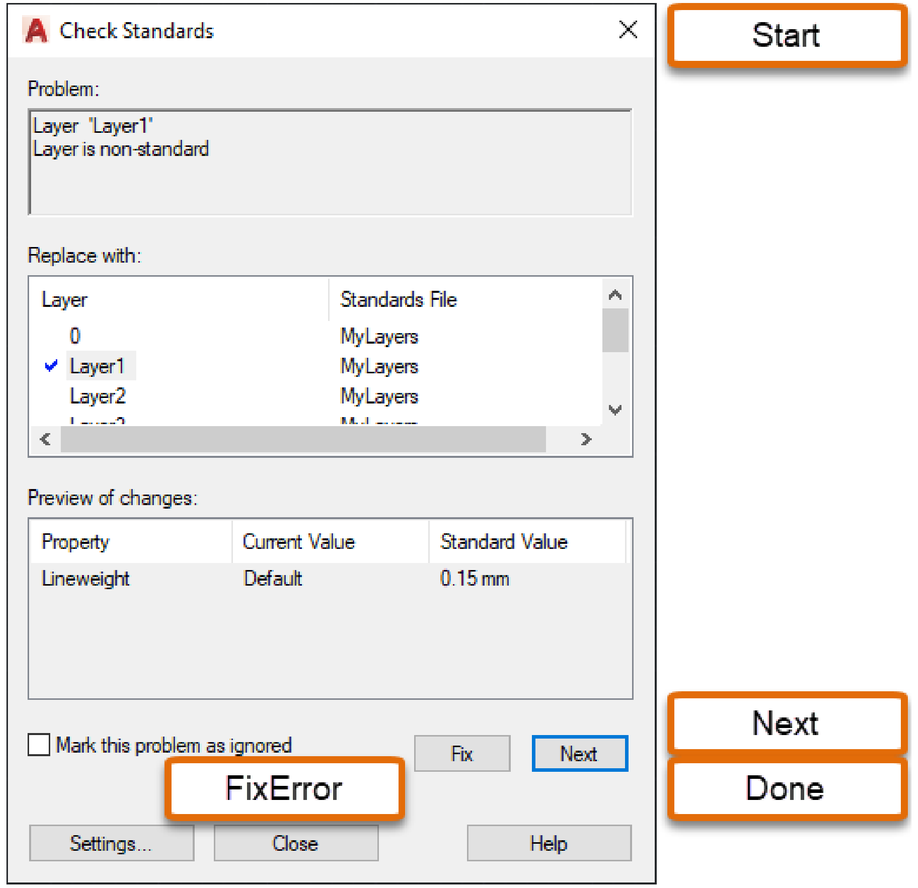
Initialized by these methods:

- `Initialize`
- `GetObjectFilter`
- `SetupForAudit`
- `SetContext`

Iterate Errors

Errors are iterated by these methods:

- `Start`
- `Next`
- `Done`



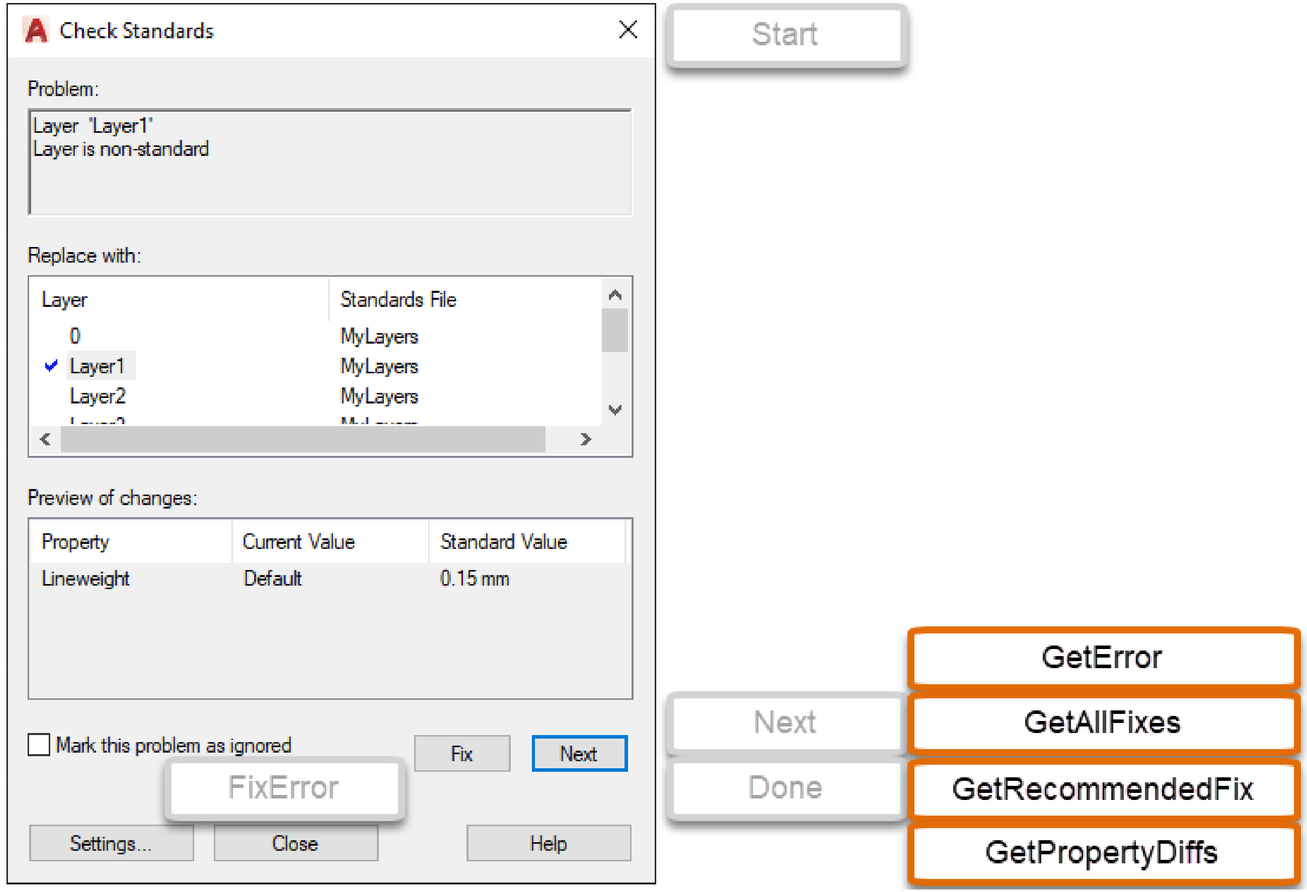
Retrieve Errors and Fixes

Error = Standards violation

Fix = Object that matches standards

Identify errors and fixes with methods:

- `GetError`
- `GetAllFixes`
- `GetRecommendedFix`
- `GetPropertyDiffs`



Fix Errors

An error is fixed when the user

- Manually applies a fix
- Applies a recommended fix

`FixError` method gives the plug-in an oppurtinuty to fix to an error object

`ResultStatus` property of an error object should be updated to reflect its fix state

- `acStResFixed`
- `acStResFixFailed`
- `acStResNotFixable`

Report Errors

Most reporting is handled by the

- CAD Standards framework
- Batch Standards Checker

`WritePluginInfo` method must be implemented to report plug-in information

Need to write other information to the XML file, such as standard files used

Other Methods

`Clear` method is called before the plug-in is released

Must be implemented, but don’t seem to be used:

- `CheckSysvar`
- `StampDatabase`

AutoCAD CAD Standards Checker Flow

- At start up/opening of a drawing:
  - Initialize()
  - GetObjectFilter()
  - SetupForAudit()

- Enabling a standards plug-in:
  - Clear()

- Checking standards:
  - Initialize()
  - GetObjectFilter()
  - SetupForAudit()
  - SetContext()
  - Start()

    (Check Standards dialog box here)

    - Next()
    - Done()
  - Clear()

Batch Standards Check Flow

- Initialize()
- GetObjectFilter()
- SetupForAudit()
- SetContext()
- Start()
  - Next()
  - Done()
- WritePluginInfo()
- CheckSysvar()
- Clear()

What is Being Covered

Extending the CAD Standards framework with a custom plug-in

- What is needed to get started
- Anatomy of a plug-in
- Define a plug-in
- Load and use a plug-in
- Debug a plug-in
- Look at a graphical plug-in

Beyond the CAD Standards API

Define a Plug-in

Define a Plug-in

1. Create a new project using the **Class Library (.NET Framework) template**.
2. Reference the following libraries from the ObjectARX SDK:
   - *AcStMgr.dll* – CAD Standards Manager library
   - *AXDBLib.dll* – AutoCAD ActiveX/COM library
3. Reference the **Microsoft XML Type library (Microsoft XML, v6.0**) COM library. Used for reporting in the Batch Standards Checker.
4. Enable **Make Assembly COM-Visible**.

Define a Plug-in

5. In the main module, add a ProgId declaration before the plug-in class.

```
56
57      <ProgId("AUPlugins.BasicLayers"), ComClass()> _
            1 reference
58      Public Class BasicLayers
59          Implements AcStMgr.IAcStPlugin2
60
```

6. Add the member functions mentioned earlier to the plug-in class.
7. Compile your plug-in as you would any other DLL.

What is Being Covered

Extending the CAD Standards framework with a custom plug-in

- What is needed to get started
- Anatomy of a plug-in
- Define a plug-in
- Load and use a plug-in
- Debug a plug-in
- Look at a graphical plug-in

Beyond the CAD Standards API

Load and Use a Plug-in

Load and Use a Plug-in

After a plug-in has been built, it must be registered to get it loaded

Each plug-in must be registered with:

- Windows
- AutoCAD

Registration is carried out by merging REG files created:

- Using *RegAsm.exe* with the plug-in DLL
- Manually with an ASCII Text editor, such as Notepad

Note: *AULayersPluginCOM.reg* in the Debug output folder and *AULayersPlugin.reg* in the project folder are examples.

Load and Use a Plug-in

Once loaded, you should test your plug-in under the following situations:

- CAD Standards checking in AutoCAD
- Automatic fixes when checking standards in AutoCAD
- Real-time checking in AutoCAD, notifications
- Batch Standards Checker

What is Being Covered

Extending the CAD Standards framework with a custom plug-in

- What is needed to get started
- Anatomy of a plug-in
- Define a plug-in
- Load and use a plug-in
- Debug a plug-in
- Look at a graphical plug-in

Beyond the CAD Standards API

Debug a Plug-in

Debug a Plug-in

1. Compile and register a debug version of the DLL.
2. Specify the location of the AutoCAD executable (*acad.exe*) to launch when debugging starts.
3. Add breakpoints to the statements in which you want to interrupt execution.
4. Open a drawing with a DWS file attached to it or attach a DWS file to the drawing to be checked.
5. Enable the plug-in for checking and start checking the drawing file.
6. Execution will pause on the first breakpoint encountered in your plug-in.

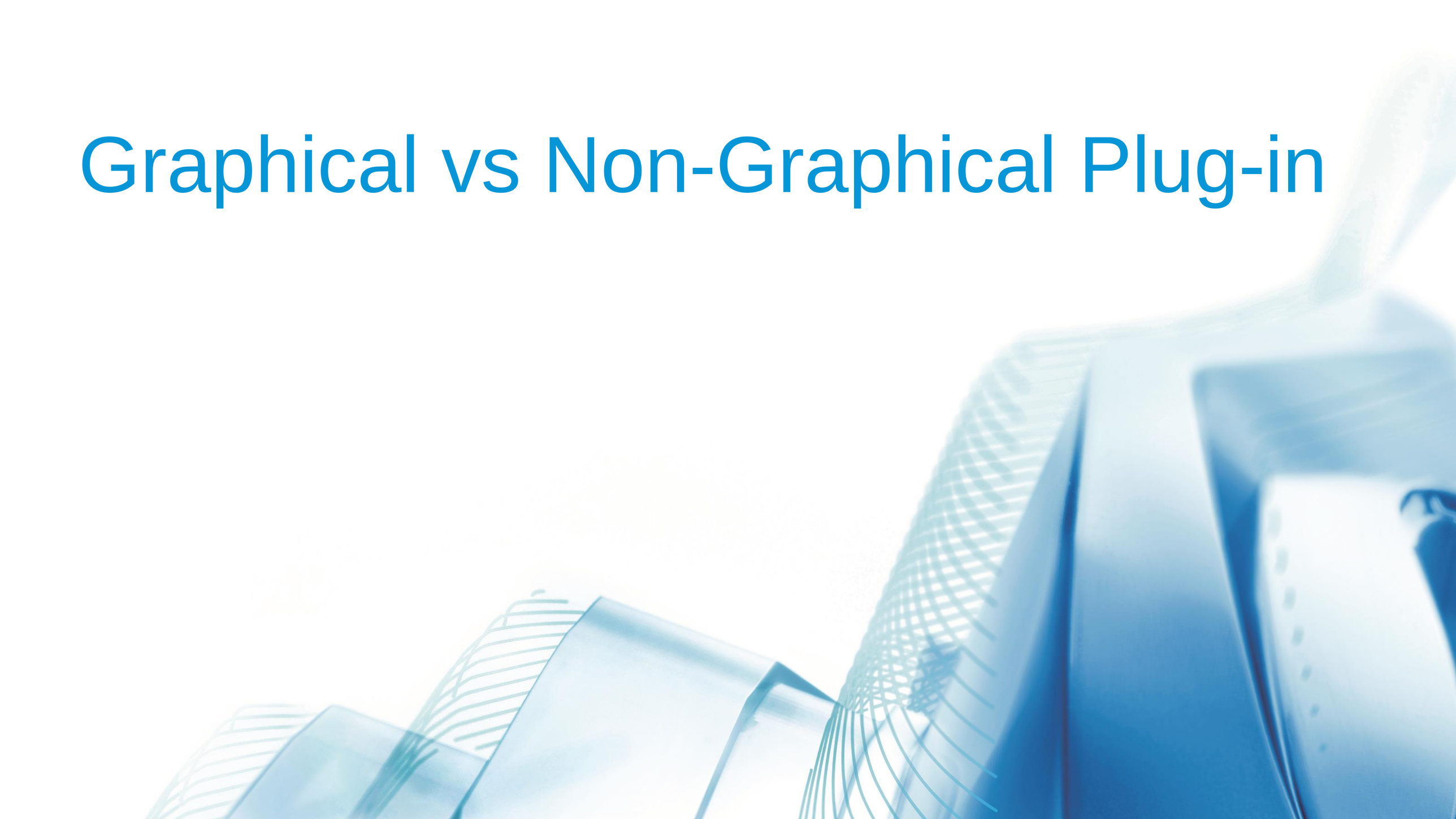
What is Being Covered

Extending the CAD Standards framework with a custom plug-in

- What is needed to get started
- Anatomy of a plug-in
- Define a plug-in
- Load and use a plug-in
- Debug a plug-in
- Look at a graphical plug-in

Beyond the CAD Standards API

Graphical vs Non-Graphical Plug-in



Graphical vs Non-Graphical Plug-in

No real differences, except for how error and fix objects are compared

Differences between the two types of plug-ins created for this session:

- Helper classes in *StandardsHelp.vb*
- Class name, ProgId, a few of the global variables
- `Description()` and `Name()` properties
- `GetObjectFilter()` and `SetupForAudit()` methods
- `PlugIn_Next()` and `PlugIn_Clear()` methods
- `GetAllFixes()` and `GetRecommendedFix()` methods
- `FixError()` and `WriteStandardsItemsInfo()` methods

Beyond the CAD Standards API

Beyond the CAD Standards API

- Compare object properties (current drawing and DWS file) – CompareObjectsBetweenDatabases
- Set a layer current when a command starts/ends – AddCommandEvent/RemoveCommandEvent
- Import objects from a DWS file into current drawing – LoadLayersFromDatabase
- Parse CHX file from the Batch Standards Checker
  - Display property changes found – ParseCHXFile
  - Create a SCR file to fix standards – CHXToSCR

Final Thoughts and Questions

Final Thoughts

Programming can help to enforce CAD standards and allow users to:

- Focus on drafting and design solutions
- Reduce time auditing drawings for errors

Programming has many similarities to *Wonderland in Lewis Caroll's Alice's Adventures*. Both:

- are virtually endless
- hold many mysteries just waiting to be discovered

Questions and ….

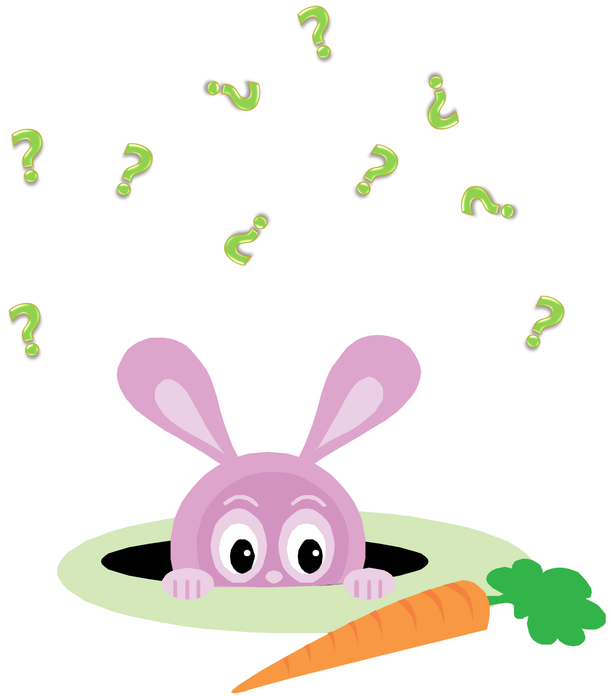
Questions? Questions? Questions?

If you have any further questions, contact me via:

email: lee.ambrosius@autodesk.com

twitter: @leeambrosius

Thanks for choosing this session!



AUTODESK®

Make anything™