AUTODESK **UNIVERSITY** 2015

MSF12017-L

Advance Steel: Hands-on to Extending Detailing Capabilities with Scripting

Mihai Dobrescu

Autodesk

Learning Objectives

- Learn how to access and modify the scripts in Advance Steel for various purposes
- Learn how to build scripts to improve Advance Steel objects filtering in order to obtain better shop drawings automatically
- Learn how to investigate and check correctness of scripts
- Discover the power and boundaries of using scripts for Advance Steel detailing

Description

One of the less-known special features of Advance Steel software is the ability to use User Scripts to aid and personalize various processes. You can use Advance Steel software scripts in several places, including numbering, object filtering for detailing, filtering for detail processes, and numerical control. In this class we will explore the scripting language and walk through the process of adding/modifying scripts to obtain different results in Advance Steel shop drawings.

Your AU Experts

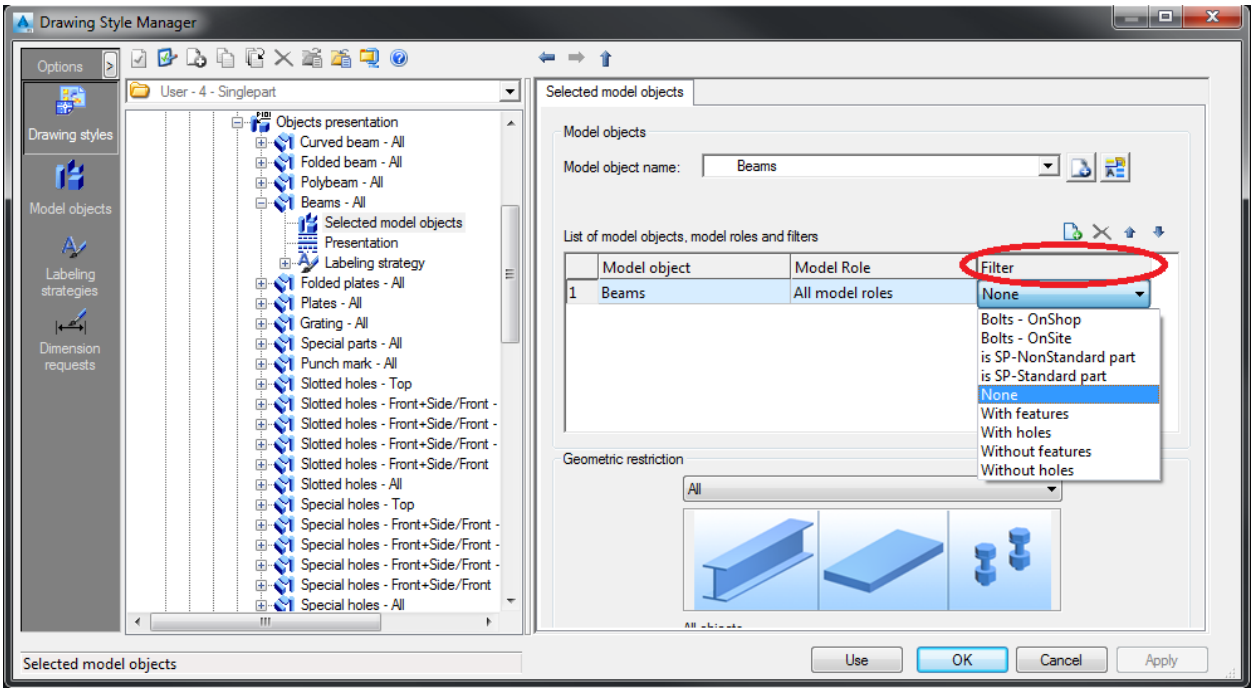
After completing his Information Engineering degree Mihai Dobrescu has been working as a software developer, team leader and software architect on Advance Steel for the past 13 years.

Learn how to access and modify the scripts in Advance Steel for various purposes

Where are the scripts in Advance Steel

In Advance Steel the detailing-related scripts can be viewed and used in two places:

In the "Drawing Style Manager"

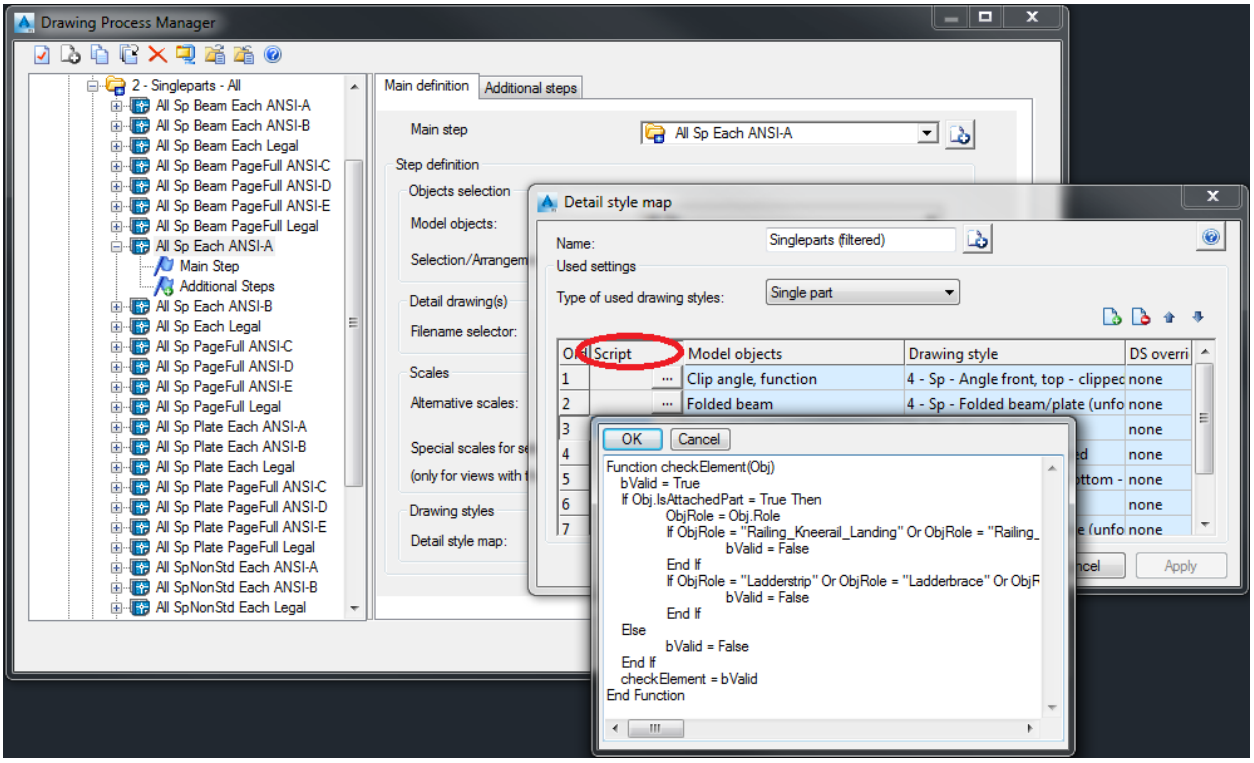


All the scripts already defined in the table "DetObjectQualifierFilter" from the database "AstorDetailsBase.mdb" will appear in the Detail Style Manager and will be available to be used in any combination with an Advance Steel Object Type and Model Role in order to produce a new "Model Object" that will be able to be configured for detailing.

So using a filter here ultimately gives the possibility to differentiate better the objects inside an Advance Steel model for the purpose of configuring differently their appearance in detail drawings:

- Presentation (visible / hidden lines, Symbols, hatches, colors, linetypes)
- Labeling
- Dimensioning

In the "Drawing Process Manager"



In this case, the scripts can be directly attached to a detail style map item from the Advance Steel UI, and the effect of using a script here is that it allows us greater flexibility in choosing which detail style should be applied to which model objects if any at all.

In case you prefer to manually edit databases, those scripts can be found in the table "DetailStyleMapItems" inside the "AstorDetails.mdb" database.

A few examples of interesting checks that can be performed on objects and which allow greater flexibility in drawings creation include:

- Checking if bolts are to be assembled on site or in shop – and placing a different label or symbol for the two cases
- Checking if a beam or plate has any holes on it or not
- Checking if a beam meets the requirements to be detailed as "unfolded" / "template" and if so skip it when detailing the other single parts

Sample script for refining Model Objects

```
'Returns true for objects with holes features
Function checkElement(Obj)
   bHasHoles = False
   'Verify only for beams and plates
   If Obj.IsKindOf(kPlateClass) = True Or Obj.IsKindOf(kBeamClass) = True Then
      'Get the features of this object
      Set features = Obj.getFeatures
       'Verify if any of the features is a hole
       For Each feature in features
         If feature.IsKindOf(kHoleFeat) = True Then
            bHasHoles = True
            Exit For
         End If
      Next
   End If
   checkElement = bHasHoles
End Function
```

Dissecting the script

'Returns true for objects with holes features

The first line starts with an apostrophe – this makes it a comment, a line that is totally ignored by the script running engine, and it is only there to help human readers understand easier what the script does.

There are a number of other comments as well in the script above, they are all meant to explain what we are about to do.

Function checkElement(Obj)

This line defines a function in this script and there are a few things I want to highlight about it:

- The name of the function should not be changed or Advance Steel will not execute the script
- The name of the function is later used to set the return value of "True" or "False" – see the line near the end saying "*checkElement = bHasHoles*" this is where we say this function will return the value of the "*bHasHoles*" variable.
- The function needs to end with the keywords "*End Function*" as you see in the example above
- The "*Obj*" parameter of this function is actually the object that Advance Steel passes to you for checking. Depending on where you insert this script in the Detailing workflow the object passed here could be a beam, plate, bolt, weld, etc.

bHasHoles = False

Here, the developer just defined a new variable and assigned its default status to “False”. Looking at the rest of the script as well, the implication of this line is that if the object passed form Advance Steel is not a beam or plate, or it is but does not have any features, or none of the features is a hole, this script wants to return “False” which is equivalent to telling Advance Steel: “skip this object, do not use it for whatever you were doing”.

If Obj.IsKindOf(kPlateClass) = True Or Obj.IsKindOf(kBeamClass) = True Then

What we see here is the beginning of a standard “If” check:

If <condition> Then

<code here that is executed when condition evaluates to “true”>

Else

<code here that is executed when condition evaluates to “false”>

End If

The condition that can be checked can sometimes be very complex and obtained by combining several different checks using “Or” or “And” operators like in the situation above.

Set features = Obj.getFeatures

At this point having checked that the input object is a beam or plate we ask for its “features” and save them in a new variable called “features”. The “Set” keyword is important here (probably) because the returned element is an object and not a simple value.

For Each feature in features

There are at least two ways to iterate an array in VBScript:

1.

```
For <counter>=<start> To <end> [Step <step>]
        <code here>
Next
```

2.

```
For Each <element> In <group>
        <code here>
Next
```

The script above simply employs the second version. Note that there is also possible to exit the for loop before processing all the elements by using an “*Exit For*” command that you can also see used in our example script.

Learn how to build scripts to improve Advance Steel objects filtering in order to obtain better shop drawings automatically

Copy-paste script before into a new notepad window

```
'Returns true for objects with holes features
Function checkElement(Obj)
   bHasHoles = False
   'Verify only for beams and plates
   If Obj.IsKindOf(kPlateClass) = True Or Obj.IsKindOf(kBeamClass) = True Then
      'Get the features of this object
      Set features = Obj.getFeatures
       'Verify if any of the features is a hole
       For Each feature in features
         If feature.IsKindOf(kHoleFeat) = True Then
            bHasHoles = True
            Exit For
         End If
      Next
   End If
   checkElement = bHasHoles
End Function
```

Decide on a different functionality that we need

After a bit of discussion with my QA colleagues who know more about what users need when it comes to Advance Steel drawings automation, I stopped at the following interesting application: let's try to build a script which knows when a certain beam needs an unwind detail, in order to separate those beams from the others for which we only need front/top views during a shop drawings generation process.

Edit the script to do what we want

```
'Returns true for beams that need unwind details
Function checkElement(Obj)
   bNeedsUnwind = False
   'Verify only for beams-and plates
   If Obj.IsKindOf(kPlateClass) = True Or Obj.IsKindOf(kBeamClass) = True Then
      'First verify that this beam is unwindable
      If 1 == Obj.getIsUnwind Then
                'Get the features of this object
                Set features = Obj.getFeatures
                 'Verify if any of the features is not a hole
                 For Each feature in features
                   If feature.IsKindOf(kHoleFeat) = False Then
                      bNeedsUnwind = True
                      Exit For
                   End If
                Next
      End If
   End If
   checkElement = bNeedsUnwind
End Function
```

Use the new script inside an Advance Steel process

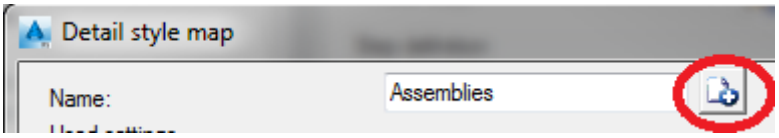
Start the Drawing Process Manager

Navigate to User / Drawing Processes / Assemblies - All.

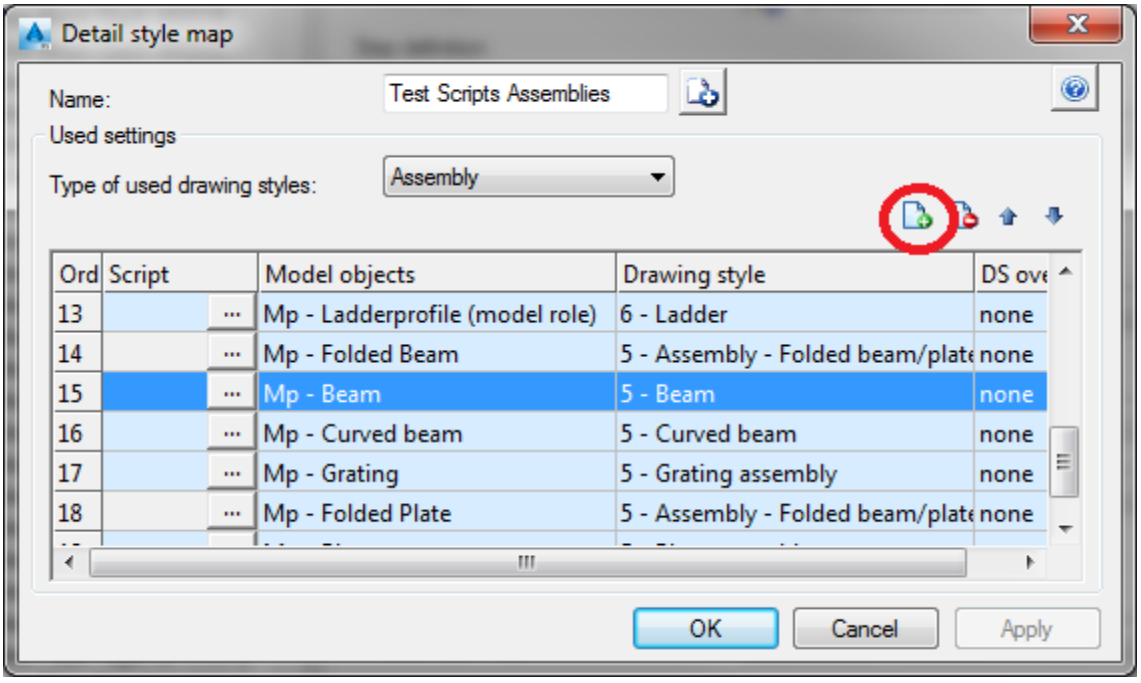
Choose a process from the list for example “All Assemblies Beam Each ANSI-A”.

Go to “Detal Style Map” and press the edit button.

In order to avoid breaking other processes and also to be able to easily roll back our changes we should take note of the current Detail style map name, ex: “Singleparts (filtered)”, press the “new” button next to the name in order to duplicate this Map under a new name.

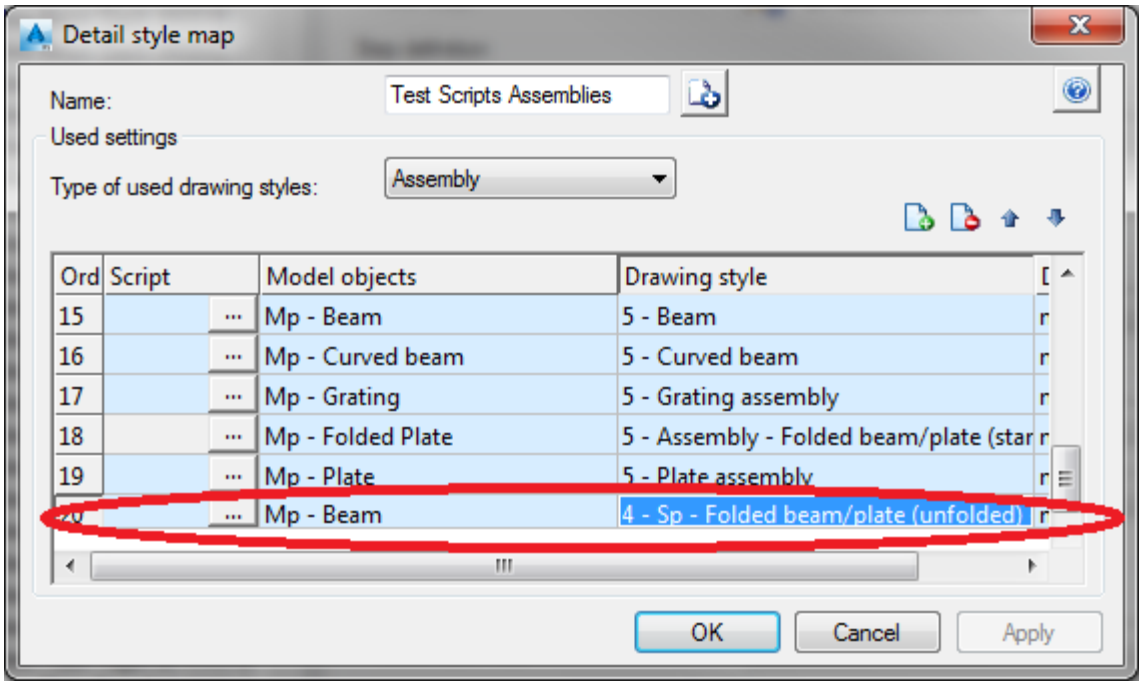


Let’s call the new map “Test Scripts Assemblies”.

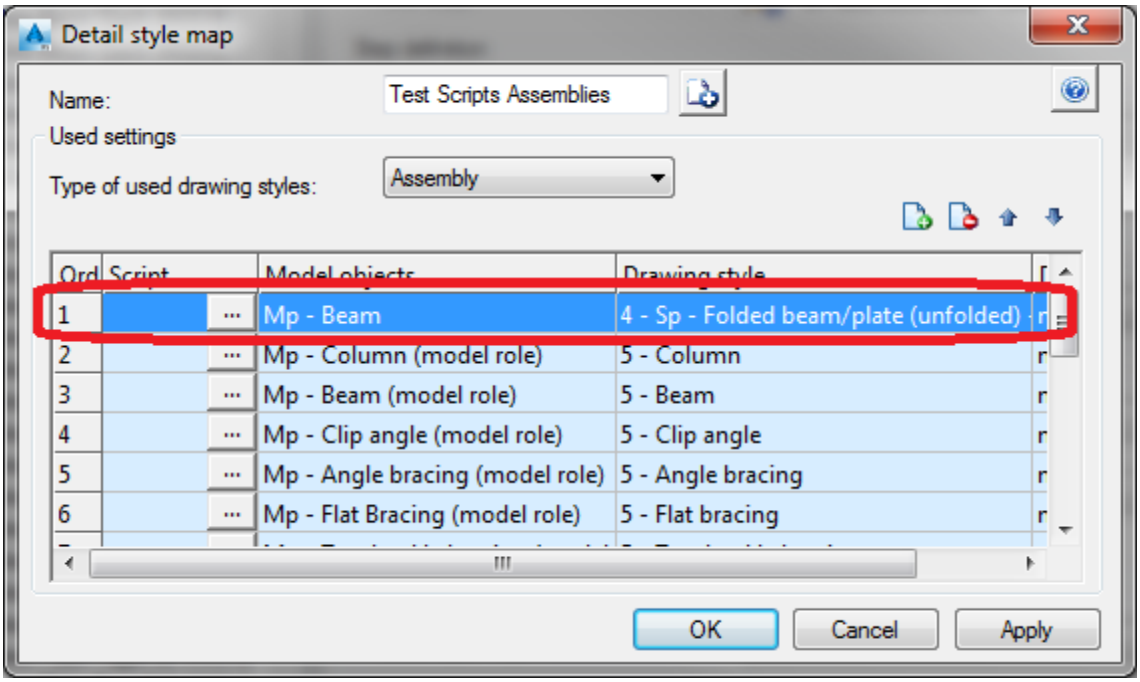


Notice that whatever line you have selected before pressing "New" will be copied and added to the end of the list – this can reduce the number of changes you need to make to the new item.

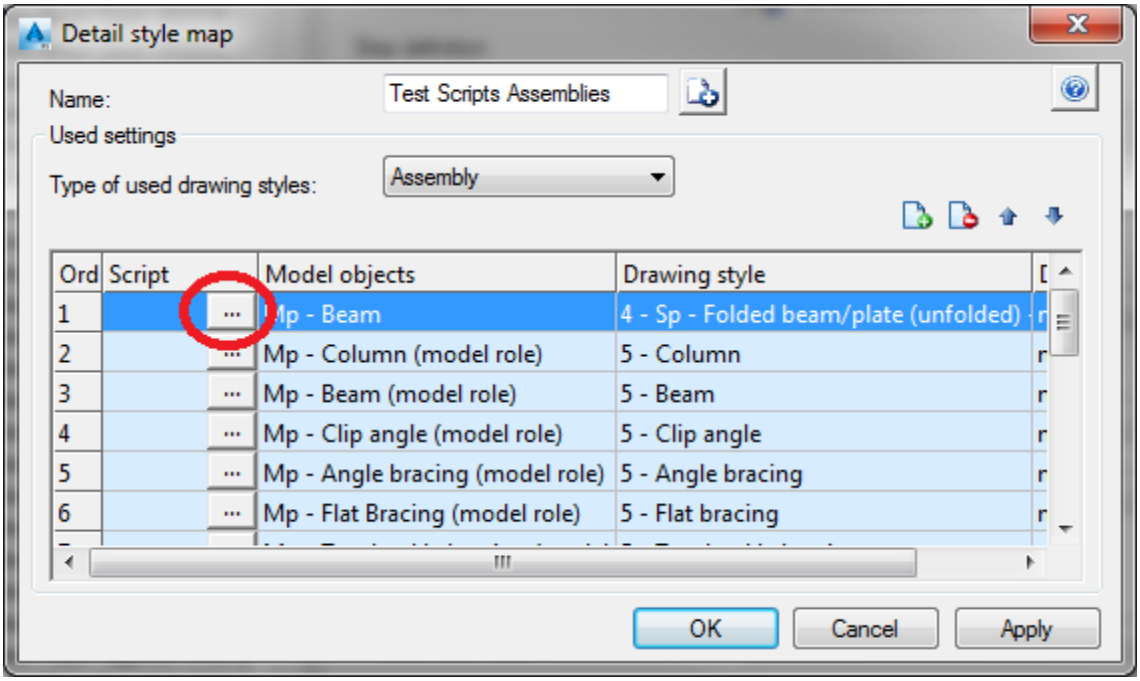
What we want to do now is add a new line that will associate "beams" to an unwind detail style, like in the following image:



The next thing we want to do is move this new line we added to be the first in the list of "model objects" – "detail styles" association



And finally, we need to click on the script edit button and paste our script code



The corrected script version without all the formatting and colors that can be copy-pasted there in Advance Steel, is this:

```
'Returns true for beams that need unwind details
Function checkElement(Obj)
   bNeedsUnwind = False
   'Verify only for beams and plates
   If Obj.IsKindOf(kBeamClass) = True Then
      'First verify that this beam is unwindable
      If Obj.getIsUnwind = 1 Then
         'Get the features of this object
         Set features = Obj.getFeatures
         'Verify if any of the features is not a hole
         For Each feature in features
            If feature.IsKindOf(kHoleFeat) = False Then
               bNeedsUnwind = True
               Exit For
            End If
         Next
      End If
   End If
   checkElement = bNeedsUnwind
End Function
```

Remember to click Ok / Apply / Ok after pasting the script in order to make sure your changes are saved in the Advance Steel databases.

Let's use the process we just configured and see how it works

Time to see what happens

Learn how to investigate and check correctness of scripts

Make sure 1-2 mistakes are slipping inside the new code

Change on = to an ==

Introduce the faulty script in Advance Steel to see what happens

Employ the simplest scripts debug method: the MsgBox

Debug the "getIsUnwind" return value

Test the value of a certain random property – say a user attribute

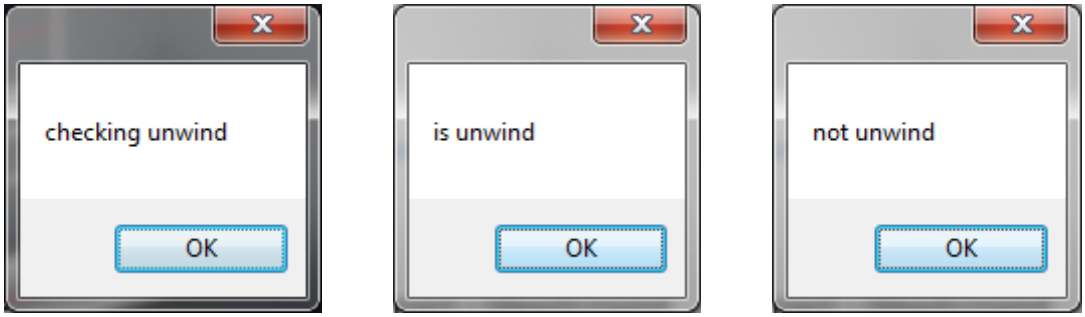
Let's modify the script as follows, copy and paste it again instead of the old one.

Attention! – After modifying the script (unless Advance Steel is restarted) we need to call the "AstorReopenDatabase" command which will invalidate the cached script content.

```
'Returns true for beams that need unwind details
Function checkElement(Obj)
  bNeedsUnwind = False
  'Verify only for beams and plates
  If Obj.IsKindOf(kBeamClass) = True Then
    'First verify that this beam is unwindable
MsgBox "Checking unwind"
    If Obj.getIsUnwind = 1 Then
MsgBox "is unwindable"
      'Get the features of this object
      Set features = Obj.getFeatures
      'Verify if any of the features is not a hole
      For Each feature in features
        If feature.IsKindOf(kHoleFeat) = False Then
          bNeedsUnwind = True
          Exit For
        End If
      Next
```

```
        Else
MsgBox "not unwindable"
        End If
    End If
    checkElement = bNeedsUnwind
End Function
```

When running the script we should see small informative windows like the ones below:



Next, instead of the *MsgBox "Checking unwind"* we could try a MsgBox Obj.`UserAttribute(2)`

Discover the power and boundaries of using scripts for Advance Steel detailing

List of Advance Steel objects class that can be used inside the scripts

kUnknownClass = 0, kAreaObjectClass = 29, kOpeningObjectClass = 30, kCamera = 90, kBeamClass = 200, kBeamStraightClass = 203, kBeamShorteningClass = 206, kBeamNotchStdClass = 210, kBeamContourNotchClass = 212, kBeamMultiContourNotch = 213, kBeamBentClass = 214, kBeamUnfoldedClass = 215, kBeamCompoundClass = 216, kBeamStraightCompoundClass = 217, kBeamPolyClass = 218, kBeamNotchExClass = 219, kConcreteBeamClass = 220, kTimberBeam = 221, kConcreteBeamBentClass = 223, kTaperedBeamClass = 225, kPlateClass = 300, kPlateFeatVertChamfer = 303, kPlateFeatVertFillet = 304, kPlateFeatContour = 305, kPlateContourNotch = 306, kPlateWeldingPreparation = 308, kPlateWeldingBevel = 309, kPlateWeldingFillet = 310, kPlateFoldClass = 312, kPlateFoldRelationClass = 313,	kPlateFoldedClass = 314, kGratingClass = 315, kWallClass = 335, kSlabClass = 336, kFootingIsolated = 337, kBoltPattern = 400, kScrewBoltPattern = 401, kCircleScrewBoltPattern = 402, kCountableScrewBoltPattern = 403, kFinitrectScrewBoltPattern = 404, kInfinitrectScrewBoltPattern = 405, kInfinitmidScrewBoltPattern = 406, kAnchorPattern = 407, kStructuralMultipleSimpleFrame = 501, kStructuralSingle3GFrameSymm = 505, kStructuralMultipleTowerFrame = 507, kStructuralSingleSlopeFrame = 508, kStructuralBox = 511, kObjectExternal = 512, kWeldPattern = 800, kAstWeldLevel1 = 801, kAstWeldStraight = 803, kHoleFeat = 1000, kHoleBeam = 1001, kHolePlate = 1002, kConnector = 1070, kSpecialPartWithBlock = 1100, kUserDefinedCS = 1204, kUserDefinedPoint = 1205

In order to get a complete list of properties that you can query for each type of objects, you should have a look at the document ***Advance Steel COM API Reference Guide.doc*** which is part of the Advance Steel disks or downloadable images staring with the 2016 version. This document is designed to help users who want to develop Advance Steel Joints (Intelligent Connections) but the objects described in that document are the same with the ones accessible through the scripts and have the same properties.

Contents