Using Automation in Civil 3D for Construction Documentation and Exports

Jowenn S. Lua
Arcadis

Andrew Milford
Autodesk

Learning Objectives

- Learn how to capitalize on Country Kit automation tools to assist in detailed modeling and drawing production
- Learn how to create efficient workflows when building Civil 3D models to include owner asset data
- Learn how to effectively use Property Sets to manage asset tagging
- Learn how to use automation (.NET and VBScript) to develop and assign asset tags based on model data

Description

Delivering a highway project is becoming time consuming, as we are now required to provide more information. Efficiently providing 3D models with asset tags with rich metadata is the key. In this class, you will learn how to capitalize on automation in your Civil 3D models and add customized asset information tags using Property Sets. Learn efficient workflows for developing corridor models, as well as adopting intelligent tools to improve production of detailed model elements, such as 3D road features.

Speaker(s)

Jowenn Lua is a Senior Civil 3D Designer for Arcadis. He holds a B.S. Civil Engineering degree from De La Salle University and MBA degree from Ateneo De Manila University. He is a member of the 2016 Autodesk Gunslinger in Australia and participated in the ‘2018 Civil Infrastructure Inside the Factory’ held in Potsdam, Germany. He has over 15 years industry experience in design and construction. He was previously working as the Lead Civil 3D designer in the Sydney Metro and is one of the key person who developed the project’s Civil 3D system, template, plan production and tool palette pushout. He is highly skilled in Civil 3D & subassembly composer and provides mentoring and solutions to the team.
Andrew Milford is a Senior Civil Infrastructure Implementation Consultant for Autodesk and is responsible for providing technical and business consulting, ensuring customers achieve successful adoption of Autodesk’s Infrastructure Solutions across the Asia/Pacific region. Prior to joining Autodesk, Andrew gained over 24 years of design experience in the civil infrastructure industry, working as a geometric road designer for large consulting companies such as SKM/Jacobs, Arcadis Consulting, and AECOM. Andrew’s experience extends from designing large highways, tunnels, and interchanges down to smaller subdivision work using a variety of different design and drafting software packages. He is an AutoCAD Civil 3D Certified Professional and loves diving deep into AutoCAD Civil 3D software to develop and automate processes through scripts, AutoLISP, .NET API C# and VB, and Python.
Introduction

Delivering a 3D model with rich asset data is fast becoming a standard requirement in the industry now. Every new project is now asking for BIM deliverables specified in a BIM Execution Plan (BEP). This class will help you improve your current Civil 3D workflow and will guide you in using a combination of intelligent tools and processes to improve your design and drawing production. This class will also demonstrate practical workflows in attaching asset information to AutoCAD objects using property set data.

ANZ Country Kit Automation Tools

In this class, we would like to introduce the ANZ Country Kit automation tools. This can help you enhance your output productivity in your company.

Where to get the ANZ Country Kit?
You can download the ANZ country kit in the Autodesk Knowledge Network:
https://knowledge.autodesk.com/support/civil-3d/troubleshooting/caas/downloads/content/civil-3d-country-kits-for-australia-new-zealand.html

Civil 3D 2019 Australia New Zealand shortcut

Target:
"C:\Program Files\Autodesk\AutoCAD 2019\acad.exe" /Id "C:\Program Files\Autodesk\AutoCAD 2019\AecBase.dbx" /p "<<C3D_ANZ>>" /product "C3D"

Start in:
"C:\Program Files\Autodesk\AutoCAD 2019\UserDataCache\"
Where to find the ANZ Tools
Toolbox > Australia and New Zealand Reports Manager > ANZ Tools

ANZ Tools were developed to enhance productivity and to provide quick and clean output that will eliminate a lot of manual drafting clean up. This helps the designer to provide a better data export.

Upon installation of the ANZ Country Kit, you will have a copy of all the .Net dll.

C:\ProgramData\Autodesk\C3D 2019\enu\Data\ToolBox\ANZ

Name

- Autodesk.Consulting.Civil3D.ANZ_ExportForConstruction.dll
- Autodesk.Consulting.Civil3D.Barriers.dll
- Autodesk.Consulting.Civil3D.FeatureLineExport.dll
- Autodesk.Consulting.Civil3D.ProfileDataBandCopy.dll
- Autodesk.Consulting.Civil3D.ProfileDatum.dll
- Autodesk.Consulting.Civil3D.SectionLabel.dll
- Genio2D.dll
Export for Construction ANZ
Export corridor and site feature lines to AutoCAD (as a 2D or 3D drawing)

Refer to sample data set: C3D2019_1_ExportForConstructionANZ.dwg
Export for construction ANZ tool is the quickest way to extract 2D and 3D strings of all corridor featurelines and site featurelines. This is useful for design collaboration and help us verify our design models.
Section View Labels

This tool will allow the user to select a single Section View (as part of a Section View Group) and annotate user-defined point codes within data bands, allowing for staggering of overlapping text labels. This requires a system variable ‘Measurement’ to be 0 for imperial and 1 for metric.

To be successful in using this tool, you need to setup your Civil 3D settings and follow the guides below:

**Step 1**
Go to General > Label Styles > Marker
Create a new style called ‘ADSK_SectionLabel’ or copy from ANZ template
Step 2
Go to General > Multipurpose Styles > Code Set Styles
Create / open code set style you want to use e.g. _NEW CODE SET
Assign the ‘ADSK_SectionLabel’ to your code set style under the Point > Label Style category

Step 3
Go to Section View > Section View Style > Band Styles > Section Data
Create your section data for Design, Existing, Codes and Offset e.g. _DESIGN, _EXISTING, _CODE and _OFFSET
Step 4
Go to Section View > Band Styles > Band Sets
Create _New Band Set and assign the section data above
Step 5
Go to Section View > Commands > CreateMultipleSectionView
Edit Command Settings then assign the _New Band Set to Section View Band Set

Step 6
Create your alignment, profile, assembly, corridor, sample lines and multiple section.
Step 7
Go to Toolbox > Australia and New Zealand Reports Manager > ANZ Tools
Click the Section View Labels and select any section view to activate label tools.

As a guide refer to ANZ Template:
Folder: %LocalAppData%\Autodesk\C3D 2019\enu\Template
File: _AutoCAD Civil 3D 2019 ANZ Design_RMS.dwt

Sample output using the guide ANZ Template:
Section View Labels (Additional User Guide)

1. The section views must be part of a section view group (No individual sections)
2. This tool annotates only corridor sections displayed on a section view. Surface sections are not used to label the specific point codes, except for the existing surface, which is used to extract levels at the corridor section cut offset locations.
3. To determine which labels to annotate, edit the Code Set Style. Under the Point category, assign a Label Style called ‘ADSK_SectionLabel’ (not case sensitive). This tells the program which codes to annotate.

Add a value to the ‘Description’ column to override the feature label in the data band.
4. Assigning a label style called ‘ADSK_SectionLabel_Sub’ will allow you to annotate a separate set of points along the Subgrade (or Datum) data band, separate from the top surface design strings.

5. In the section view style dialog, adding the text string ‘#XYZ’ to the description box annotates the master baseline (alignment) label just above the datum (LHS).
6. The SectionLabel tool annotates all specified text values within existing Section View Data Bands, including labelling the existing surface at the same 'cut' locations, specified by the user. To avoid excessive user-interface, the program is hard-coded to search through all Section View Data Bands in your DWG file, and return text information (height, style etc.) that matches any of the following naming criteria. Note the data band labels only should contain any of the following text strings and is not case sensitive.

Name the Data Bands in accordance with the following criteria:
- Feature lines / codes
  - FEAT
  - LABEL
  - CODE
- Design Levels
  - DESI
  - PROP
- Existing Levels
  - EXIS
  - NATU
- Offset
  - OFF
  - DIST
- Subgrade / Datum
  - SUB
  - STRAT
  - DATUM

7. In the Section Data Band Style dialog, edit the Data Band text style through the 'Summary' tab – Band Details – Band Text Style.
8. In the Section Data Band Style dialog, edit the Data Band text height through the ‘Band Details’ tab – Grade Breaks – Compose Label.

In the Label Style Composer, add a Text Component, and change its text height value. The tool will read this value to control the text heights for the data band.
9. Inside the Section Data Band Style dialog, adding the text string ‘#NoTicks’ into the Description will remove ticks from the Data Band.
10. To annotate labels on corridor sections, the corridor section code set style must be setup for the section labels to be cut.

11. The SectionLabel tool annotates an existing surface at the same ‘cut' locations as specified by the user. To avoid excessive user-interface, the program is hard-coded to search through all TIN surfaces in your DWG file, and return the first surface that matches any of the following naming criteria.

Note the surface only should contain any of the following text strings and is not case sensitive.

- EX
- EG
- GROU
- TERR
- NGL
- TX
- SURV

For example, a surface called ‘Existing Ground' will be returned, as it contains ‘EX' and ‘GROU' within the name.
Barriers
Create 2D & 3D ANZ Style barriers from Civil 3D Alignments.

To be successful in using this tools, you need to have the following:
1. 3D blocks for RMS wire rope or equivalent user defined block.
2. 3D blocks for RMS G4 barrier or equivalent user defined block.

You can get sample 3D blocks from the ANZ Template:
Folder: %LocalAppData%\Autodesk\C3D 2019\enu\Template
File: _AutoCAD Civil 3D 2019 ANZ Design_RMS.dwt

<table>
<thead>
<tr>
<th>Block Name</th>
<th>2D View w/ Insertion</th>
<th>2D View</th>
<th>3D View</th>
</tr>
</thead>
<tbody>
<tr>
<td>G4_PostLHS</td>
<td><img src="image1" alt="" /></td>
<td><img src="image2" alt="" /></td>
<td><img src="image3" alt="" /></td>
</tr>
<tr>
<td>G4_PostRHS</td>
<td><img src="image4" alt="" /></td>
<td><img src="image5" alt="" /></td>
<td><img src="image6" alt="" /></td>
</tr>
<tr>
<td>TL3_Post</td>
<td><img src="image7" alt="" /></td>
<td><img src="image8" alt="" /></td>
<td><img src="image9" alt="" /></td>
</tr>
<tr>
<td>TL3_Terminal</td>
<td><img src="image10" alt="" /></td>
<td><img src="image11" alt="" /></td>
<td><img src="image12" alt="" /></td>
</tr>
</tbody>
</table>

3. Create alignment or offset alignment for the barrier horizontal location.
4. Create surface tin with name “Barrier” (Not case sensitive) for the barrier profile.
5. Insert the blocks you want to use in the drawing before running the Barrier tools.
The step-by-step guides below will show you how easy you can use the ANZ Barrier tools and how you can create your own block with the name “Post” or “Term” then the tool will automatically detect it and add it to the list. This opens up the possibility of you creating a lighting pole block and use the tool to layout your lighting pole easily in 3D.

**Step 1**
Refer to sample data set: C3D2019_3_BARRIER_Demo.dwg
Or
Create your own sample drawing:
  a. Open new drawing from Civil 3D 2019 ANZ Template
  b. Bring in a survey tin for example “EG”
  c. Create alignment, profile, assembly and build “Corridor Road”.
  d. Create surface “Corr” from Corridor Road
  e. Create new tin surface called “Barrier-E+C”
  f. Paste the “EG” surface then Paste the “Corr” surface to “Barrier-E+C”
  g. Insert all the blocks needed in to model space of your drawing (G4_PostLHS, G4_PostRHS, TL3_Post and TL3_Terminal)
Step 2
Create alignment where you want to place the barrier or wire rope and name them properly for you to easily identify. For example, all wire rope alignment will use alignment names that start with “BX” and all LHS guard rail will be “BF” and RHS guard rail will be “BG”.

Step 3
Type -VPORT and select 2 then choose horizontal to split the view into two and make the top Plan view (2D Wireframe) and the bottom 3D view (Conceptual).
Step 4
Go to Toolbox > Australia and New Zealand Reports Manager > ANZ Tools
Click the Barriers

- Tick the box if the alignment is intended to be used as Barrier.
- Choose Barrier Type to use (None, Rail Left, Rail Right, WireRope).
- Choose what post to use (This will list all the blocks inserted in the drawing with name “Post”) e.g. G4_PostLHS, G4_PostRHS and TL3_Post.
- Enter value for post spacing, by default it is set to 2.5 units.
- For wire rope, you can add Terminal – Lead and Terminal – Trail or leave as blank or <None> if no end terminal is needed. (This will list all the blocks inserted in the drawing with name “Term”)
- Choose what layer to assign for each barrier.
Step 5
Barrier Tool will pick the first surface that has “Barrier” name and create the barrier post drape to the top of the surface making a 2D and 3D representation of the wire rope and G4 Barriers.

You can go to Toolbox > Australia and New Zealand Reports Manager > ANZ Tools
Click the Barrier again and amend any changes you want to do and the tool will delete and recreate it for you again.
**Step 6**
If you change the surface style of “Barrier-E+C” to ANZ_Triangulation

You can view the barrier and see if it is placed as expected with the correct layers.

**Notes:**
1. Barrier tools won’t work if the alignment is on top of a surface that has a hole in it as the barrier tools need a surface to drape to and if the surface is missing then it won’t return any output.
2. If the alignment includes a curve, it will still try to do a ‘best fit’ to represent how the barrier should look like. Test and try different scenarios to be familiar with the tools.
Sample Output of Barrier Tools
Genio 2D
Genio (Generalized Input Output) is a text file format for exchanging data to MOSS (Modelling by String System). This tool converts a 3D Genio Import to a 2D flattened drawing. By default, it is created to suit Road and Maritime Services (RMS) standard layers and blocks for survey strings. This tool can be customized to suit your own standard layers and blocks.

This is in conjunction with using the following:
1. Default Template
   %LocalAppData%\Autodesk\C3D 2019\enu\Template\_AutoCAD Civil 3D 2019 ANZ Survey RMS.dwt

2. Default Genio Import table mapping file
   C:\ProgramData\Autodesk\C3D 2019\enu\Data\Import Export Extension for GENIO\Genio Import Survey RMS No Layer Prefix.tbl

3. Default Genio 2D mapping txt file
   C:\ProgramData\Autodesk\C3D 2019\enu\Data\ToolBox\ANZ\Settings\genio_import_app_settings_2D.txt

For Autodesk Genio Import-Export Extension, refer to link below:
Download from: https://manage.autodesk.com
Step 1
Create new file from default survey template that has correct layers and blocks.

Sample ANZ Survey_RMS.dwt has default legend created inside the template.

Blocks are built-in inside the template.
Step 2
Go to Toolbox > Subscription Extension Manager > Import from Genio

Step 3
Update Genio Import Options and Genio Import Selection
Step 4
Imported Genio will give you 3D polyline and Cogo Point.
Step 5
Go to Toolbox > Australia and New Zealand Reports Manager > ANZ Tools

Step 6
Converted 2D Genio will give you polyline, block and mtext.
Step 7
Save the drawing and open it again to see all blocks are applied properly. This can now be used as an xref drawing and a 2D representation of the survey genio.

To be successful in using Genio 2D to your company standards, create a copy of the following files below and modify them to suit to your own customized template and mapping file:

1. _AutoCAD Civil 3D 2019 ANZ Survey_RMS.dwt
   - Update the layers
   - Update the blocks

2. Genio Import Survey RMS No Layer Prefix.tbl
   - Make your own mapping to suit your string label

3. genio_import_app_settings_2D.txt
   - Update the code to suit your customized layer and block
Export Feature line XYZ
Export multiple feature lines report into a single CSV file. The report will include feature line name, chainage, easting (X), northing (Y) and elevation (Z).

Step 1
Open a drawing that have feature lines.

Step 2
Go to Toolbox > Australia and New Zealand Reports Manager > ANZ Tools
Step 3
Select a folder then save the csv file.

Step 4
Open the csv report to view.
Both tools above are ideal to use for modifying multiple profiles. If you have created multiple profiles, say 50 profile views, and then you are tasked to change a band settings like adding another band or removing a band or even updating the surface profile of the bands, then you need to do it 50 times. By using “Profile View Data Band Copy” it simplifies the process and save the designer valuable time.

If you are tasked to adjust the profile datum to be of a certain distance from the top most profile band, then using the “Profile View Datum Adjust” is the perfect tool to do this.
Sample Case Scenario for Multiple Profile Views
Create Multiple Profile Views with blank profile data e.g. “Standard”
When you need to add profile data band sets to all your profile views with correct surface data, weeding and stagger options, see below.
Then use Profile View Datum Adjust to adjust the datum with user defined vertical gap.

All profile view will now adjust the datum to allow minimum gap of 20 from the top band.
Civil 3D Assemblies

Naming of assemblies in Civil 3D is critical to ensure consistency across the project, in addition to avoiding conflict where projects require multiple designers.

As part of the property set / asset tagging requirements, a typical assembly coding convention could be as follows:

\[
\text{<Item>-<Item> + <Item>-<Item>}
\]

Where \text{<Item>} represents the component type (i.e. P = Pavement, S = Shoulder, V = Verge, SA = SA Kerb, I = Interface/Daylight). The numeral following the letter represents the number of layers. In our example, \text{P8} depicts an 8-layer pavement. Each item in the group left or right of the baseline is separated by a ‘-’ symbol.

The ‘+’ symbol represents the actual baseline, with subassemblies grouped either left or right of the baseline, annotated by working from left to right across the assembly.

\[
\text{S8-P8-P8 + P8-P8-S8}
\]

In the above example, the first group of three subassemblies to the left of the plus ‘+’ symbol, ‘S8-P8-P8’, indicate (working from left to right) a shoulder containing 8 layers (S), a pavement of 8 layers (P), and a second pavement of 8 layers (P).

For the right side, we simply mirror the three subassemblies from the left.

\[
\text{I-V-SA + + SA-V-I}
\]

In the above example, the left assembly ‘I-V-SA +’, working from left to right, we have an Interface (I), a Verge (V) and a Type SA Kerb (SA). The plus ‘+’ symbol to the far right of the
name indicates the baseline off to the right (i.e. all subassemblies are applied to the left of the baseline)

Likewise, the right assembly ‘+ SA-V-I’, we have a mirrored copy of the previous example, including a Type SA Kerb (SA), a Verge (V) and an Interface (I). The plus ‘+’ symbol to the far left of the name indicates the baseline off to the left (i.e. all subassemblies are applied to the right of the baseline)

With this naming approach, and only a basic understanding of the convention, a designer can edit the project file and easily understand what specific assemblies are used throughout a corridor.

Civil 3D Subassemblies

Naming of subassemblies, and their Point, Link and Shape codes, is critical to ensure consistency across the project from a modelling perspective.

Additionally, following a simple shape naming guideline will help us greatly when the time comes to extract corridor solids and apply our property set asset information. Using property sets, we can manipulate the source corridor shape code information, combining this with the default corridor solid output information to create even more complex and more importantly, unique property set asset tags to the output solids.

For example, some states in Australia are beginning to require complex asset tagging as part of a project deliverable, with an excerpt of the specifications illustrated below.

To begin, each layer within a pavement type requires its own unique codes (or Family), and will require unique property asset tags

Secondly, individual lanes are to be tagged with the following standards:
Adding to this, on a typical roadway project there are usually multiple pavement types with varying depths and materials applies (i.e. concrete pavement, asphalt pavement etc).

Figure 2: Lane Type Naming

Figure 3: Pavement Type Naming and Layers
To satisfy the client requirements for asset tagging, a typical sub assembly shape coding convention could be as follows:

<Family>-<Lane Type>-<Pavement Type>-<Layer No.>

Where <Family> represents the family type (i.e. PLWC = Pavement Layer – Wearing Course, SHLD = Shoulder, KERB = Kerb etc.). The <Lane Type> represents the direction and type of lane (Prescribed or Counter, Through Lane or Shoulder etc.).

The <Pavement Type> is the name typical reserved for the pavement drawings and reports, and finally, a <Layer No.> is assigned in order of material placement (i.e. lower numbers for subgrade, higher numbers for wearing course).

To assist in the creation of complex asset types through property sets, a ‘-’ symbol is placed between each element.

![Diagram showing subassembly properties and layer codes](image)
In the image above, the Layer 1 Shape/Link Code contains the symbol ‘PLWC-PT1-TML1-08’, meaning:

**PLWC = Wearing Course**

**PT1 = Prescribed Through Lane 1 (closest to baseline/median)**

**TML1 = Pavement Type (size, material and name usually specific by pavement engineers)**

08 = Layer Number

**Renaming Subassemblies**

Naming each layer in this manner can take some time upfront, but with the assistance of a little automation, we can streamline this process. For example, a program written using the Civil 3D .NET API allows the user to replace the name of elements within subassemblies. The program, ‘Autodesk.Consulting.Civil3D.SubassemblyParamRename.dll’, will allow the user to select a subassembly (or more than one), enter a string to find, and another string to replace.

To load the dll, enter ‘NETLOAD’ on the AutoCAD Civil 3D command line, navigate to the file location and load the abovementioned file.

To run the program, type ‘SUBPARAMRENAME’ on the command line, and enter the relevant values in the dialog box.

In the image below, all instances where the string ‘PT1’ is found in the selected subassemblies will be replaces with ‘PT2’

![Figure 4: Renaming Subassembly Parameters](image-url)
Extract Corridor Solids
There are multiple ways to extract corridor solids in Civil 3D, each version giving slightly different outputs, which we will need to consider when setting up corridor solids models.

Codes to Extract
The Layer Name Template for the Codes to Extract section contains several fields to create unique layer names from your corridor information, some of which include:

- Region Start/End Station
- Region Name
- Corridor Name
- Incremental Counter
- Subassembly Name, Shape Index, Side
In this instance, a simple `<[Codes]>` field is used, as this will generate several layers from which we will use as a base for more complex attribute data.

Property Data
As part of the solid generation process, Civil 3D will now automatically attach property set information to solids extracted from the corridor. The categories of ‘Corridor Model Information’ and ‘Corridor Shape Information’ are hard-coded, and therefore cannot be deleted or amended.

The ‘Corridor Property Data - User Defined’ category is where custom properties can be defined. To add a user-defined property, expand the Property Set Definitions and select the ‘Corridor Property Data – User Defined’ row. Right click in an empty row and clock ‘Add’ to add a new Manual property. Note that only Manual property definitions can be added here.

To assist in the creation of more complex property set definitions downstream, it is recommended to add an AutoCAD/Civil 3D object’s ‘Handle’ Automatic property into this category via the Style Manager (i.e. not through the ‘Extract Corridor Solids’ dialog, as this only supports Manual definitions)
Output Options

*Insert into Current Drawing*
This option enables a Dynamic link to the corridor, and works most effectively when creating solids from a model containing a dref’d corridor model. This is a desirable option as it maintains a single solid for each linked corridor region, and updates even when the station ranges are adjusted.

It should also be noted that by using this output option, certain elements are not populated in the automated Property Sets, including:

- AssemblyName
- Side
**Add to an existing drawing**
This will add solids to an existing drawing. In this option existing entities will not be overwritten, which could possible result in duplicate objects within the same baseline and region. Additionally, this option will extract more information than when using the ‘Insert into Current Drawing’ option.

**Add to a new drawing**
This allows you to export the corridor solids to a new drawing. It behaves in a similar manner to extracting to an existing drawing and extracts the same level of information.
Civil 3D Property Sets

Property Sets in Civil 3D are used to add additional custom asset information to Civil 3D and AutoCAD objects.

The Property Sets can be accessed from the ‘Manage’ tab in the ribbon and selecting ‘Define Property Sets’.

Inside the Style Manager dialog, note the folder hierarchy, and how it is depicted as:

- `<Document Name>`
  - Documentation Objects
    - Property Set Definitions

![Figure 6: Accessing Property Sets](image_url)
However, it should be noted that accessing the Style Manager via the Civil 3D ribbon produces a cut-down version of the toolbox. To access a more feature-rich version of the dialog, enter ‘STYLEMANAGER’ at the command line and hit Enter.

Examine the folder hierarchy, and how it is depicted as:

- `<Document Name>`
  - Documentation Objects
    - Property Data Formats *
    - Property Set Definitions
  - Multi-Purpose Objects *
    - List Definitions *

* Denotes new items in Style Manager. Note there are more categories available, this document highlights items only to be used in this example.
To apply a property set definition to a drawing, select the ‘Property Set Definition’, right-click and select ‘New’

**Figure 8: Style Manager from Command Line**
In the ‘General’ tab, Enter a new Property Set Definition name

In the ‘Applies To’ tab, select the object to which the Property Set Definition is to be applied
In the ‘Definition’ tab, add the relevant Property Definitions (Manual, Auto or Formula). These are explained in more detail below.

The 5 Stages of Property Sets
When creating property set definitions, it is desirable to setup the properties to pre-populate as many properties through automation as possible. This can be achieved in a variety of ways. The diagram below shows the five key methods in which property sets can be established within Civil 3D, including:
Manual Property Definitions

These are the simplest of property definitions, and subsequently require the most manual input in order to populate their respective data fields. We can leverage Manual definitions inside of Formula definitions to create more complex expressions.

To create a manual property, select the button in from the list on the dialog. Manual properties require, you guessed it, manual inputs for values, but some of the additional advantages include custom formatting and list selection, which are explained in more detail below.

List Selection

To provide consistency in the values entered, creating lists ensures data compliance by allowing the end use to select only from a specified range of values.

To create a list, enter the Style Manager via the command line ‘STYLEMANAGER’, and navigate to Multi-Purpose Objects -> List Definitions. Right-click and select ‘New’

- In the General tab, enter a new name.
- From the ‘Applies To’ tab, select the ‘Manual Property Definition’ checkbox.
- Finally, add the items to your list in the ‘Items’ tab with the ‘Add’ button.
  - Selecting the ‘Allow individual property value to vary from this list’ checkbox allows user to enter their own values outside of those specified in the list. Use this only if there is the likelihood of the list changing.
### General Table

<table>
<thead>
<tr>
<th>Asset Code</th>
<th>Attribute</th>
<th>Domain Name</th>
<th>List Code</th>
<th>Code Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAC</td>
<td>AC Type</td>
<td>PLAC_TYPE</td>
<td>CM</td>
<td>Cold Mix</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CR</td>
<td>Crumbled Rubber</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DG</td>
<td>Dense Grade</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FG</td>
<td>Fine Gap Grade</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OG</td>
<td>Open Grade</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PR</td>
<td>Proprietary</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SMA</td>
<td>Stone Mastic</td>
<td></td>
</tr>
</tbody>
</table>

### Additional Table

- **PLWC_TYPE**

### Manual Property Definition
- **Space Names**
- **Zone Names**
Custom Formatting
To provide consistency in the manually created data values, creating a custom Property Data Format ensures that data entered into a property set retains a specific constant format

To create a Property Data Format, enter the Style Manager via the command line ‘STYLEMANAGER’, and navigate to Documentation Objects -> Property Data Formats. Right-click and select ‘New’

- In the General tab, enter a new name.
- From the ‘Formatting’ tab, enter such formatting items such as
  - Prefix / Suffix
  - Rounding / Precision
  - Zero Padding
  - Separators

- *In this example, a Zero Padding of 7 digits was used*
Creating Manual Properties (with Lists and Custom Formatting)
Select the relevant Property Set Definition and create a new Manual property

Under the Type Column, note we can set the data type to any of the following:

- **Auto-Increment – Character**: Creates a unique text character for each element
- **Auto-Increment – Integer**: Creates a unique integer for each element
- **Integer**: A number with no fraction (i.e. 1, 2, -1, etc)
- **List**: A list of predefined values (see previous section)
- **Real**: A number with decimal places (i.e. 12.356)
- **Text**: A text or string value (i.e. “Civil 3D Rocks”)
- **True / False**: A Boolean value (i.e. True or False)

When a ‘List’ item is selected from the ‘Types’ column, the ‘Source’ column populates with the predefined lists (defined in the previous section)
In addition to setting simple data types, we can add custom formatting to the property. Not in the image below, we can apply our 'Integer-Pad-7' format to the property. This will ensure that all values added share the same 7 digits. For example, entering '120' into the Extended Data will result in a value of '0000120'.
Automatic Property Definitions

Automatic properties are assigned to AutoCAD / Civil 3D objects, which can extract object properties to display as attributes. Automatic properties will vary depending on what objects are defined in the Property Set Definition. Some examples of useful commonly used automatic properties are:

**All Objects**
- Handle
- Object ID
- Layer

Adding a ‘Handle’ or ‘Object ID’ automatic property is a useful way to extract more information downstream using Formula definitions with COM interop.

Note that Civil 3D elements (i.e. Corridor, Surfaces, Feature Lines etc) added to a Property Set Definition do not expose any useful properties. To expose Civil 3D object properties, we need to use a combination of Formula property definitions combined with some COM

**3D Solids**
- Volume (m3)

**Block References**
- Position (X, Y, Z)
- Name
- Rotation

To create an automatic property (for example, adding Easting and Northing values to a block reference), select the relevant Property Set Definition and create a new Automatic property. In the ‘Automatic Property Source’ dialog, select the property to add to the definition. In this example, we have added two properties for a block reference (Easting and Northing) These values will be used in a later exercise to determine additional information such as converting to Lat-Long, and Station/Offset.
Formula Property Definitions

This is where we start to create more complex property values, by leveraging some basic VB Scripting.

Some basic programming knowledge is required to get the most out of using Formula Property Definitions, a good resource for learning is located at:

https://www.w3schools.com/ASP/asp_ref_vbscript_functions.asp

It is good practice to begin your formula property definition with a header declaring the output value, and a simple VB expression

On Error Resume Next. This will ensure that if an error occurs, the program will continue to run.

RESULT="--"

On Error Resume Next

<Code goes here>

'Return the results

RESULT = <Output>

The code above initialises a variable called ‘RESULT’, and sets this to a value of ‘--’.

The ‘On Error Resume Next’ is standard VB code that tells the compiler to continue if an error is found.

After body of the code we can return the ‘RESULT’ variable.

In our example, we will create a Property Set Definition called ‘RMS - Asset Information – General’, which is required to provide an overall ‘AssetID’ property. This is essentially a concatenation built of out shape codes (defined in a previous section) and some Manual properties. This definition will be applied to all 3D solid objects in the model.

Previously, the corridor shape codes have been setup with a ‘-‘ separator, and have been named in the following format
We will use a combination of Manual and Formula properties to create a complex name, based on the ‘Long Attribute’ column shown below:

[Family]-[Road Num]-[Road Loc]-[Chainage]-[Attribute ID]-[Category1]-[Category2]
Breaking this attribute down, we can determine which elements are required to create our complex expression:

- **Manual Property**
- **Formula Property**

\[\text{[Family]}-\text{[Road Num]}-\text{[Road Loc]}-\text{[Chainage]}-\text{[Attribute ID]}-\text{[Category1]}-\text{[Category2]}\]

Begin by creating a new Property Set Definition called ‘RMS - Asset Information – General’ and apply it to BODY and Solid(3D) objects.

Create the two manual properties (labelled ‘RMSRoadNo’ and ‘RMSRoadLoc’) as Integer and Text, respectively, and set some default values, as these should be constant for all objects in the model.

For the ‘RMSRoadNo’ asset, apply a custom format depicting 5 padded integers. To create the custom format, you must enter the Style Manager via the command line prompt ‘STYLEMANAGER’ to display the extended options.

Figure 9: Property Data format - 5 padding places
The **Family**, **AttributeID**, **Category1** and **Category2** definitions are to be created as Formula Property Definitions. To extract the correct values, we will use some basic VB string editing functions to extract the relevant data from the auto-generated shape codes.

For example, from the extracted Corridor Solids, the ‘**Corridor Shape Information**’ Property Set contains the entire shape code in the ‘**CodeName**’ definition, with each field separated by a minus ‘—’ sign (defined earlier in this document).

Create a new Formula Property Definition, called ‘**Family**’, and add the following code to the Formula:

RESULT="—"

On Error Resume Next

```
txt = Split("[Corridor Shape Information:CodeName]", "—")
```

‘Return the results

RESULT = CStr(txt(0))

In the above code, the variable ‘RESULT’ is declared and initialised with an arbitrary value ‘—’. The variable ‘txt’ uses the VB Script text function ‘Split’, and takes the previously defined Corridor Shape Code as the input, with the separator defined as a ‘—’. This will return an array of strings. For example, in the image below, we are inputting a string “PLWC-CT1-TML1-08”. As we are splitting the string using a ‘—’ delimiter, the output will consist of an array containing the following items
• “PLWC”
• “CT1”
• “TML1”
• “08”

To access the correct value from the array - in this instance, the “PLWC” value – we will access the first element in the list, which is element 0 (arrays are zero-based in VBScript)

Therefore our output variable ‘RESULT’ will be called as txt(0). The ‘CStr’ method ensures the result is returned as a string (Convert String)

The Sample Result panel (top-right) will return the value from the formula. If the formula is successful, the value will appear. If not, the entire formula will be replicated in the panel

Note that when copying and pasting code into the Formula, any text that will utilise an existing Property Definition MUST be re-input. In other words, delete the text within the square brackets (i.e. [Corridor Shape Information:CodeName]), and replace by double clicking the relevant Property Definition from the panel in the bottom left of the dialog. A light grey background mask will indicate whether a Property Definition is being used, or just hard-coded text.

![Figure 12: Formula Property Definition Dialog](image-url)
**Figure 13: Correct formula in Sample Result**

**Figure 14: Incorrect formula in Sample Result**
To complete the remaining values, simply copy the formula from ‘Family’ into a new formula definition and change the return array value, For example:

- **Family**  
  \[ \text{RESULT} = \text{CStr(txt(0))} \]
- **AttributeId**  
  \[ \text{RESULT} = \text{CStr(txt(1))} \]
- **Category1**  
  \[ \text{RESULT} = \text{CStr(txt(2))} \]
- **Category2**  
  \[ \text{RESULT} = \text{CStr(txt(3))} \]

![Figure 15: Property Set with Manual and Formula Definitions](image)

To extract the ‘Chainage’ value, simply take the auto-generated value – ‘AssemblyStartStation’ - from the ‘Corridor Shape Information’ Property Set as a base. Note the output contains three decimal places and an ‘m’ symbol on the end. For our example, we can use scripting to strip the last 5 characters from the AssemblyStartStation value (3 decimal places, the “m” and the decimal dot point)

\[ \text{RESULT} = "--" \]

On Error Resume Next

' declare variable

startStn = "[Corridor Shape Information:AssemblyStartStation]"

' Station

stlen = CInt(Len(startStn))

stn = Mid(startStn, 1, stlen - 5)

'Return the results

RESULT = CStr(stn)

In the above code, again the variable ‘RESULT’ is declared and initialised with an arbitrary value ‘--’.

The variable ‘startStn’ references the auto-generated value from the corridor solids extraction "[Corridor Shape Information:AssemblyStartStation]"

The variable ‘stlen’ uses the “Len” method to extract the number of characters in the start station value, and then converts this into an integer

Lastly, the variable ‘stn’ uses the “Mid” method to create a new string, starting from the first character, and ending 5 places from the end of the string length
Bring it all together
The final property definition – "AssetId" – is a simple concatenation of the previously extracted data. Concatenating strings in VBScript is achieved through the "&" key. For example:

RESULT="--"
On Error Resume Next
stOut = "[Family]" & "." & "[RMSRoadNo]" & "." & "[RMSRoadLoc]" & "." & "[Chainage]" & "." & "[AttributeId]" & "." & "[Category1]" & "." & "[Category2]"
'Return the results
RESULT = CStr(stOut)

The output string variable ‘stOut’ is contented using the “&” key, and in the above example, has a minus “-“ symbol added as a delimiter.

Figure 16: AssetId Formula using string concatenation
Conditional Statements
We can start to add some more intelligence into the auto-asset tagging process by including some basic decisional processes into the VBScripts

For example, we will setup an external text file containing the details of the pavement, so that when we encounter a specific pavement type (Category1), we can search through a text file to find that specific pavement and report back its associated pavement layer type (Category2). This will add more detailed information, as opposed to simply using the layer number (01, 02, 03 etc)
For example, a text file called ‘D:\Projects\AU 2018 Property Sets\Datasets\Pavement.txt’ is created and contains information about the pavement layers in the following format (comma separated)

<Pavement Name>, <Layer 1>, <Layer 2>, <Layer 3 etc>

RESULT="--"
On Error Resume Next
Set fso = CreateObject("Scripting.FileSystemObject")
Set file = fso.OpenTextFile("D:\Projects\AU 2018 Property Sets\Datasets\Pavement.txt", 1)
Dim arrFileName() ' Empty Array
i = 0
Do While file.AtEndOfStream <> true
    text = file.ReadLine
    ReDim Preserve arrFileName(i)
    arrFileName(i) = text
    i = i + 1
Loop
file.Close
Set file = Nothing
Set fso = Nothing
paveType = "--"
cat1 = "[Category1]"
cat2 = CInt("[Category2]")
' Loop through the array
For i = 0 to UBound(arrFileName)
    ' Split the string rom the text file
    txtArr = Split(arrFileName(i), ",")
    ' Search the first item in the split array - Pavement Type
    Select Case CStr(txtArr(0))
       Case "TML1", "TML2"
          paveType = txtarr(cat2)
    End Select
Next
RESULT = paveType

In the code above, we access the text file through the ‘FileSystemObject’ library, and the subsequent ‘OpenTextFile’ method.

The code reads through each line of text, and stores the values in a variable array ‘arrFileName’.

We then loop through the array, splitting the text line into another array ‘txtArr’ (split by commas). Using the ‘Select Case’ VBScript function will allow us to compare the first value (i.e the Pavement Type name) against the previously coded ‘[Category1]’ property definition. If the name matches, we then set the new mapped pavement layer name from the remainder of the split array to the [Category2] number previously defined in the steps above.
Formula Property Definitions (including COM Interop)

Formula property definitions can also be used to perform some basic interrogation of Civil 3D / AutoCAD object. For example, we can extract information from a Civil 3D Surface, such as Number of Triangles, 2D Area, 3D Area etc.

This technique requires connecting to the Civil 3D Application and retrieving the current object handle through the following three lines of code:

```vba
Set oApp = GetObject(, "AutoCAD.Application")
Set oCivilApp=oApp.GetInterfaceObject("AeccXUiLand.AeccApplication.13.0")
Set obj=oCivilApp.ActiveDocument.HandleToObject("[Handle]")
```

The first line connects us to the current Civil 3D application using the ‘GetObject’ method, whereas the application’s ‘GetInterfaceObject’ method (second line) retrieves the relevant Civil 3D library to be called. In this instance, we are using the ‘AeccXUiLand’ COM Interop library.

The ’13.0’ refers to the version of Civil 3D you are currently running (i.e. 13.0 refers to Civil 3D 2019, 12.0 for Civil 3D 2018 etc.)

A quick way to determine which version to use is to locate the libraries in your installation folder (C:\Program Files\Autodesk\AutoCAD 2019\C3D\ AeccXUiLand.dll). Right click the file name and select ‘Properties’

Under the Details tab, the File Version indicates the library version number to use.
The third line uses the defined Civil 3D application to extract the current object’s handle and declare the variable ‘obj’ to the actual object. We will then use the ‘obj’ variable to extract further information from the Civil 3D object.

Additional Civil 3D libraries can be called to extract difference information. For example:

- AeccXUiPipe.AeccPipeApplication.13.0 Civil 3D Pipes
- AeccXUiRoadway.dll.AeccAplication.13.0 Civil 3D Corridors

For more information on the COM libraries, including the AeccXUiLand library, use the following links:


We will create a new Property Set Definition (AU2018 Surfaces) and apply it to a Civil 3D Tin Surface.
Create a new Automatic Property Definition, and call it ‘Handle’, and select the Handle property. We need Handle value to ensure the object from which we are extracting information from is the same object that has the property set definition attached.

Figure 20: Automatic 'Handle' Property Definition
Create a new Formula Property Definition, call it ‘Area2D’ (or similar).

You can copy and paste code snippets into the Formula area of this dialog. However, in the code where we set the ‘obj’ variable to inherit the actual object handle (“[Handle]”), we are required to manually delete the value within the square brackets i.e. [Handle], and replace this with the actual automatic property ‘Handle’ by double-clicking the ‘Handle’ Property Definition from the ‘Insert Property Definitions’ section of the dialog (lower left corner). You will see that the value in the Formula updates to show a light-grey background, indicating we are now referencing another definition’s value, as opposed to a hard-coded text string.

The following code snippet will extract the 2D area from the surface

```
RESULT = "--"
On Error Resume Next
Set oApp = GetObject(, "AutoCAD.Application")
Set oCivilApp = oApp.GetInterfaceObject("AeccXUiLand.AeccApplication.13.0") ' 2019
Set obj = oCivilApp.ActiveDocument.HandleToObject("[Handle]")
RESULT = obj.Statistics.Area2D & " m2"
```

![Figure 21: Replacing text with a dynamic Property Definition value](image)
Some other values to consider with extracting surface information include:

- RESULT = obj.Statistics.NumberOfTriangles  Number of Triangles
- RESULT = obj.Statistics.NumberOfPoints   Number of Points
- RESULT = obj.Statistics.MaxElevation   Maximum Elevation
- RESULT = obj.Statistics.MinGrade   Minimum Grade
- RESULT = obj.Statistics.RevisionNumber   Surface Revision Number

Figure 22: IAeccTinSurface Interface
Add the property set to the surface object though the Properties Palette as depicted below. Ensure you only have a Civil 3D Tin Surface (or surfaces) selected before applying the property set.
Figure 24: Adding a Property Set to a Civil 3D Surface

Figure 25: Property Set assigned to Civil 3D Tin Surface
Some examples of using the COM Interop include:

### Civil 3D Featurelines

RESULT="--"

On Error Resume Next

Set oApp=GetObject(, "AutoCAD.Application")
Set oCivilApp=oApp.GetInterfaceObject("AeccXUiLand.AeccApplication.13.0")
Set obj=oCivilApp.ActiveDocument.HandleToObject("[Handle]")

'Return the results
RESULT=obj.Length3D

### Civil 3D Surfaces – Extracting levels on surface from known XY

RESULT="--"

On Error Resume Next

Set oApp=GetObject(, "AutoCAD.Application")
Set oCivilApp=oApp.GetInterfaceObject("AeccXUiLand.AeccApplication.13.0") ' Civil 2019
Set obj=oCivilApp.ActiveDocument.HandleToObject("[Handle]")

elev = obj.FindElevationAtXY(180, 1180) ' add coordinate here, or use another objects XY

RESULT = elev

### Civil 3D Pipes

RESULT="--"

On Error Resume Next

Set oApp=GetObject(, "AutoCAD.Application")
Set oCivilApp=oApp.GetInterfaceObject("AeccXUiPipe.AeccPipeApplication.13.0")
Set obj=oCivilApp.ActiveDocument.HandleToObject("[Handle]")

RESULT = obj.StartStructure.Name

### .NET Automation

This is the most complex of stages and involves the development of customised external tools (dll’s, exe’s) using tools such as Microsoft Visual Studio (C# or VB.Net) and the AutoCAD/Civil 3D API to assist in the creation and population of property set definitions and values.

For our project, we are required to apply a single property set “**RMS - Asset Information – General**” to all of the different pavement layers in the model. Additionally, each pavement layer is required to hold it’s own unique property set (i.e. Wearing Course, Sprayed Seal, Subbase etc)
Whilst it is possible to add these unique property sets manually, it will take considerable time to:

1. Select the individual pavement layers
2. Add the correct property set(s) to the layers

A simpler solution was to develop a small .Net program that leverages an external text file (*.txt) to control what property sets are assigned to which object. The format uses a simple comma-delimiter to control the following parameters

<CODE/BLOCK NAME>,<PROPERTY_SET_NAME>,<ASSET_TYPE>, where

- **CODE/BLOCK NAME** is the object layer name / block name
- **PROPERTY_SET_NAME** is the name of the property set to assign
- **ASSET_TYPE** depicts whether the asset to be tagged is a point or linear element (i.e. a block or a swept solid)

To load the program, type ‘NETLOAD’ on the command line, navigate to the .dll program location and click Open.

To run the program, enter ‘PROPERTYSETASSIGN’ on the command line.

![Figure 26: Loading the PropertySets dll](image)

Click on the ‘Select’ button to select the relevant Property Set Configuration File. Once the file is loaded, a preview will appear in the window below, highlighting each row in red if the property set defined in the file does not exist in the model.
Click Apply or OK to finish applying the property sets.
Additionally, through the API, the program detects single point objects from the configuration file, and attempts to automatically populate fields such as Chainage, Offset, Latitude and Longitude.

For example, the Property Set “RMS – LITE Lighting” contains an Automatic Property definition to extract the ‘Easting’ and ‘Northing’ of the block. Through the API, when the program is run, these Easting and Northing values are processed into corresponding ‘StartLatitude’ and ‘StartLongitude’ values.

Similarly, the values of the ‘Chainage’ and ‘Offset’ are calculated through the API by first reading the value populated in the ‘AlignmentRef’ (which corresponds to the relevant alignment string name), and then extracting the chainage and offset values relative to the nominated alignment.

This approach to asset tagging automates many of the repetitive tasks, resulting in a more streamlined workflow and increased time savings.
Figure 29: Light Pole with property set attached

Figure 30: Light Pole with additional data automatically applied through API
Conclusion
Using a combination of in-product tools, customized tools and – most importantly – a well-defined process, it is possible to create a robust and efficient workflow for your construction documentation and exports.

Using a well-structured and documented approach we can build our Civil 3D BIM models to cater for any design scenario and – most importantly – create a workflow that is adaptable to change.

References
Recommended Past Au
http://au.autodesk.com/au-online/classes-on-demand/search?full-text=&productName=AutoCAD+Civil+3D&video-only=on

a. CI123653 – Jowenn Lua
b. CI10903 – Andrew Milford
c. CI4264 – Peter Funk
d. CI3001 and CI4252 – Kati Mercier
e. CI11634 and CS21034 – Eric Cylwik
f. WF21421 – Jerry Bartels and Jeff Bartels

Recommended links for learning
Autodesk Help Videos

Sample VBScript Functions
https://www.w3schools.com/asp/asp_ref_vbscript_functions.asp

Link for the Civil 3D API Reference Guide

Additionally, these are some helpful references for programming with AutoCAD Property Sets

Blog: Civil3dplus – Brian Hailey
https://civil3dplus.wordpress.com/

Blog: From the Ground Up – Andrew Milford, Cristina Savian and Jack Strongitharm
http://fromthegroundup.typepad.com/from_the_ground_up/