

Supplemental Information for MX Users

Note this help should be read in conjunction with the main help on the topic and is only intended as a supplement.

MX Input File Workflows

MX input files form an essential part of current workflows that provide repeatability and efficiency. The Bentley Civil Openroads tools provide for a more interactive dynamic modelling experience that requires rules and relationships that MX input files were never conceived of having to capable of supporting.

With this SELECTseries 3 some MX specific integration has been put in place to allow some interaction between old and new workflows through input files that assist in the transfer of strings and surfaces from MX so they can be included in input files.

In the first instance, data transfer occurs only interactively through the GUI options and multiple imports. The GUI itself preserves last used settings and provides saved settings to assist interactive productivity. However MX also records interactive import process in the MX project log, this can then be applied later via input files to help automate processes from MX.

MXScript

MX uses the major option 'r;script' to suspend the MXengine and allow processing of external / CAD operations. This process was originally implemented to facilitate Final Drawings annotation and page layout as well as attachment of reference files.

Script can be used to call Microstation key-in commands directly using commands and parameters.

Syntax

```
Script, key-in {command}, {parameter}
```

The following example can be used to reset the view extents of the drawing.

```
script,FIT VIEW EXTENDED 1
```

Geometry Commands

Import commands are captured in the MX project log and can be used to publish MX geometry from inoutfiles into the civil model.

```
Geometry import file {parameter}
```

Parameters

```
Model=MX Model
```

```
String=MXString Name
```

```
Maskfile=Mask table in the form "PrivateStyles\Test Mask.msk" (in lieu of String)
```

Imports the specified model from the active MX project file specific.

Example

```
script, geometry file import model=PROPOSED name=GC20
```

Terrain Commands

Import commands are captured in the MX project log. These commands can be utilised in input files to control the initial creation of Civil Terrains from MX models.

```
Terrainmodel import file {parameter}=
```

Parameters

```
File=Model.fil (path is not required for active project)
```

```
Model=MX Model in File
```

```
Edgemethod=NoRemove / RemoveMaxSide / Slivers
```

```
Maxsidelength=linear length ie 50
```

```
FeatureDefinition=Defined surface feature definition
```

Imports the model specified from the active MX project file specific.

Example

```
script, terrainmodel import file=model.fil model=DESIGN
edgemethod=NoRemove
```

When passing information forward to the active dgn model from MX, it advisable to ensure that a previous instance is cleared. This can be likened in MX to the use delete create models in inputfile. To facilitate updating of existing Civil Models, Civil Objects can be defined to act as containers for the many strings that can be passed and make selection / updating and deletion easier.

Note These options are only available through key-in syntax.

Example

The following is a simple example that manages the civil object and imports data into it from the current model.fil

```
script,GEOMETRY civilobject delete=MX GROUND MODEL
```

```
script,GEOMETRY civilobject create=MX GROUND MODEL
```

```
script,GEOMETRY civilobject setactiveobject=MX GROUND MODEL
```

```
script,TERRAINMODEL import model=GROUND MODEL edgemethod=RemoveMaxSide
maxsidelength=50 featuredefinition=Large Terrain
```

```
script,GEOMETRY civilobject closeactiveobject
```

Civil Object General Commands

```
Civilobject {parameter} =
```

Parameters

```
New={name}
```

Creates a new civil object from the supplied file.

Delete={name}

Deletes the supplied named civil object.

Remove={name}

Sets the supplied civil object active and removes subsequently supplied entities. May not be required if 'r;open' state could be achieved

SetActiveobject={name}

Sets the supplied civil object active and ready for write / deletes of subsequently supplied entities.

Closeactiveobject={name}

Closes the active civil objectDeletes subsequently supplied entities

Note: Civil objects are not supported interactively, they are visible in the Civil Model inside Project Explorer.

Export Civil Geometry to MX

During the export of geometry to MX (manual or automatic) only the 'r;Active' civil profile is exported to MX. The respective M & G Strings are automatically created in conjunction with the corresponding linemode for horizontal (Halign) and vertical (Valgn / Verat). The respective linemode code is written to the MX log file and associated ASCII files are created in ..\mxdata folder. These input files can then be incorporated back into the existing input file process.

MX Feature Definitions

This section describes additional functionality implemented specifically for MX and needs to be considered with the main help on this subject.

MX Stylesets were designed for a single display case. Civil Feature definitions have been set up to allow different display styles for the same feature in different use cases and this provides a significant benefit in controlling plan, profile and 3D representations.

By default when importing plan MX Stylesets the native pss style is defined for each feature definition available. There is a single exception for string features in that the intersecting feature definitions for profile and section require a point style. This point style is by default linked to an Element template that contains a symbol to make it visible as an intersection in profile. Element template is defined as MX Default\Intersection Point using MSL3. It is created automatically when importing the styleset and is stored in the same file as the styleset feature definitions so as to make it available with the feature.

Linking PSS

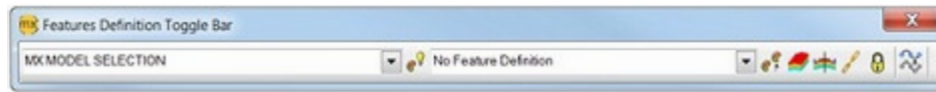
When linking a pss, MX automatically creates a category named the same as the pss and imports into a flat structure inside the container category. Feature definitions are created for point strings and surfaces based on the MX string type.

Importing Long Section, Cross section stylesets creates both a string and surface definitions for each feature. This allows the use of the feature to point to a different style depending on the use case.

Element Templates are automatically created when linking Long Section, Cross section and Triangulation stylesets and all surface definitions point to the Element Template.

Feature Definition Toggle Bar

The feature definition toolbar used in MX contains additional controls required to aid MX users in the creation of MX String Models.



The first control is a MX Model listing. Any MX string models in the current project will be available from the dropdown menu. With a string model defined, only the linear features available for the models associated styleset are available, this functionality is propagated across all commands when a active MX model is set on the feature Toolbar.



Associate MX Strings allows the projection of points from a reference onto adjacent strings. This ensure that the all the base elements points are included and align as would normally occur when using MX Design options.

Parent topic: [Getting Started](#)