

# Bentley MicroStation Design Reader/Writer

The Design Reader and Writer modules provide the Feature Manipulation Engine (FME) with access to files used by the MicroStation and Intergraph Interactive Graphics Design System (IGDS). Intergraph has made public the specification for this file format, which they call the Intergraph Standard File Format (ISFF)<sup>1</sup>. This chapter assumes familiarity with this format.

## Overview

Design files consist of a header, followed by a series of *elements*. The header contains global information including the transformation equation from design units to user coordinates, as well as the dimension of the elements in the file. Each element contains standard display information, such as its color, level, class, and style, as well as a number of attributes specific to its element type. For example, a text element has fields for font, size, and the text string in addition to the standard display attributes.

---

**Tip:** The IGDS reader and writer modules support both two- and three-dimensional Design files and cell libraries.

---

Individual design file elements must be less than a system-imposed maximum number of bytes. *Complex elements* solve this problem by physically grouping individual elements together into an object that will be manipulated as a whole. The FME transparently handles such complex elements as single FME features. This situation occurs when text elements are grouped together into a single complex element headed up by a text node, and when linear or polygonal features have more than 101 vertices (Microstation V7) or 5000 vertices (Microstation V8). *Cells* are complex elements used as symbols, and are treated as atomic entities by the FME.

Each IGDS file element may have one or more *attribute linkages* associated with it. The IGDS reader and writer support both user data and database linkages. (Note, however, that the DGN V8 reader and writer do not support the interpretation of user linkages. Database linkages and MSLINKS are supported.) The linkage values may be used to join elements with attributes stored in relational tables through the use of the @Relate FME Transformation Function. Linkages may also be used to specify a fill color for IGDS Shape elements, and other application specific data. (Note, however, that the `igds_fill_color` attribute will override any fill color linkage specification if both are present.)

Because Design files support three interpretations of units, the IGDS reader and writer must be told how to interpret the feature coordinate units and how they will be converted to and from Units of Resolution (UORs). The feature coordinate units may be interpreted as Master Units, SubUnits, or as raw UORs, depending on the setting of `IGDS_UNITS` in the mapping file. However, when writing to DGN V8 files, the writer ignores these settings from the mapping file and adopts the settings as read from the seed file chosen. This means that if you want to do something special with the working units, you have to do that in the V8 seed file.

---

1. Throughout this chapter, the terms *IGDS file* and *Design file* are used interchangeably to refer to the ISFF format.

The IGDS reader and writer use symbolic names for the IGDS element types rather than the IGDS numeric values. This greatly simplifies element type specification. The following table maps the IGDS element type number to its corresponding FME feature `igds_type` attribute value that is used by the IGDS reader and writer. Subsequent subsections describe the handling of each of these element types in detail.

<b>IGDS Element Type</b>	<b>FME <code>igds_type</code></b>
2	<code>igds_cell</code>
3	<code>igds_point</code>
3,4,12	<code>igds_line</code>
6,14	<code>igds_shape</code>
7	<code>igds_text_node</code>
11,12	<code>igds_curve</code>
12	<code>igds_complex_string</code>
14	<code>igds_complex_shape</code>
15	<code>igds_ellipse</code>
16	<code>igds_arc</code>
17	<code>igds_text</code>
7,17	<code>igds_multi_text</code>
2,6,14	<code>igds_solid</code>
34,35	<code>igds_shared_cell</code>
19	<code>igds_3d_solid</code>

The Reader/Writer has been enhanced to support enhanced geometry. When features are read/written using enhanced geometry then all the complex chains, complex shapes and solids (unnamed cells) will preserve arcs and ellipses within them.

## Design File Quick Facts

Format Type Identifier	IGDS
Reader/Writer	Version 7 Both Version 8 Both
Licensing Level	Base
Dependencies	None
Dataset Type	File
Feature Type	Level number
Typical File Extensions	.dgn
Automated Translation Support	Yes
User-Defined Attributes	No
Coordinate System Support	No
Generic Color Support	Yes
Spatial Index	Never
Schema Required	No
Transaction Support	No
Enhanced Geometry	Yes
Geometry Type Attribute	igds_type

Geometry Support			
Geometry	Supported?	Geometry	Supported?
aggregate	no	point	yes
circles	yes	polygon	yes
circular arc	yes	raster	no
donut polygon	yes	solid	yes
elliptical arc	yes	surface	no
ellipses	yes	text	yes
line	yes	z values	yes
none	no		

## Reader Overview

The FME reader detects the version of the source dataset (version 7 or 8) internally and handles it accordingly. There is no difference to users in terms of the reader key-word or attribute names of the elements.

The IGDS reader first reads the header information from the Design file being processed, and extracts the conversion parameters required to translate coordinates from internal IGDS UORs to ground units. It also determines the dimension of the input file.

It then extracts each individual element, one at a time, and passes it on to the rest of the FME for processing. Complex elements are extracted as single FME features. If a complex element contains an arc, then the reader automatically converts it to a linestring enabling it to be processed by all other readers and writers within the FME.

If the element had any attribute linkages attached to it, these are read and added as attributes to the FME feature being created.

When the IGDS reader encounters an element type it does not know how to process, it simply ignores it and moves on to read the next element.

DGN Version 8 also reads the models to which the features belong. All the models read retain their respective working units and global origin values.

## Reader Directives

The IGDS reader processes the <ReaderKeyword>\_DATASET directive in the mapping file. The value for this directive is the file name of the IGDS file to be read. By default, the <ReaderKeyword> for the IGDS reader is IGDS, so a typical mapping file fragment specifying an input IGDS file looks like:

```
IGDS_DATASET /usr/data/dgn/92b034.dgn
```

The IGDS reader also processes the <ReaderKeyword>\_UNITS directive in the mapping file. This directive controls the conversion between UORs in the Design file and FME coordinates. There are three possibilities, outlined in the table below. If no UNITS directive is specified, then IGDS\_SUB\_UNITS is assumed.

IGDS_UNITS Value	Description
IGDS_MASTER_UNITS	The UORs read from the Design file are converted into <b>master units</b> , according to the conversion factor read from the Design file header, before being stored in an FME feature.
IGDS_SUB_UNITS	The UORs read from the Design file are converted into <b>sub-units</b> , according to the conversion factor read from the Design file header, before being stored in an FME feature. <b>This is the default.</b>
IGDS_UORS	The UORs read from the Design file are stored directly in an FME feature with no conversion.

The IGDS reader processes several other directives in the mapping file, as shown below. These enable the FME to override the Global Origin and Scaling information. The first four directives are normally used only when reading Design files that have bad header information. If the FME detects a difference between these settings and those read from the Design file, a warning is output to the log file and these settings prevail.

The IGDS reader can also be configured to output all the elements composing cells, or symbols. This is useful if the graphical representation of the Design file is to be preserved. This is true when, for example, a Design file is translated to a GIF image.

### UOR\_SCALE

**Required/Optional:** *Optional*

The number of ground units per UOR.

**Workbench Parameter:** <WorkbenchParameter>

**UOR\_GLOBAL\_ORIGIN\_X****Required/Optional:** *Optional*

The global origin of x measured in UORs.

**Workbench Parameter:** [<WorkbenchParameter>](#)**UOR\_GLOBAL\_ORIGIN\_Y****Required/Optional:** *Optional*

The global origin of y measured in UORs.

**Workbench Parameter:** [<WorkbenchParameter>](#)**UOR\_GLOBAL\_ORIGIN\_Z****Required/Optional:** *Optional*

The global origin of z measured in UORs.

**Workbench Parameter:** [<WorkbenchParameter>](#)**SUBS\_PER\_MASTER****Required/Optional:** *Optional*

The number of sub units per master unit. This is only used if UOR\_SCALE is not present.

**Workbench Parameter:** [<WorkbenchParameter>](#)**UORS\_PER\_SUB****Required/Optional:** *Optional*

The number of UORs per sub unit. This is only used if UOR\_SCALE is not present.

**Workbench Parameter:** [<WorkbenchParameter>](#)**EXPAND\_CELLS****Required/Optional:** *Optional*

Controls whether or not all components of a cell will be output by the reader.

If the value is YES, then they are and the cell header itself is not output.

If it is NO, then only the cell header is output.

**Values:** YES | NO**Default Value:** NO**Workbench Parameter:** [<WorkbenchParameter>](#)**EXPAND\_UNNAMED\_CELLS****Required/Optional:** *Optional*

This directive should not be confused with `EXPAND_CELL` in terms of its usage. It is better understood in relation to `igds_solid`. When it is set to `YES`, then no donuts are formed even if they existed and the cell members retain their colors. When it is set to `NO`, then donuts will be formed if they existed, and the pieces may lose their original colors.

**Values:** `YES` | `NO`

**Default Value:** `NO`

**Workbench Parameter:** [<WorkbenchParameter>](#)

### **PRESERVE\_CELL\_INSERTS**

**Required/Optional:** *Optional*

When `EXPAND_CELLS` is `YES`, this directive controls whether or not the insert points of the cells are also output.

If the value is `YES`, then the cell insert points are output as `igds_cell` features in addition to the cell components.

If it is `NO`, then only the cell components are output.

**Values:** `YES` | `NO`

**Default Value:** `NO`

**Workbench Parameter:** [<WorkbenchParameter>](#)

### **PRESERVE\_UNNAMEDCELL\_INSERTS**

**Required/Optional:** *Optional*

If the value is `YES`, then the cell insert points are output in addition to the cell components.

If it is `NO`, then only the cell components are output.

**Values:** `YES` | `NO`

**Default Value:** `NO`

**Workbench Parameter:** [<WorkbenchParameter>](#)

### **PROPAGATE\_CHAIN\_ELEMENT\_LINKAGES (applicable only with classic geometry)**

**Required/Optional:** *Optional*

Controls how the linkages attached to complex chain element features are handled.

If the value is `YES`, then the linkages attached to the first component of the complex chain are returned on the FME feature, supplementing any existing linkages.

If it is `NO`, then any linkages on the component elements themselves will be lost and only those linkages attached to the complex chain itself will be returned.

**Values:** `YES` | `NO`

**Default Value:** *NO*

**Workbench Parameter:** [<WorkbenchParameter>](#)

### **SPLIT\_COMPLEX\_CHAINS (applicable only with classic geometry)**

**Required/Optional:** *Optional*

Controls whether or not complex chain elements are returned, merged as a single linear feature or as their component parts.

If `SPLIT_COMPLEX_CHAINS` is `YES`, then FME adds the attribute `igds_chain_number` which is added to each element of a chain split. If desired, this can later be used to aggregate chain elements.

If the value is `YES`, then each component of a complex chain will be returned as its own feature and no feature will be returned for the complex chain as a whole. This is equivalent to dropping the complex chain in MicroStation. If the header had any linkage attributes, these will be propagated to the component elements.

If the value is `NO`, then all elements of the complex chain will be merged into a single linear feature, any arcs in the complex chain will be converted into linestrings and any linkages on the component elements themselves will be lost.

**Values:** *YES | NO*

**Default Value:** *NO*

**Workbench Parameter:** [<WorkbenchParameter>](#)

### **AGGREGATE\_COMPLEX\_CHAINS (applicable only with classic geometry)**

**Required/Optional:** *Optional*

Controls whether or not complex chain element features are returned as aggregates. If the value is `YES` then the individual element properties are held in the `igds_complex_elements{}` list, and the `igds_type` is set to `igds_complex_string` or `igds_complex_shape`. If specified, this setting takes precedence over `SPLIT_COMPLEX_CHAINS`. In other words, if the value of `AGGREGATE_COMPLEX_CHAINS` is `YES`, any value specified for `SPLIT_COMPLEX_CHAINS` is ignored.

**Values:** *YES | NO*

**Default Value:** *NO*

**Workbench Parameter:** [<WorkbenchParameter>](#)

### **TAGS\_AS\_TEXT**

**Required/Optional:** *Optional*

Controls whether or not visible tag data elements are output as separate text elements, in addition to having their data attached to the primary graphic element they go with.

**Values:** *YES | NO*

**Default Value:** *NO* (visible tag data elements are not output as text elements)

[Workbench Parameter: <WorkbenchParameter>](#)

### **PRESERVE\_CURVES**

**Required/Optional:** *Optional*

Controls whether or not curve elements will be stroked into lines by adding vertices.

**Values:** YES | NO

**Default Value:** NO (curves are not preserved and are stroked into lines)

[Workbench Parameter: <WorkbenchParameter>](#)

### **ELEVATION\_SHIFT\_FACTOR**

**Required/Optional:** *Optional*

If an elevation shift is desired to build "fake" 3D topology, this is the scaling factor used to generate the shift. Specifically, the Z value is divided by this factor and the result is added to the X value.

[Workbench Parameter: <WorkbenchParameter>](#)

### **CURVE\_VERTICES**

**Required/Optional:** *Optional*

This is used only when PRESERVE\_CURVES is NO. It controls the number of interpolated points per segment when the curve is stroked into a line.

**Default Value:** 5

[Workbench Parameter: <WorkbenchParameter>](#)

### **TRIM\_DOWN\_TAGS**

**Required/Optional:** *Optional*

Removes the tag attributes when set to *yes*.

**Values:** YES | NO

**Default Value:** NO

[Workbench Parameter: <WorkbenchParameter>](#)

### **SPLIT\_MULTITEXT**

**Required/Optional:** *Optional*

When set to *yes*, the reader splits the multi-text into text nodes and outputs the member text elements as individual text elements. When set to *no*, the text elements are not split.

**Values:** YES | NO

**Default Value:** YES

[Workbench Parameter: <WorkbenchParameter>](#)

**PRESERVE\_UNNAMEDCELL\_INSERTS****Required/Optional:** *Optional*When set to `true`, it outputs the unnamed cell as a point.**Values:** `TRUE` | `FALSE`**Default Value:** `FALSE`**Workbench Parameter:** [<WorkbenchParameter>](#)**READ\_BYTE\_OFFSET****Required/Optional:** *Optional***Version:** *supported for version 7 only*If set to `YES`, adds the `igds_element_byteoffset` attribute (which contains the position of the element in the `.dgn` file) to the feature. Note, however, that turning this option on might significantly slow down reading on some platforms like UNIX.**Values:** `YES` | `NO`**Default Value:** `NO`**Workbench Parameter:** [<WorkbenchParameter>](#)**EXPLODE\_DIMENSION\_ELEM****Required/Optional:** *Optional*If set to `YES`, explodes the dimension element into its pieces. If set to `NO`, then imports the dimension element as an aggregate. When importing as an aggregate, the text members are not output as features but are stored as list attributes of the dimension, and the arc members are stroked.**Values:** `YES` | `NO`**Default Value:** `YES`**Workbench Parameter:** [<WorkbenchParameter>](#)**READ\_XREF\_FILES****Required/Optional:** *Optional*If set to `YES`, reads all the external reference files attached to the source data set. If the reference file has nested references then they are also imported.**Values:** `YES` | `NO`**Default Value:** `NO`**READ\_XREF\_UPTO\_FIRST\_LVL****Required/Optional:** *Optional*If set to `YES`, reads all the external reference files attached to the source data set upto the first level of nesting only. This keyword is honoured only if `READ_XREF_FILES` is set to `yes`.

**Values:** YES | NO

**Default Value:** NO

**Workbench Parameter:** [<WorkbenchParameter>](#)

### USE\_XREF\_PARENT\_MODEL

**Required/Optional:** *Optional*

**Version:** *applicable to version 8 only, since models are supported in version 8 but not in version 7)*

If set to YES, uses the model of the parent file of the xref file.

**Values:** YES | NO

**Default Value:** YES

**Workbench Parameter:** [<WorkbenchParameter>](#)

### EXPLODE\_MULTI\_LINE

**Required/Optional:** *Optional*

If set to yes, then multilines are exploded into its pieces.

**Values:** YES | NO

**Default Value:** NO

**Workbench Parameter:** [<WorkbenchParameter>](#)

### READ\_DELETED\_ELEMENTS

**Required/Optional:** *Optional*

This directive is used to read deleted elements. **Note:** This directive will not be made available in the settings box. To use this directive, it has to be set to TRUE in the mapping file.

**Values:** TRUE | FALSE

**Default Value:** FALSE

**Workbench Parameter:** [<WorkbenchParameter>](#)

### READ\_BYTE\_OFFSET

**Required/Optional:** *Optional*

This directive is added to read byte offset of each element. **Note:** This directive will not be made available in the settings box. To use this directive, it has to be set to TRUE in the mapping file.

**Values:** TRUE | FALSE

**Default Value:** FALSE

**Workbench Parameter:** [<WorkbenchParameter>](#)

## APPLY\_WORLD\_FILE

**Required/Optional:** *Optional*

Use this directive when you have an ESRI World file (\*.wld) that you want FME to use when determining the coordinates for features in your dataset. When this directive has a value of YES FME will search the directory of the dataset for a file with the same name as your dataset but with a .wld extension. If it cannot find a file with that name it will then look for the file "esri\_cad.wld" within the dataset directory. If either of those files exist then FME will use the information in the files to translate the coordinates of the features in the dataset to their new geospatial coordinates. If the files cannot be found then the translation will continue, using the coordinate information found in the dataset, without performing any additional transformation.

**Values:** YES | NO

**Default Value:** NO

**Default Workbench Value:** YES

**Workbench Parameter:** [<WorkbenchParameter>](#)REMOVE\_DUPLICATES

**Required/Optional:** *Optional*

Set this directive to Yes when it is intended to remove the duplicate points (same x and y coordinates) from the geometry of the feature.

**Values:** YES | NO

**Default Value:** YES

**Default Workbench Value:** NO

## Reader Directives for FME Objects

### SCHEMA\_INCLUDE\_MSLINKS

**Required/Optional:** *Optional*

This directive can be used for FME Objects only. When set to YES, schema for MSLINKS are added to the feature.

**Values:** YES | NO

**Default Value:** NO

### SCHEMA\_INCLUDE\_FRAMME

**Required/Optional:** *Optional*

This directive can be used for FME Objects only. When set to YES, schema for FRAMME linkages are added to the feature.

**Values:** YES | NO

**Default Value:** NO

## Writer Overview

To create a new Design file, header information is obtained from an existing Design file, called a *seed file*. The IGDS writer first copies the seed file's header information (including type 68 FRAMME elements for V7 only) to the destination file, and then extracts the conversion parameters required to translate coordinates from feature coordinate units to internal IGDS UORs<sup>1</sup>. The IGDS writer uses the seed file to determine whether the destination file will be two-dimensional or three-dimensional.

Because seed files with a sufficient ground range and resolution may be difficult to obtain, the IGDS V7 writer allows seed parameters to be overridden in the mapping file. When a seed file with insufficient range available is used, the IGDS V7 writer will report that features were outside of the bounds of the seed file, and suggest values for the global origin and UOR/subunit/master unit ratios to use. The FME can also automatically adjust the V7 Design file by setting the `COMPUTE_SEED_FILE_PARAMS` flag to `yes`. Note that this facility has been taken away from the V8 writer – it is no longer necessary since V8 has a much larger design plane than V7.

---

**Note:** When translating from DGN version 8 to DGN version 7 or vice versa in FME Workbench, by default a v8 seed file is chosen from the set of seed files as provided by FME. This has to be changed to an appropriate version 7 or version 8 seed file in order to achieve a successful conversion. The seed file is used to determine which version the user intends to write. Also note that if the user picked a v7 seed file at the time of generating the workspace, the same workspace can be used to write to v7 or v8 by changing the seed file accordingly. But if a workspace was initially generated to write to v8, then it cannot be used to write to v7.

---

A cell library file may optionally be used by both V7 and V8 writer. Cell libraries contain named symbol definitions which can be used to depict point features. If a cell library is specified, the IGDS writer reads in all the cell definitions for later when cell features are output. The IGDS writer can use either 2- or 3-dimensional cell libraries, and will automatically convert the cell definitions to be of the correct dimension for output.

The IGDS writer then outputs each FME feature it is given. Most often, a single FME feature corresponds to a single IGDS element. If any linkages are specified for the element, they are also output. However, some of the IGDS element types cause several elements to be output as a complex unit, with the complex bit turned on. This occurs when a multi-line text object, a cell, or a closed shape or linear feature with more than 101 coordinates (5000 coordinates in V8) is output. The IGDS writer hides all of the details of complex element output.

The IGDS writer can be configured to do one of two things with linear features that have exactly two points. By default, a type 4 linestring will be created for such features. However, if `IGDS_CREATE_LINE_ELEMENTS` is set to `yes` in the mapping file, then a type 3 line element will be created for the two-point linear feature.

---

**Note:** Design files (V7) can be a maximum of 32 MB in size. Files larger than this will not be completely read by Microstation. The IGDS writer will automatically split any design file it is writing into pieces to avoid overrunning this maximum size. When this happens, features that would have caused the size limit to be exceeded are written to additional design files as necessary. The additional files are named `<base-name>_#.dgn`, where # starts at 1 and increases.

---

1. Since coordinates in Design files are ultimately stored as integer UORs, it is possible for precision to be lost or overflow to occur when they are output. Care must be taken to ensure that the conversion parameters in the seed file preserve the data precision and range.

## Writer Directives

By default, the <WriterKeyword> for the IGDS writer is IGDS, so a typical mapping file fragment configuring the IGDS writer would be:

```
IGDS_DATASET /usr/data/dgn/92b034.dgn
IGDS_SEED_FILE /usr/data/dgn/2dseed.dgn
IGDS_CELL_LIBRARY /usr/data/dgn/cartog.cel
```

### DATASET

**Required/Optional:** *Required*

The file name of the output IGDS File.

### SEED\_FILE

**Required/Optional:** *Required*

The file name of the Design file which will be used to *seed* the output file's header information. The default seed file is:

```
$(FME_HOME)/design/seed3d_m_v8.dgn
```

which is a V8 file. To write to V7, you will have to select a valid V7 seed file.

It is important to note that the seed file determines which destination version to write.

**Workbench Parameter:** [<WorkbenchParameter>](#)

### ALLOW\_FILL

**Required/Optional:** *Optional*

Controls whether or not fill linkages will be written out for ellipses, shapes, and solids.

**Values:** YES | NO

**Default Value:** YES

**Workbench Parameter:** [<WorkbenchParameter>](#)

### CELL\_LIBRARY

**Required/Optional:** *Optional*

The file name of an IGDS cell library, which contains the definitions of cells which may later be output.

**Workbench Parameter:** [<WorkbenchParameter>](#)

### DEFAULT\_CELL\_NAME

**Required/Optional:** *Optional*

**Version:** *not currently supported by the V8 writer*

The default cell used in place of any cells requested but not found in the cell library.

## UNITS

**Required/Optional:** *Optional*

**Version:** *not currently supported by the V8 writer*

Specifies how FME feature coordinates will be interpreted and converted into UORs. See the documentation under the heading *IGDS Reader* for details.

## CREATE\_LINE\_ELEMENTS

**Required/Optional:** *Optional*

Controls whether or not type 3 line elements will be created for two point linear features. Valid values are YES or NO. If set to NO, then type 4 elements will be created.

**Values:** YES | NO

**Default Value:** NO

## COMPUTE\_SEED\_FILE\_PARMS

**Required/Optional:** *Optional*

**Version:** *Not currently supported by the V8 writer – this directive is ignored if chosen with a V8 seed file.*

Automatically adjusts the origin and scaling of the seed file to provide an optimum set of parameters for the input data.

## UOR\_GLOBAL\_ORIGIN\_X

**Required/Optional:** *Optional*

**Version:** *Not currently supported by the V8 writer*

The global origin of x, measured in UORs. Overrides that read from the seed file.

## UOR\_GLOBAL\_ORIGIN\_Y

**Required/Optional:** *Optional*

**Version:** *Not currently supported by the V8 writer*

The global origin of y, measured in UORs. Overrides that read from the seed file.

## UOR\_GLOBAL\_ORIGIN\_Z

**Required/Optional:** *Optional*

**Version:** *Not currently supported by the V8 writer*

The global origin of z, measured in UORs. Overrides that read from the seed file.

## MASTER\_UNIT\_NAME

**Required/Optional:** *Optional*

**Version:** *Not currently supported by the V8 writer*

The two-character master unit name to use. Overrides that read from the seed file.

### **SUB\_UNIT\_NAME**

**Required/Optional:** *Optional*

**Version:** *Not currently supported by the V8 writer*

The two-character sub unit name to use. Overrides that read from the seed file.

### **SUBS\_PER\_MASTER**

**Required/Optional:** *Optional*

**Version:** *Not currently supported by the V8 writer*

The number of sub units per master unit. Overrides that read from the seed file.

### **UORS\_PER\_SUB**

**Required/Optional:** *Optional*

**Version:** *Not currently supported by the V8 writer*

The number of UORs per sub unit. Overrides that read from the seed file.

### **MANGLE\_DBCS\_TEXT**

**Required/Optional:** *Optional*

**Version:** *Not currently supported by the V8 writer*

Controls whether or not double-byte-character-set text is *mangled* when text strings are written. MicroStation uses special header bytes to single DBCS text. If this directive is set to `yes` in the mapping file, then these special bytes will be output when a DBCS text string is encountered. The default value is `no`. Note that the IGDS reader automatically de-mangles DBCS text.

### **SPLIT\_BIG\_DGN7\_FILES**

**Required/Optional:** *Optional*

**Version:** *Applies to V7 writer only*

Allows user to split Version 7 DGN files bigger than 32 MB. Note that this keyword can be manually set to `no` in the mapping file.

**Values:** `YES` | `NO`

**Default Value:** `YES`

### **SPLIT\_SIZE\_DGN7\_FILES**

**Required/Optional:** *Optional*

**Version:** *Applies to V7 writer only*

Allows user to set the size of the output file in MB in case it is supposed to be split. It is applicable only if `SPLIT_BIG_DGN7_FILES` is set to `true`.

**Default Value:** 32 MB

## WRITE\_TAGS

**Required/Optional:** *Optional*

**Version:** *currently supported by the V8 writer only*

Controls whether or not tags should be written for the elements which have necessary tag information attached to them as attributes.

**Default Value:** yes

## Feature Representation

In addition to the generic FME feature attributes that FME Workbench adds to all features (see *About Feature Attributes*), this format adds the format-specific attributes described in this section.

Special FME feature attributes are used to hold IGDS element parameters. The IGDS writer will use these attribute values as it fills in an element structure during output. The IGDS reader will set these attributes in the FME feature it creates for each element it reads.

The FME considers the IGDS level to be the *FME feature type* of an IGDS feature. Each IGDS element, regardless of its geometry type, shares a number of other parameters, as described in the following table. Subsequent subsections describe parameters specific to each of the supported element types.

When writing elements, `igds_type` has precedence over the `igds_element_type`, unless there is more than one element type for a given type. For example, for `igds_line` the `igds_element_type` can be used to force the element to be a type 4 line element, even if there are only 2 vertices on the line (that is, by rights it should be a type 3 element).

Attribute Name	Contents
<code>igds_basename</code>	The base filename (without extension) of the design file the elements were read from. This attribute is ignored by the writer. <b>Range:</b> 0..254
<code>igds_color</code>	The element's color setting. This is the element's color index into the color table stored in the design file. This attribute will be overridden by the <code>igds_symbology</code> value. <b>Range:</b> 0..254 <b>Default:</b> 0
<code>igds_color.red</code>	The element's red color intensity, as determined by looking up the element's color index in the color table. Reader only. <b>Range:</b> 0..254
<code>igds_color.green</code>	The element's green color intensity, as determined by looking up the element's color index in the color table. Reader only. <b>Range:</b> 0..254

Attribute Name	Contents
igds_color.blue	The element's blue color intensity, as determined by looking up the element's color index in the color table. Reader only. <b>Range:</b> 0..254
igds_class	The element's class. <b>Range:</b> 0..15 <b>Default:</b> 0
igds_element_type	The numeric Design file element type code of the element. When writing to a Design file, the <code>igds_type</code> field overrides this attribute. This attribute will be overridden by the <code>igds_type</code> value. <b>Range:</b> See the <i>Overview</i> subsection. <b>Default:</b> No default
igds_graphic_group	The element's graphic group number. <b>Range:</b> 0..65535 <b>Default:</b> 0 <b>Tip:</b> By using a common value for graphic group value, several otherwise separate elements may be tied together into a <i>logical</i> super-element for later processing by application programs.
igds_hole	If present, it sets the "hole" bit on the element it is creating. Writer only. <b>Range:</b> string <b>Default:</b> No default
igds_level	The IGDS level of the feature. The value of this attribute is the same as the feature type. The Writer will use the value of this attribute if the feature's type cannot be converted into a valid IGDS level. <b>Range:</b> 0..64 (There is no upper limit on levels for Version 8 DGN files.) <b>Default:</b> No default
igds_level_comment	The comment associated with the level from which the element originated. Reader only. <b>Range:</b> String <b>Default:</b> No default
igds_level_group_id	The group identification of the level from which the element originated. Reader only. (Does not exist for Version 8 DGN files.) <b>Range:</b> String <b>Default:</b> No default
igds_level_name	The name of the level from which the element originated. Reader only. <b>Range:</b> String <b>Default:</b> No default
igds_color_set_bylevel	Set to <code>yes</code> if the element's color is set by level; otherwise it is set to <code>no</code> . If it is set to <code>yes</code> , the writer sets the element's property to pick the color from the level it is on. ( <b>Note:</b> Applies to Version 8 DGN files only.) <b>Range:</b> yes/no <b>Default:</b> No default

Attribute Name	Contents
igds_style_set_bylevel	Set to <i>yes</i> if the element's style is set by level; otherwise it is set to <i>no</i> . If it is set to <i>yes</i> , the writer sets the element's property to pick the style from the level it is on. ( <b>Note:</b> Applies to Version 8 DGN files only.) <b>Range:</b> yes/no <b>Default:</b> No default
igds_weight_set_bylevel	Set to <i>yes</i> if the element's weight is set by level; otherwise it is set to <i>no</i> . If it is set to <i>yes</i> , the writer sets the element's property to pick the weight from the level it is on. ( <b>Note:</b> Applies to Version 8 DGN files only.) <b>Range:</b> yes/no <b>Default:</b> No default
igds_snappable	The element's snappability. <b>Range:</b> yes or no <b>Default:</b> yes
igds_style	The element's line style. This attribute will be overridden by the <i>igds_symbology</i> value. <b>Range:</b> 0..7 <b>Default:</b> 0
igds_symbology	A single integer encoding the element's style, weight, and color according to this formula: $\text{symbology} = \text{style} + 8 * \text{weight} + 256 * \text{color}$ This attribute will override the individual settings for style, weight, and type if it is specified. <b>Range:</b> 0..65536 <b>Default:</b> None
igds_type	The FME name for the type of element this feature represents. <b>Range:</b> See the table in the Overview subsection. <b>Default:</b> No default
igds_weight	The element's line weight. This attribute will be overridden by the <i>igds_symbology</i> value. <b>Range:</b> 0..31 <b>Default:</b> 0
igds_xlow	The element's minimum X value in ground units. The value of this attribute is ignored when writing. <b>Range:</b> Any Number
igds_xhigh	The element's maximum X value in ground units. The value of this attribute is ignored when writing. <b>Range:</b> Any Number
igds_ylow	The element's minimum Y value in ground units. The value of this attribute is ignored when writing. <b>Range:</b> Any Number
igds_yhigh	The element's maximum Y value in ground units. The value of this attribute is ignored when writing. <b>Range:</b> Any Number

Attribute Name	Contents
igds_zlow	The element's minimum Z (elevation) value in ground units. The value of this attribute is ignored when writing 3D files. <b>Range:</b> Any Number <b>Default:</b> No default
igds_zlow_uor	The element's minimum Z (elevation) value in UORs. The value of this attribute takes precedence over <code>igds_zlow</code> when the feature is written. The value of this attribute is ignored when writing 3D files. <b>Range:</b> Any Number <b>Default:</b> No default
igds_zhigh	The element's maximum Z (elevation) value in ground units. The value of this attribute is ignored when writing 3D files. <b>Range:</b> Any Number <b>Default:</b> No default
igds_zhigh_uor	The element's maximum Z (elevation) value in UORs. The value of this attribute takes precedence over <code>igds_zhigh</code> when the feature is written. The value of this attribute is ignored when writing 3D files. <b>Range:</b> Any Number <b>Default:</b> No default
igds_custom_ linestyle	If an element has a custom line style, then this attribute will contain the name of the custom line style. It does not appear as part of the attributes of the element in case it does not have any custom line styles defined for it. <b>Range:</b> String <b>Default:</b> No default
igds_custom_ linestyle_rbit	This is used to write the custom line styles. This value sets the <code>rbit</code> of the user linkage. <b>Range:</b> 0 or 1 <b>Default:</b> 0
igds_custom_ linestyle_mbit	This is used to write the custom line styles. This value sets the <code>mbit</code> of the user linkage. <b>Range:</b> 0 or 1 <b>Default:</b> 0
igds_custom_ linestyle_ibit	This is used to write the custom line styles. This value sets the <code>ibit</code> of the user linkage. <b>Range:</b> 0 or 1 <b>Default:</b> 0
igds_custom_ linestyle_class	This is used to write the custom line styles. This value sets the <code>class</code> of the user linkage. <b>Range:</b> 0 or 1 <b>Default:</b> 0
igds_element_ byteoffset	This is used to tell the position of the element. <b>Range:</b> Any Number <b>Default:</b> No default

Attribute Name	Contents
igds_model_name	The name of the model to which the feature belongs. <b>Note:</b> Applies to Version 8 DGN files. <b>Range:</b> String <b>Default:</b> No default
igds_model_id	The ID of the model to which the feature belongs. <b>Note:</b> Applies to Version 8 DGN files. <b>Range:</b> Any positive integer <b>Default:</b> No default
igds_element_new	The NEW property of the element. <b>Range:</b> YES or NO <b>Default:</b> No default
igds_element_modified	The MODIFIED property of the element. <b>Range:</b> YES or NO <b>Default:</b> No default
igds_date_last_modified	Stores the date the element of last modified in the format YYYYMMDD hh:mm:ssAM/PM. <b>Note:</b> Applies to Version 8 DGN files (Reader only) <b>Default:</b> No default
igds_element_locked	The LOCKED property of the element. <b>Range:</b> YES or NO <b>Default:</b> No default
igds_element_id	The unique ID of each element in a DGN file. <b>Note:</b> Applies to Version 8 DGN files. <b>Range:</b> Any positive integer <b>Default:</b> No default
mslink_x	Value of mslink key of the corresponding linkage, where x is the linkage number starting with 0. <b>Default:</b> No default
entity_num_x	Value of <code>entity_number</code> of the corresponding linkage where x is the linkage number starting with 0. <b>Default:</b> No default
link_type_x	Value of link type of the corresponding linkage, where x is the linkage number starting with 0. <b>Default:</b> No default
igds_element_association_id	The <code>tags</code> store this ID as the element ID it is attached to.
igds_z_value	This attribute is for the writer only and should be used only when 3D is intended to be forced. <b>Default:</b> 0
igds_chain_number	If <code>SPLIT_COMPLEX_CHAINS</code> is YES, then FME adds the attribute <code>igds_chain_number</code> which is added to each element of a chain split. <b>Default:</b> No default
igds_deleted	This attribute is set to yes only when the element read was a deleted element. <b>Default:</b> No default

Attribute Name	Contents
<code>igds_element_visibility</code>	This attribute has the value <code>yes</code> if the level the element is on has its display property set to "on"; otherwise the value is <code>no</code> . ( <b>Note:</b> Applies to Version 8 DGN files only.) <b>Default:</b> No default
<code>igds_element_view_independent</code>	This attribute has the value <code>yes</code> if the element is view independent; otherwise, the value is <code>no</code> . ( <b>Note:</b> Applies to Version 8 DGN files only.) <b>Default:</b> <code>yes</code>
<code>igds_is_graphic_cell_relative</code>	If this attribute is set to <code>yes</code> then the graphic cell is written as relative graphic cell which means that the cell member with the lowest level number will be put on the current i.e. feature's level whereas all the subsequent ones are offset accordingly e.g. if a cell had members on level 4,6 and 7 respectively and we are writing this cell feature on level 2 then the member with level 4 gets written on level 2 whereas the members with level 6 and 7 are written on level 4 and 5 respectively. This applies to members of nested cell also. Note that all the offsetted levels should be provided in the seed file otherwise the cell would be skipped. This attribute is for graphic cells only whereas for point cells and shared cells it is ignored. <b>Default:</b> <code>no</code>

## Attribute Linkages

Each element in an IGDS file may have one or more attribute linkages attached to it. The IGDS reader and writer support both user data and database attribute linkages. (Note, however, that the DGN V8 reader and writer do not support the interpretation of user linkages. Database linkages and MSLINKS are supported.) Because an element may have more than one linkage, linkages are stored in an FME feature attribute list named `igds_linkage{#}`. As with other feature attribute lists, # starts at zero and increments for each successive linkage. Currently, only database and DMRS linkages are supported for Version 8 DGN files (reading and writing).

### Attribute Lists – all linkages

The following attribute list item names are used by all linkages. Note that the `class` and various `bit` fields are not used when the linkage type is `dmrs`.

Linkage Parameter	Contents
<code>type</code>	The type of linkage. <b>Range:</b> <code>user   dbase   odbc   oracle   informix   ris   dmrs   framme</code> <b>Default:</b> No default <b>Note:</b> User and FRAMME linkages are not currently supported for Version 8 DGN files.

Linkage Parameter	Contents
class	Linkage Class. <b>Range:</b> 0..15 <b>Default:</b> 0
ibit	Linkage <i>ibit</i> value. This bit represents whether the linkage is informational or non-informational. <b>Range:</b> 0 1 <b>Default:</b> 0
mbit	Linkage <i>mbit</i> value. Indicates linkage has been modified. <b>Range:</b> 0 1 <b>Default:</b> 0
rbit	Linkage <i>rbit</i> value. The bit is set for remote linkages. <b>Range:</b> 0 1 <b>Default:</b> 0
ubit	Linkage <i>ubit</i> value. <b>Range:</b> 0 1 If set to 1 then linkage is user data linkage; if set to 0, then the linkage type is always dmrs. <b>Default:</b> 1

### Attribute Lists – user linkage

If the linkage is of type `user` (Version 8 DGN files supports user linkages having `userId` of 22244 and 39030 only), then these attribute list item names are used to specify the values for the user linkage:

Linkage Parameter	Contents
userId	The user -ID of the linkage. This is application specific. <b>Range:</b> 0..65535 <b>Default:</b> No default
long{#}	The user data associated with a user linkage may be specified as a list of 32-bit long integers or as a list of 16-bit words. If 32-bit long integers are used to fill out the attribute linkage, they have this suffix and are numbered sequentially starting from 0. <b>Range:</b> 32-bit integer <b>Default:</b> 0
word{#}	If 16-bit words are used to fill out the attribute linkage, they have this suffix and are numbered sequentially starting from 0. <b>Range:</b> 0..65535 <b>Default:</b> 0

### User linkages having `userId` of 22234 and 39030 (Extended entity data linkage)

In V7 these linkages are supported like any other user linkages but in v8 we store them as a blob and that gets carried over to v8 as a blob. Note that in order to get Extended Entity Data linkages to carry over correctly, the original file containing these linkages should be picked as the seed file. The support for these linkages would work for v8 to

v8 only. An attempt to transfer them from v7 to v8 or vice versa will not work. In v8 they store linkage attributes as follows

### Attribute Lists – dbase, odbc, oracle, ris, dmrs, informix linkages

Linkage Parameter	Contents
userId	The user -ID of the linkage. This value would be either 22244 or 39030 <b>Default:</b> No Default
blob	This stores the linkage as binary data <b>Default:</b> No Default
blobsize	Stores the size of the blob. <b>Range:</b> 0..256 <b>Default:</b> No Default
flags	Flags for the user linkage <b>Range:</b> 0..256 <b>Default:</b> No Default
type	Type of the linkage <b>Default:</b> user

If the linkage is of type *dbase*, *odbc*, *oracle*, *ris*, *dmrs*, or *informix*, then these attribute list item names are used to specify the values for the database linkage.

Linkage Parameter	Contents
entity_number	The entity number of the linkage. <b>Range:</b> 0..65535 <b>Default:</b> 1
key	The key value of the database linkage. This value corresponds to the value in a field in the attribute row associated with the element in the database. <b>Range:</b> 32-bit integer for 8 word linkage formats (i.e., Oracle, ODBC) and 24-bit integer for 4 word linkage formats (i.e., DMRS) <b>Default:</b> No default
readonly	This applies to the <i>dmrs</i> linkages and indicates whether or not the linkage is <i>readonly</i> . MGE systems also use this to differentiate between feature (which are <i>readonly</i> ) and attribute (which are not) linkages. <b>Range:</b> yes no <b>Default:</b> yes
trailing_flags	The trailing flags of the database linkage. This can be used to set the "daskey". (Not supported for Version 8 DGN files.) <b>Range:</b> signed 32-bit integer <b>Default:</b> 0

Linkage Parameter	Contents
firstword	This is the actual value of the first word of the <code>dmrs</code> linkage. It is stored for <code>dmrs</code> linkages only. <b>Range:</b> Unsigned 16-bit integer <b>Default:</b> 0
ltype	This attribute is used internally by the V8 writer and is not intended for users.
key2	This attribute is used internally by the V8 writer and is not intended for users.
suspectlinkage	This attribute is stored only if the reader detected that a certain linkage had an odd number of words, rather than an even number.

### Attribute Lists – framme linkage

If the linkage is a `framme` type, then the following attribute list item names are used to specify the values for the Facilities Rulebase Application Model Management Environment (FRAMME) linkage. Familiarity with the FRAMME system is necessary to fully understand the meaning of these attributes. Note that FRAMME linkages are not yet supported by Version 8 DGN files.

Linkage Parameter	Contents
ufid	The unique feature ID of the linkage. This is part of the database key used by FRAMME. <b>Range:</b> unsigned 32-bit integer <b>Default:</b> 0
design_file	The base name of the design file holding the linkage. This makes up the second part of the database key used by FRAMME. <b>Range:</b> character string <b>Default:</b> No default – not used when writing
state_num	The state number of the FRAMME feature <b>Range:</b> unsigned 16-bit integer <b>Default:</b> 0
rule_base_id	The FRAMME rule base identifier which is fixed at 0x20. <b>Range:</b> 0x20 (32 decimal) <b>Default:</b> 0x20
component_num	The component number of the FRAMME feature. <b>Range:</b> unsigned 16-bit integer <b>Default:</b> 0
component_count	The component count, or occurrence, of the FRAMME feature. <b>Range:</b> unsigned 16-bit integer <b>Default:</b> 0
feature_num	The feature number of the FRAMME feature. <b>Range:</b> unsigned 16-bit integer <b>Default:</b> 0

Linkage Parameter	Contents
long{#}	A list of 16-bit words that associated with "long" FRAMME linkages. <b>Range:</b> unsigned 16-bit integer <b>Default:</b> 0

### Attribute Lists – incosada linkage

If the linkage is an *incosada* type, then the following attribute names are added by the reader to hold the values for the British Columbia Forestry File (INCOSADA) linkage. Note that INCOSADA linkages are not supported in Version 8 DGN files, nor are they supported by the Design file writer.

Linkage Parameter	Contents
incosada_fid	The unique feature ID of the linkage. <b>Range:</b> Character string of size 32 consisting of hex digits
incosada_sequence_num	Sequence number of the linkage. <b>Range:</b> integer
incosada_feature_code	The feature code of the INCOSADA feature <b>Range:</b> unsigned 32-bit integer

### Example

For example, the FME feature specified by the partial transfer specification below would have two linkages. The first linkage is a user linkage which specifies that the shape is to be filled with color 12, and the second linkage is a dBASE linkage which links the element to the record with the key value of 1001. Note that if the same feature were to have an *igds\_fill\_color* attribute, its value would override that specified in the linkage.

```
MACRO fillUserId 65
MACRO fillMagic 67586
IGDS 32 igds_type igds_shape\
  igds_color 8 igds_weight 1\
  igds_linkage{0}.type user\
  igds_linkage{0}.userId $(fillUserId)\
  igds_linkage{0}.long{0} $(fillMagic) \
  igds_linkage{0}.long{1} 12\
  igds_linkage{1}.type dbase\
  igds_linkage{1}.key 1001
```

## Custom Line Styles

The custom line styles are stored as linkages of the element. Each custom line style name has an ID, which is a negative integer, and is stored as `igds_linkage{n}.long{0}`. (**Note:** V8 does not currently support custom line styles.)

It is fairly simple to write custom line styles while translating from *dgn* to *dgn*. However, it is the user's responsibility to provide the correct seed file containing the definitions of the custom line styles and to copy the `.rsc` file into the directory containing

other Microstation resource files. Here are the steps to configure Microstation for Custom Line Styles:

1. Open the seed file or destination file.
2. Select **Workspace > Configuration > Symbology**.
3. Click **Select**.
4. Select the `.rsc` file that you want to use.
5. Click **Add** to add the `.rsc` file to the list.
6. Click **OK** and then **Done**.
7. Close the file and then reopen it. It is important close the file; otherwise, the changes in the configuration just made are not reflected
8. From the **Active Line Style** pull-down menu, select **Custom**. Then select the name of line style that you want to use, and double click to activate it.
9. Select **File > SaveSettings**.

It is also possible to write a new custom line style when writing to a `dgn` file in cases where it was not originally provided in the source dataset. The user has to provide the value of `igds_custom_linestyle` whereas it is optional to provide the values for `igds_custom_linestyle_rbit`, `igds_custom_linestyle_mbit`, `igds_custom_linestyle_ibit` and `igds_custom_linestyle_class`. If they are not provided, the writer uses the default values.

---

**Note:**

- When translating from DGN to DGN where a complex chain in the source data set has custom line styles, then you have to set the keyword `PROPAGATE_CHAIN_ELEMENT_LINKAGES` to `true` to translate the custom line styles properly to the destination format.
  - If a complex chain has different line styles, then in order to retain those line styles, set "Drop Complex Chain" to yes.
- 

## Arcs

**igds\_type:** `igds_arc`

This geometry type is stored in an IGDS type 16 element. Arc features are just like ellipse features, except that two additional angles control the portion of the ellipse boundary that is drawn. Arcs with 3D rotations will be stroked into lines and returned as `igds_line` elements.

---

**Tip:** The function `@Arc()` can be used to convert an arc to a linestring. This is useful for storing arcs in systems which do not support them directly.

---

Attribute Name	Contents
<code>igds_primary_axis</code>	The length of the semi-major axis in ground units. <b>Range:</b> Any real number > 0 <b>Default:</b> No default
<code>igds_secondary_axis</code>	The length of the semi-minor axis in ground units. <b>Range:</b> Any real number > 0 <b>Default:</b> No default

Attribute Name	Contents
<code>igds_start_angle</code>	Refer to the @Arc (function) in the <i>FME Functions and Factories manual</i> for a detailed definition of <code>start_angle</code> . <b>Range:</b> 0.0..360.0 <b>Default:</b> No default
<code>igds_sweep_angle</code>	Refer to the @Arc (function) in the <i>FME Functions and Factories manual</i> for a detailed definition of <code>sweep_angle</code> . <b>Range:</b> Any real number > 0 <b>Default:</b> No default
<code>igds_rotation</code>	The rotation of the major axis. The rotation is measured in degrees counterclockwise up from horizontal. <b>Range:</b> -360.0..360.0 <b>Default:</b> 0
<code>igds_arc_orientation</code>	The orientation of the arc. As the sweep angle is always returned as positive, this field can be used to determine the original orientation of the arc. This attribute is only used during reading. <b>Range:</b> clockwise   counterclockwise <b>Default:</b> none
<code>igds_quat_p</code> <code>igds_quat_q</code> <code>igds_quat_r</code> <code>igds_quat_s</code>	Values of quaternion for 3D arcs <b>Default:</b> none

## Cells

### **igds\_type:** `igds_cell`

Cells correspond to IGDS element type 2. The FME feature used to hold a cell element does **not** contain the complete set of elements which make up the cell's definition. Instead, FME features representing IGDS cells contain only the cell's name, as well as rotation and scaling parameters. The IGDS reader skips all elements that define the cell (extracting only the text strings from any text elements in the cell), and the IGDS writer extracts the cell description from the supplied cell library to be output. Cell features are point features and have only a single coordinate. Writing of named cells is not currently supported by the V8 writer. However, the V8 writer can successfully handle unnamed cells (groups).

The IGDS reader may be set to expand cells. If the mapping file contains a `yes` setting for `IGDS_EXPAND_CELLS`, then each member element of the cell is read and output. However, the cell insertion point itself is **not** output. In addition, the cell members are assigned a unique cell sequence number in the `igds_cell_sequence_number`. This number can be used to later regroup the cell components if that is required.

If the setting for `IGDS_EXPAND_CELLS` is `no`, then only the cell insertion point is output.

Both *graphic* and *point* cells are supported. Graphic cells use the level, color, and style information from the cell library, and must always have a feature type of 0. Point cells use the level, color, and style information provided in the mapping file.

Both V7 and V8 can write cells. V8 can also preserve the cell structure. For example, if the cell had any nested cells, complex chains or complex shapes, then the whole nesting is preserved.

The IGDS reader also supports orphan or unnamed cells and is controlled by the keyword `IGDS_EXPAND_UNNAMED_CELLS`. A named/unnamed cell can have further nested named/unnamed cells. The way the IGDS Reader treats them, depending on their respective keywords, is explained in these sections:

- *Case-I: Named cell (root) nested named cell:*
- *Case-II: Named cells (root) nested unnamed cells:*
- *Case-III: Unnamed cells (root) nested unnamed cells:*
- *Case-IV: Unnamed cells (root) nested named cells:*

#### **Case-I: Named cell (root) nested named cell:**

`IGDS_EXPAND_CELLS (YES)` : The cell insertion point is not stored. Members of root and nested cells are stored as independent features.

`IGDS_EXPAND_CELLS (NO)` : Only the root cell's insertion point is stored.

#### **Case-II: Named cells (root) nested unnamed cells:**

`IGDS_EXPAND_CELLS (YES) AND IGDS_EXPAND_UNNAMED_CELLS (YES)` : Neither of the two cells is preserved. All members of both cells are output as independent features. No donuts are formed in case the unnamed cell contained overlapping polygons.

`IGDS_EXPAND_CELLS (YES) AND IGDS_EXPAND_UNNAMED_CELLS (NO)` : Neither of the two cells is preserved. All members of both cells are output as independent features. No donuts are formed in case the unnamed cell contained overlapping polygons.

`IGDS_EXPAND_CELLS (NO) AND IGDS_EXPAND_UNNAMED_CELLS (YES)` : Only the root cell is output. Nested unnamed cells are ignored.

`IGDS_EXPAND_CELLS (NO) AND IGDS_EXPAND_UNNAMED_CELLS (NO)` : Only the root cell is output. Nested unnamed cells are ignored.

#### **Case-III: Unnamed cells (root) nested unnamed cells:**

`IGDS_EXPAND_UNNAMED_CELLS (YES)` : The insertion point of the root cell is not preserved. No donuts are formed if existed. All elements of the root and the nested cells are given.

`IGDS_EXPAND_UNNAMED_CELLS (NO)` : Make donuts of all members of the root and nested cells.

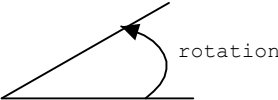
#### **Case-IV: Unnamed cells (root) nested named cells:**

`IGDS_EXPAND_UNNAMED_CELLS (YES) AND IGDS_EXPAND_CELLS (YES)` : Neither of the cells is preserved. No donuts are formed. All elements of both cells are output as independent features.

`IGDS_EXPAND_UNNAMED_CELLS (YES) AND IGDS_EXPAND_CELLS (NO)` : Only elements of the root unnamed cell are output. Nested cells are preserved and output as points.

`IGDS_EXPAND_UNNAMED_CELLS (NO) AND IGDS_EXPAND_CELLS (YES)` : All elements of nested named cells are output. Donuts are formed. If both cells have donuts then an aggregate of donuts is formed.

IGDS\_EXPAND\_UNNAMED\_CELLS (NO) AND IGDS\_EXPAND\_CELLS (NO) : The insertion point of the root unnamed cell is preserved. Donuts are formed from the root cell only. The nested cell is ignored and so are its members.

Attribute Name	Contents
igds_cell_name	The name of the cell. Corresponds to the name of the cell in a cell library. <b>Range:</b> Character String <b>Default:</b> No default
igds_cell_x_scale igds_cell_y_scale igds_cell_z_scale	The scaling factors to apply to the cell. <b>Range:</b> Any real number > 0 <b>Default:</b> 1
igds_cell_size	The size in ground units of the maximum span of the cell. If this is specified, the settings for igds_cell_x_scale, igds_cell_y_scale, and igds_cell_z_scale are ignored. If it is not specified, then the scaling factors described above are used. This attribute is not assigned any value by the reader. <b>Range:</b> Any real numbers > 0 <b>Default:</b> No default
igds_rotation	The rotation of the entire cell. The rotation is measured in degrees counterclockwise up from horizontal. <b>Range:</b> -360.0..360.0 <b>Default:</b> 0
	
igds_text_string{#}	When reading only, this contains the text string of the # <sup>th</sup> text element in the cell. <b>Range:</b> Any string
igds_cell_sequence_number	When reading only with IGDS_EXPAND_CELLS set to yes, this contains a unique number that can be used to regroup a cell with its component elements.
igds_cell_size_x	This is the difference of minX and maxX stored in ground units. <b>Note:</b> If igds_cell_size_x and igds_cell_size_y are both specified, then igds_cell_size_x_scale, igds_cell_size_y_scale and igds_cell_size_z_scale values are ignored.
igds_cell_size_y	This is the difference of minY and maxY stored in ground units.
igds_cell_num_members	Stores a cell's total number of members. <b>Range:</b> Any real numbers > 0 <b>Default:</b> No default
igds_unnamedcell_num_of_elements	Stores number of elements of an unnamed cell (group) <b>Range:</b> Any real numbers > 0 <b>Default:</b> No default

Attribute Name	Contents
igds_cell_insertion_x igds_cell_insertion_y igds_cell_insertion_z	Stores cell insertion point <b>Range:</b> Any real number <b>Default:</b> No default
igds_cell_element_class igds_cell_element_style igds_cell_element_color igds_cell_element_weight igds_cell_element_level	Stores properties of the cell if the cell is graphic. <b>Default:</b> No default
igds_cell12DTMat11 igds_cell12DTMat12 igds_cell12DTMat21 igds_cell12DTMat22	Cell's 2D matrix containing rotation and scale information. <b>Default:</b> No default
igds_cell13DTMat11 igds_cell13DTMat12 igds_cell13DTMat13 igds_cell13DTMat21 igds_cell13DTMat22 igds_cell13DTMat23 igds_cell13DTMat31 igds_cell13DTMat32 igds_cell13DTMat33	Cell's 3D matrix containing rotation and scale information. <b>Default:</b> No default

## Cells (Shared)

**igds\_type:** igds\_shared\_cell

Shared cells correspond to IGDS element type 34 and 35. They consist of two parts: the definition (Type 34) and the element (Type 35). The definitions list the component elements of the cells. The elements are made up of an insertion point as well as rotation and scaling parameters. The IGDS reader skips all elements that define the cell and only processes the element features as point features that have only a single coordinate.

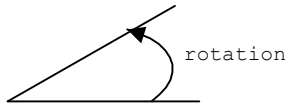
If `IGDS_EXPAND_CELLS` is set to yes, then the shared cells are expanded into its pieces; otherwise only the cell insertion point is output for each shared cell instance. Expansion of shared cells is supported by both V7 and V8.

The IGDS V8 writer can write shared cells.

Shared cell instances have the following attributes:

Attribute Name	Contents
igds_cell_name	The name of the cell. <b>Range:</b> Character String <b>Default:</b> No default
igds_cell_x_scale igds_cell_y_scale igds_cell_z_scale	The scaling factors to apply to the cell. <b>Range:</b> Any real number > 0 <b>Default:</b> 1

Attribute Name	Contents
igds_rotation	The rotation of the entire cell. The rotation is measured in degrees counterclockwise up from horizontal. <b>Range:</b> -360.0..360.0 <b>Default:</b> 0
igds_sharedcell_description	The description of the cell. (Supported for version 8 DGN files only.) <b>Range:</b> Character String <b>Default:</b> No default
igds_cell_num_members	Stores a cell's total number of members. <b>Range:</b> Any real numbers > 0 <b>Default:</b> No default



## Complex Shapes/Strings

**igds\_type:** igds\_complex\_shape

**igds\_type:** igds\_complex\_string

Complex shape/string elements are normally treated the same as shape/linestring elements by the IGDS reader. However, if the exact original composition of the complex shape is required, the `IGDS_AGGREGATE_COMPLEX_CHAINS` directive can be set to `yes` and then complex shape/string elements will be returned as single FME features with `igds_complex_shape/igds_complex_string` as their `igds_type`. This allows preservation of any arc elements that made up the boundary of the shape, for example.

The IGDS writer will accept and write out complex shape/string elements at any time.

The complex shape/string feature consists of an aggregate geometry. Each aggregate geometry corresponds to an entry in an attribute list. The list is called `igds_complex_elements{#}`, where # starts at 0 and increments for each aggregate element. The list's item names are identical to the component feature's attributes.

## Curves

**igds\_type:** igds\_curve

Curve features are used in Design files to represent smooth bezier curves. Curve features have four extra points which are used to determine the slope at the starting and ending points of the curve. These points are not part of the real coordinates of the feature, and are stored in the attribute list `igds_curve_slope{}`. The first two entries in the list define the slope points for the start of the feature, and the last two define the slope points for the end of the feature. The IGDS reader and writer interpret the curves coordinates as the points which define the curve. If the `PRESERVE_CURVES` directive is `YES`, then the reader does not interpolate points along the curve. If curves are not preserved, they will have interpolated points added to them and `igds_curve` elements will be returned as `igds_line` elements.

A curve feature has these attributes:

Attribute Name	Contents
igds_curve_slope{0}.x igds_curve_slope{0}.y igds_curve_slope{0}.z igds_curve_slope{1}.x igds_curve_slope{1}.y igds_curve_slope{1}.z	The ground coordinates of the slope points for the beginning of the feature. If the design file was two-dimensional (2D), then the .z attributes will not be present.
igds_curve_slope{2}.x igds_curve_slope{2}.y igds_curve_slope{2}.z igds_curve_slope{3}.x igds_curve_slope{3}.y igds_curve_slope{3}.z	The ground coordinates of the slope points for the end of the feature. If the design file was 2D, then the .z attributes will not be present.

**Tip:** When a curve feature is reprojected, its slope points are automatically reprojected.

## BSpline Curves

**igds\_type:** igds\_line

This is stored as an IGDS type 27 element. The information of the poles, knots and weights of a spline are stored in element types 21, 26 and 28 respectively. Currently, only *reading* of bsplines is supported. The bsplines are read and stroked into segments (which is why its igds\_type is stored as igds\_line).

## External Reference Files

In order to read the reference files, the keyword `READ_XREF_FILES` has to be set to *yes*. The default is *no*. All the reference files inherit the working units and offsets from the parent file and their respective units and offsets are ignored. The V8 reader can read both v7 and v8 attachments, whereas V7 will read only V7 references. Both V7 and V8 can read nested references. The nesting can be restricted to first level only by setting the keyword `READ_XREF_UPTO_FIRST_LVL` as *true*.

## Multilines

**igds\_type:** igds\_line

The multilines are stored with their igds\_type as igds\_line, but the fact that they are multilines can be detected from igds\_element\_type, which is stored as type 36. The multilines are stored as lines, therefore they are written as lines when performing a DGN to DGN translation. Currently, the multilines are imported with their centerlines only. However, the attributes such as offset and symbology (style, weight, color) of the pieces are stored in the list attribute igds\_multiline{}. When reading multilines from a V7 dataset, the multilines are ignored if they are part of a cell. In addition, if the keyword `READ_BYTE_OFFSET` is set to *true*, then it gets ignored for multilines.

A multiline has the attributes shown below.

Attribute Name	Contents
igds_mlineStyle{#}.offset igds_mlineStyle{#}.style igds_mlineStyle{#}.color igds_mlineStyle{#}.weight igds_mlineStyle{#}.level	Where <code>offset</code> is the perpendicular distance of the piece from the centerline, and <code>{}.style</code> , <code>{}.color</code> , <code>{}.weight</code> and <code>{}.level</code> are the line styles, color, weight and level of the individual pieces.
igds_mlinehdr_num_lines	Number of pieces of the multiline. <b>Range:</b> Any real number > 0 <b>Default:</b> No default
igds_mlinehdr_num_breaks	Number of breaks of the multiline. <b>Range:</b> Any real number > 0 <b>Default:</b> No default
igds_mlinehdr_num_nodes	Number of nodes of the multiline. <b>Range:</b> Any real number > 0 <b>Default:</b> No default
igds_mlinehdr_startcap_angle	Angle in degrees of the start cap. <b>Range:</b> -360.0..360.0 <b>Default:</b> 0
igds_mlinehdr_endcap_angle	Angle in degrees of the end cap. <b>Range:</b> -360.0..360.0 <b>Default:</b> 0
igds_mlinehdr_freeze_group	Multiline header attribute - value is always 0 (for internal use by the toolkit)
igds_mlinehdr_version	Multiline header version - currently it is 3 (for internal use by the toolkit)
igds_mlinehdr_closed	Whether or not multiline is closed. <b>Range:</b> 0 or 1 <b>Default:</b> No default
igds_mlinehdr_arc_cap_by_profile_line	Multiline header flags (for internal use by the toolkit)
igds_mlinehdr_offset_model_valid	Multiline header flags (for internal use by the toolkit)
igds_mlinehdr_offset_mode	Multiline header flags (for internal use by the toolkit)
igds_mlinehdr_placement_offset	Global offset from definition points. <b>Range:</b> Real Number <b>Default:</b> No default
igds_mlinehdr_style_id.lo igds_mlinehdr_style_id.hi	ID of multiline style element. <b>Range:</b> Integer <b>Default:</b> No default
igds_mlinehdr_styleScale	Scale of multiline style element. <b>Range:</b> Real Number <b>Default:</b> No default

Attribute Name	Contents
igds_mlinehdr_upd.v.x igds_mlinehdr_upd.v.y igds_mlinehdr_upd.v.z	Up direction vector for 3D to determine side orientation. <b>Range:</b> Real Number <b>Default:</b> No default
igds_mlattrib_startcap.usestyle igds_mlattrib_startcap.useweight igds_mlattrib_startcap.usecolor igds_mlattrib_startcap.cap_on_arc igds_mlattrib_startcap.cap_out_arc igds_mlattrib_startcap.cap_line igds_mlattrib_startcap.use_class igds_mlattrib_startcap.customstyle igds_mlattrib_startcap.cap_color_ from_segment igds_mlattrib_startcap.construction_ class	Flags specifying the properties of the multi-line. <b>Range:</b> 0 or 1 <b>Default:</b> No default
<p>The same list of attributes is repeated for end_cap and joint; for example:            igds_mlattrib_endcap.usestyle etc.            and            igds_mlattrib_joint.usestyle etc.</p>	
igds_mlattrib_startcap.style	Style of start cap. <b>Range:</b> 0..7 <b>Default:</b> 0
igds_mlattrib_startcap.weight	Weight of start cap. <b>Range:</b> 0..31 <b>Default:</b> 0
igds_mlattrib_startcap.color	Color of start cap. <b>Range:</b> 0..254 <b>Default:</b> 0
igds_mlattrib_endcap.style	Style of end cap. <b>Range:</b> 0..7 <b>Default:</b> 0
igds_mlattrib_endcap.weight	Weight of end cap. <b>Range:</b> 0..31 <b>Default:</b> 0
igds_mlattrib_endcap.color	Color of end cap. <b>Range:</b> 0..254 <b>Default:</b> 0
igds_mlattrib_joint.style	Style of joint cap. <b>Range:</b> 0..7 <b>Default:</b> 0
igds_mlattrib_joint.weight	Weight of joint cap. <b>Range:</b> 0..31 <b>Default:</b> 0

Attribute Name	Contents
igds_mlattrib_joint.color	Color of joint cap. <b>Range:</b> 0..254 <b>Default:</b> 0
igds_mlineStyle{#}.offset	Offset of each member of multiline <b>Default:</b> No Default
igds_mlineStyle{#}.lineattrib.usestyle igds_mlineStyle{#}.lineattrib.useweight igds_mlineStyle{#}.lineattrib.usecolor igds_mlineStyle{#}.lineattrib.cap_on_arc igds_mlineStyle{#}.lineattrib.cap_out_arc igds_mlineStyle{#}.lineattrib.cap_line igds_mlineStyle{#}.lineattrib.use_class igds_mlineStyle{#}.lineattrib.customstyle igds_mlineStyle{#}.lineat- trib.cap_color_from_segment igds_mlineStyle{#}.lineat- trib.construction_class	Values defining the flag of line attribute <b>Default:</b> No Default
igds_mlineStyle{#}.lineattrib.style	Style of line attribute <b>Range:</b> 0..7 <b>Default:</b> 0
igds_mlineStyle{#}.lineattrib.weight	Weight of line attribute <b>Range:</b> 0..31 <b>Default:</b> 0
igds_mlineStyle{#}.lineattrib.color	Color of line attribute <b>Range:</b> 0..254 <b>Default:</b> 0
igds_mlineStyle{#}.lineattrib.level	Level of line attribute <b>Default:</b> 0
igds_mlinenode_props	If a multiline has a large number of nodes, which is very likely, it will need to be cleaned up before viewing it in the Universal Viewer. Store these two attributes as strings with comma-separated values. This will be the protocol b_index1, bCount1, b_index2, bCount2, ..... b_indexn, bCountn The writer will parse them in the same order and use them. <b>Default:</b> No default
igds_mlinebreak{#}.segmask	Mask bit set for each line that is broken. <b>Default:</b> No default
igds_mlinebreak{#}.from_joint igds_mlinebreak{#}.to_joint	Flags setting the line break properties <b>Default:</b> No default
igds_mlinebreak{#}.point_offset	Offset from point <b>Default:</b> No default
igds_mlinebreak{#}.length	Break length <b>Default:</b> No default

Attribute Name	Contents
<code>igds_mlinebreak{#}.angle</code>	Reserved - should be 0.0 <b>Default:</b> No default

## Dimensions

### **igds\_type:** `igds_line`

The dimensions are stored with their `igds_type` as `igds_line` but the fact that they are dimensions can be detected from `igds_element_type` which is stored as type 33. The keyword `EXPLODE_DIMENSION_ELEM` controls the way the dimensions are imported. When it is set to `yes`, the dimensions are exploded into its pieces; when it is set to `no`, it is imported as an aggregate. The default is `yes`. When dimensions are imported as aggregates, the arcs are stroked and text features are output as list attributes only. Therefore, when performing a DGN-to-DGN translation with the option `EXPLODE_DIMENSION_ELEM` set to `no`, the text features will be lost. When reading dimensions from a V7 dataset, the dimensions are ignored if they are part of a cell. If the keyword `READ_BYTE_OFFSET` is set to `true`, then the dimensions will also be ignored.

A dimension element has the following attributes

Attribute Name	Contents
<code>igds_dim_text{#}.font</code> <code>igds_dim_text{#}.original_justification</code> <code>igds_dim_text{#}.text_size</code> <code>igds_dim_text{#}.text_width_multiplier</code> <code>igds_dim_text{#}.text_used_string_len</code> <code>igds_dim_text{#}.text_rotation</code> <code>igds_dim_text{#}.text_insertion_x</code> <code>igds_dim_text{#}.text_insertion_y</code> <code>igds_dim_text{#}.text_string</code>	Stores the standard attributes of the text member of the dimension. For example, <code>font</code> , <code>original_justification</code> , <code>size</code> and <code>width</code> etc.
<code>igds_dim_refx{#}.pt.x</code> <code>igds_dim_refx{#}.pt.y</code> <code>igds_dim_refx{#}.pt.z</code> <code>igds_dim_refx{#}.base_offset</code> <code>igds_dim_refx{#}.segment_text_offset</code> <code>igds_dim_refx{#}.flags</code> <code>igds_dim_refx{#}.rxflags.mode</code> <code>igds_dim_refx{#}.rxflags.hide_extension</code> <code>igds_dim_refx{#}.rxflags.use_text_margin</code> <code>igds_dim_refx{#}.rxflags.primary_text_exists</code> <code>igds_dim_refx{#}.rxflags.display_primary_plus_tolerance</code> <code>igds_dim_refx{#}.rxflags.display_primary_minus_tolerance</code> <code>igds_dim_refx{#}.rxflags.secondary_text_exists</code> <code>igds_dim_refx{#}.rxflags.display_secondary_plus_tolerance</code> <code>igds_dim_refx{#}.rxflags.display_secondary_minus_tolerance</code>	Stores the attributes of reference points as list attributes of the dimension.
<code>igds_dim_type</code>	Type of dimension to draw. <b>Range:</b> 1..53 <b>Default:</b> No default
<code>igds_dim_scale</code>	Scale of dimension <b>Range:</b> Any real number > 0 <b>Default:</b> No default

Attribute Name	Contents
igds_dim_style	Style of dimension <b>Range:</b> Any real number > 0 <b>Default:</b> No default
igds_dim_weight	Weight of dimension <b>Range:</b> 0..31 <b>Default:</b> 0
igds_dim_color	Color of dimension <b>Range:</b> 0..254 <b>Default:</b> 0
igds_dim_primary_accuracy	Primary accuracy of dimension <b>Range:</b> 0..254 <b>Default:</b> None
igds_dim_secondary_accuracy	Secondary accuracy of dimension <b>Range:</b> 0..254 <b>Default:</b> None
igds_witness_line_offset	Offset of witness lines <b>Range:</b> Real Number >0 <b>Default:</b> None
igds_witness_line_extension	Witness line extension <b>Range:</b> Real Number >0 <b>Default:</b> None
igds_base_to_text_dist	Base to text distance <b>Range:</b> Real Number >0 <b>Default:</b> None
igds_leader_to_text_dist	Leader to text distance <b>Range:</b> Real Number >0 <b>Default:</b> None
igds_text_min_leader	Text minimum leader <b>Range:</b> Real Number >0 <b>Default:</b> None
igds_arrow_width	Dimension arrow width <b>Range:</b> Real Number >0 <b>Default:</b> None
igds_arrow_height	Dimension arrow height <b>Range:</b> Real Number >0 <b>Default:</b> None
igds_center_mark_size	Center mark size <b>Range:</b> Real Number >0
igds_base_line_cos_value	Cos value of bearing angle of base line. <b>Range:</b> -1..1

<b>Attribute Name</b>	<b>Contents</b>
<code>igds_witness_line_cos_value</code>	Cos value for bearing angle of witness line. <b>Range:</b> -1..1
<code>igds_base_line_sin_value</code>	Sin value of bearing angle of base line. <b>Range:</b> -1..1
<code>igds_witness_line_sin_value</code>	Sin value of bearing angle of witness line. <b>Range:</b> -1..1
<code>igds_quat.w</code> <code>igds_quat.x</code> <code>igds_quat.y</code> <code>igds_quat.z</code>	Components of quaternion for 3D orientation <b>Range:</b> Any real number
<code>igds_view_id</code>	To allow to align by view <b>Range:</b> 1 character (1 byte)
<code>igds_num_ref_points</code>	Number of reference points <b>Range:</b> Any real number
<code>igds_options_count</code>	Count of options <b>Range:</b> Any real number
<code>igds_dim_text_width</code>	Width of text member <b>Range:</b> Real number > 0
<code>igds_dim_text_height</code>	Height of text member <b>Range:</b> Real number > 0
<code>igds_text_font</code>	Font of text member <b>Range:</b> 0..254
<code>igds_text_color</code>	Color of text member <b>Range:</b> 0..254
<code>igds_text_weight</code>	Weight of text member <b>Range:</b> 0..31
<code>igds_use_text_color</code>	Use text color and not the dimension element color <b>Range:</b> 1/0 (1 bit)
<code>igds_use_text_weight</code>	Use text weight and not the dimension element weight <b>Range:</b> 1/0 (1 bit)
<code>igds_measure_angle</code>	Measure angle and not the distance <b>Range:</b> 1/0 (1 bit)

## Ellipses

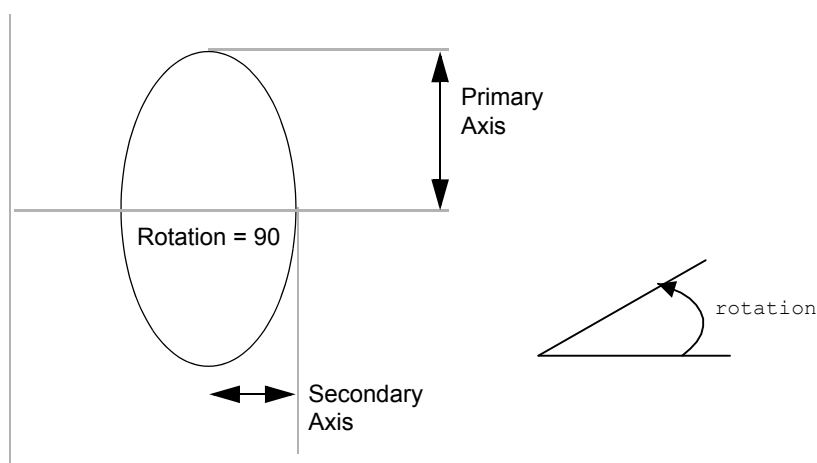
**igds\_type:** `igds_ellipse`

This geometry type is stored in an IGDS type 15 element. Ellipse features are point features, and only have a single coordinate. This point serves as the centre of the el-

lipse. Additional attributes specify the rotation, major axis, and minor axis of the ellipse.

**Tip:** The function @Arc() can be used to convert an ellipse to a polygon. This is useful for storing ellipses in systems which do not support them directly.

Attribute Name	Contents
igds_primary_axis	The length of the semi-major axis in ground units. <b>Range:</b> Any real number > 0 <b>Default:</b> No default
igds_secondary_axis	The length of the semi-minor axis in ground units. <b>Range:</b> Any real number > 0 <b>Default:</b> No default
igds_rotation	The rotation of the major axis. The rotation is measured in degrees counterclockwise up from horizontal. <b>Range:</b> -360.0..360.0 <b>Default:</b> 0
igds_fill_color	The color used to fill the shape. This will override any fill linkage that may be present on the feature. <b>Range:</b> 0..254 <b>Default:</b> No fill
igds_fill_color.red	The fill's red color intensity, as determined by looking up the fill color index in the color table. Reader only. <b>Range:</b> 0..254
igds_fill_color.green	The fill's green color intensity, as determined by looking up the fill color index in the color table. Reader only. <b>Range:</b> 0..254
igds_fill_color.blue	The fill's blue color intensity, as determined by looking up the fill color index in the color table. Reader only. <b>Range:</b> 0..254



---

**Tip:** The primary ellipse axis is **not** necessarily the longest axis, but rather the one whose orientation is specified by the rotation value.

---

## Lines

**igds\_type:** igds\_line

Features with their `igds_type` set to `igds_line` are stored in and read from IGDS files in one of three ways, depending on the number of coordinates they have.

Number of Coordinates	IGDS Element Type	Description
2	3	If the feature contained exactly two points, then an IGDS type 3 element is used to store the data if <code>IGDS_CREATE_LINE_ELEMENTS</code> was <code>yes</code> ; otherwise, a type 4 element will be created.
In V7: Between 3 and 101 In V8: Between 3 and 5000	4	If the coordinates can fit in a single element, then an IGDS type 4 element is used to store the line.
In V7: Greater than 101 In V8: Greater than 5000	12, 4	If the coordinates cannot fit into a single element, then they are grouped together into a complex line string element (type 12). This consists of a single type 12 element, followed by as many type 4 elements as required to hold all the coordinate data. The type 4 elements have their <i>complex</i> bit turned on.

There are no attributes specific to this type of element.

## Points

**igds\_type:** igds\_point

Strictly speaking, the IGDS file format does not support point data. However, for easier interoperability with other formats, the IGDS reader and writer define a synthetic IGDS type for point data. Such features have only a single coordinate, and are stored in an IGDS type 3 element<sup>1</sup> as a zero length line with the start and the end point the same. When the IGDS reader encounters such an element, it assigns an `igds_type` of `igds_point`. If the IGDS reader encounters a type 3 element with a different start and end point, it will assign an `igds_type` of `igds_line`.

There are no attributes specific to this type of element.

## Shapes

**igds\_type:** igds\_shape

---

1. FME treats IGDS type 3 elements with different start and end points as `igds_line` features.

Shape features are used in IGDS to represent closed polygons. The first coordinate in a shape feature must be equal to the last coordinate. Such features are stored in and read from IGDS files in one of two ways, depending on the number of coordinates in their boundaries:

Number of Coordinates	IGDS Element Type	Description
<b>In V7:</b> Between 3 and 101 <b>In V8:</b> Between 3 and 5000	6	If the coordinates can fit in a single element, then an IGDS type 6 element is used to store the shape.
<b>In V7:</b> Greater than 101 <b>In V8:</b> Greater than 5000	14, 4	If the coordinates cannot fit into a single element, then they are grouped together into a complex shape element (type 14). This consists of a single type 14 element, followed by as many type 4 elements as required to hold all the coordinate data. The type 4 elements have their <i>complex</i> bit turned on.

Shape elements have the following attributes.

Attribute Name	Contents
<code>igds_fill_color</code>	The color used to fill the shape. This will override any fill linkage that may be present on the feature. <b>Range:</b> 0..254 <b>Default:</b> no fill
<code>igds_fill_color.red</code>	The fill's red color intensity, as determined by looking up the fill color index in the color table. Reader only. <b>Range:</b> 0..254
<code>igds_fill_color.green</code>	The fill's green color intensity, as determined by looking up the fill color index in the color table. Reader only. <b>Range:</b> 0..254
<code>igds_fill_color.blue</code>	The fill's blue color intensity, as determined by looking up the fill color index in the color table. Reader only. <b>Range:</b> 0..254

**Tip:** Shapes will not be filled in MicroStation unless the 'View Attributes: Fill' checkbox is ticked, and a fill color is specified.

## Solids

**igds\_type:** `igds_solid`

Solids correspond to the grouped shapes in MicroStation. Solids consist of polygons or donut polygons. When a donut polygon is written out as a solid, all holes are output with the hole bit turned on, and are grouped together with the enclosing polygon. Groups are created in the Design file by creating an unnamed cell header element, and making each shape in the donut polygon a member of the group.

If `EXPAND_UNNAMED_CELLS` is set to `yes`, then unnamed cell components are output but the cell header itself is not output. In this case, donut polygons will not be formed from member shape elements. All member elements will retain their original colors. If it is `NO` (which is the default), then the cell is not exploded into its components and only the cell header is output. Donut polygons may be formed if multiple intersecting polygons existed.

If a solid consists of polygons without holes, then it is written out as `igds_shape`.

Solids are always filled, and accept an additional parameter to define the fill color. Holes within a solid will not be filled with the color.

The IGDS file format imposes a limit on the number of coordinates which can be present in a solid. This limit is around 16,000 for two-dimensional IGDS files, and around 10,000 for three-dimensional IGDS files. If a solid with more than an allowable number of coordinates is encountered, it is rejected and a message to that effect is logged to the FME log file.

---

**Tip:** Solids will not be filled in MicroStation unless the 'View Attributes: Fill' checkbox is ticked.

---

Solid elements have the following attributes.

Attribute Name	Contents
<code>igds_fill_color</code>	The color used to fill the solid. This will override any fill linkage that may be present on the feature. <b>Range:</b> 0..254 <b>Default:</b> 0
<code>igds_fill_color.red</code>	The fill's red color intensity, as determined by looking up the fill color index in the color table. Reader only. <b>Range:</b> 0..254
<code>igds_fill_color.green</code>	The fill's green color intensity, as determined by looking up the fill color index in the color table. Reader only. <b>Range:</b> 0..254
<code>igds_fill_color.blue</code>	The fill's blue color intensity, as determined by looking up the fill color index in the color table. Reader only. <b>Range:</b> 0..254

## 3D Solids

**igds\_type:** `igds_3d_solid`

This is supported for v8 only. This element should not be confused with `igds_solid` as it correspond to dgn element type 19 and not to unnamed cell. FME supports both reading and writing, however, its support is limited to only extrusions for now. Work to add support for other solid types is currently underway.

## Tags

Elements in a design file may have user-defined attributes attached to them. Such attributes are called *tags*, and these may be read and written (DGNV8 only) by FME. In addition, to supply a value for a user-defined attribute, tags may also be displayed as

text in the original design file. The `TAGS_AS_TEXT` directive controls whether or not tag data elements will be returned as text elements.

When reading a design file, FME first scans for all the tag data elements and tag set definition elements. Then as it reads each graphical element from the design file, it uses the element association ID to reconnect the data and attribute names with the graphical element. All the tag data values are then added to the feature returned into FME.

The attributes shown in the following table are added to an element for each associated tag.

---

**Note:** `<tag name>` is replaced by each `TAG NAME` that may be associated with the element. For example, if the element is associated with tags called "NUMLANES" and "PAVETYPE", then the feature would have attributes like "NUMLANES", `NUMLANES.height`, `PAVETYPE`, `PAVETYPE.rotation`, etc.

---

Note that most of the tag attributes are same as those of text. For example, `igds_tag_names{}.height` is the same as `igds_text_height` and is therefore not explicitly documented. All the other tag attributes are documented as follows:

Attribute Name	Contents
<code>igds_tag_names{}</code>	List of tag names attached to an element. <b>Default:</b> No default Required when writing tags through list attributes.
<code>&lt;tag name&gt;.tagset_name</code>	The name of the tagset the tag belongs to. <b>Default:</b> No default Required when writing tags to DGNV8
<code>&lt;tag name&gt;.tagtype</code>	The unique tag id <b>Default:</b> 1 <b>RANGE:</b> 1 = tag of type character string 3 = tag of type integer 4 = tag of type double Optional when writing tags to DGNV8. In case tagtype is not provided, it always defaults to character string i.e type 1.
<code>&lt;tag name&gt;.prompt</code>	The value of tag prompt as defined in the tagset <b>Default:</b> No default Optional when writing tags to DGNV8
<code>&lt;tag name&gt;.default_value</code>	The value of tag default as defined in the tagset <b>Default:</b> No default Optional when writing tags to DGNV8
<code>&lt;tag name&gt;.display</code>	The display value of tag as defined in the tagset. Note that the writer will always set it to NO if tag offsets are not found on the feature <b>Default:</b> no <b>RANGE:</b> yes/no Optional when writing tags to DGNV8

<code>&lt;tag name&gt;.x_offset</code> <code>&lt;tag name&gt;.y_offset</code> <code>&lt;tag name&gt;.z_offset</code>	Tag offset from the element. In case these values are not provided the writer uses some default values to offset tags from the element <b>Default:</b> No default Optional when writing tags to DGNV8
<code>&lt;tag name&gt;.urx</code> <code>&lt;tag name&gt;.ury</code> <code>&lt;tag name&gt;.urz</code>	Tag upper right range of rectangle. <b>Default:</b> No default Not required when writing tags to DGNV8
<code>&lt;tag name&gt;.llx</code> <code>&lt;tag name&gt;.lly</code> <code>&lt;tag name&gt;.llz</code>	Tag lower left range of rectangle. <b>Default:</b> No default Not required when writing tags to DGNV8

Note that tag writing is supported by DGNV8 only. In order to attach tags to an element, set writer keyword WRITE\_TAGS to yes. There are two ways tagset and tags definitions can be carried over to V8 writer:

1. *Through DEF lines:* This is the default behavior. The writer looks at the DEF lines to extract the information of tagsets and tag names. The feature type is assigned as the tagset whereas the user attributes become its tags. For instance, if the DEF line looks like this:

```
DGNV8_DEF Roads
    Name char(50)
    Type integer
```

then a tagset gets written with the name "Road" consisting two tags namely "Name" of data type string and "Type" of data type integer. The possible data types are char(n), integer and double. Note that this approach is introduced to automate the tag writing process and to avoid the amount of work involved using the "list attribute" approach as explained later. This approach has the following limitations:

- a. can write one tagset per feature only
- b. can define tag names and their corresponding data types only. The tag default, prompt and display properties cannot be set. The display property is automatically set to no and in order to turn it on the user need to set `<tag_name>.display` attribute on the feature with the value of yes.

If someone wants to write multiple tagsets then he can do so by using the "list attribute" approach. Also note that in case the writer sees list attributes under the name `igds_tag_names{ }` it ignores the tagset definitions provided on the DEF lines.

2. *Through list attributes:* Another way of writing tags is by providing all tag names as list attributes to `igds_tag_names{ }` on the feature.

The DGNV8 writer looks for the following attributes only when writing tags and uses them to calculate all other values. Therefore, any tag related attribute provided other than the following will be ignored.

```
igds_tag_names{ }
<tag name>.tagset_name
<tag name>.tagtype
<tag name>.prompt
```

```
<tag name>.display
```

```
<tag name>.default_value
```

```
<tag name>.x_offset
```

```
<tag name>.y_offset
```

```
<tag name>.z_offset
```

In case tag offsets are not provided then the writer uses some default values for the offsets and turns off tag's display property.

### Some tips of tag writing to avoid surprises:

When going from dgn->dgn, it is advised to ensure that the option "TAG\_AS\_TEXT" is turned off to avoid getting extra text element on top of the tags being written. Note that this option is set to "no" by default.

When going from dgn->dgn, if the source has tags attached to a cell then note that exploding cell will result into attaching tags to each cell member. Thus, each cell member will have same tags written in the output file.

## Multi-text Strings

**igds\_type:** igds\_multi\_text

Multi-text string features correspond to an IGDS text node (element type 7) grouping together a series of IGDS text elements (element type 17), each of which have their complex bit turned on. This feature uses the same attribute names as a text node, plus it has a feature attribute list of text string attributes. The list is called `igds_text_elements{#}`, where # starts at 0 and increments for each text element. The list's item names are identical to the text string features attributes.

---

**Tip:** Multi-text strings can be used to group together text so that it will be manipulated as a single entity with MicroStation.

---

Multi-text features are point features, and only have a single coordinate. This coordinate is used when the text node is created. If the feature had no coordinates of its own, the text node is created with the coordinates of the first text string. The coordinates for each of the text strings are stored in the FME feature using the following attribute names.

Attribute Name	Contents
<code>igds_text_elements{#}.x</code>	The x coordinate of the # <sup>th</sup> text element. <b>Range:</b> Any real number <b>Default:</b> No Default
<code>igds_text_elements{#}.y</code>	The y coordinate of the # <sup>th</sup> text element. <b>Range:</b> Any real number <b>Default:</b> No Default
<code>igds_text_elements{#}.z</code>	The z coordinate of the # <sup>th</sup> text element. <b>Range:</b> Any real number <b>Default:</b> 0
<code>igds_number_of_strings</code>	The number of text elements in the multi-text feature

Attribute Name	Contents
igds_split_multitext	Is added to the feature with the value "yes" if splitting multi-text. <b>Default:</b> No Default

If a setting for a particular text element is not present in the `igds_text_elements` list, then the setting specified for the previous text element will be used. If the first element does not have some settings specified, then the corresponding settings will be borrowed from the text node.

**Tip:** When a multi-text string feature is reprojected, its rotation and text size are also automatically adjusted to be correct in the new coordinate system.

For example, the FME feature specified by the below partial transfer specification would create a text node, followed by two text strings, as a single complex element.

```
IGDS 32 igds_type      igds_multi_text      \
  igds_node_number 15      \
  igds_font        31      \
  igds_rotation    0      \
  igds_text_size   40      \
  igds_color       2      \
  igds_justification1      \
  igds_text_elements{0}.igds_font33      \
  igds_text_elements{0}.igds_rotation3.1      \
  igds_text_elements{0}.igds_text_size52      \
  igds_text_elements{0}.igds_color 4      \
  igds_text_elements{0}.igds_text_stringHello      \
  igds_text_elements{0}.x477556      \
  igds_text_elements{0}.y5360183      \
  igds_text_elements{0}.z20      \
  igds_text_elements{1}.igds_text_stringWorld      \
  igds_text_elements{1}.x47755      \
  igds_text_elements{1}.y5359177      \
  igds_text_elements{1}.z20
```

Note that in this example, the justification code (1) used for the text node would be propagated to each of the text elements, but that the color used in the text node (2) would not be used in any of the text elements because the first one set the color to 4.

The in-memory snapshot of the FME feature created by the IGDS writer from this transfer specification is shown below.

<b>Feature Type: 32</b>	
<b>Attribute Name</b>	<b>Value</b>
igds_type	igds_multi_text
igds_node_number	15
igds_font	31
igds_weight	1

<b>Feature Type: 32</b>	
<b>Attribute Name</b>	<b>Value</b>
igds_text_size	40
igds_color	2
igds_rotation	0
igds_justification	1
igds_text_elements{0}.igds_text_string	Hello
igds_text_elements{0}.igds_font	33
igds_text_elements{0}.igds_rotation	3.1
igds_text_elements{0}.igds_justification	1
igds_text_elements{0}.igds_text_size	52
igds_text_elements{0}.x	477556
igds_text_elements{0}.y	536018
igds_text_elements{0}.z	20
igds_text_elements{1}.igds_text_string	World
igds_text_elements{1}.igds_font	33
igds_text_elements{1}.igds_rotation	3.1
igds_text_elements{1}.igds_justification	1
igds_text_elements{1}.igds_text_size	52
igds_text_elements{1}.x	477556
igds_text_elements{1}.y	5359177
igds_text_elements{1}.z	20
Coordinates: (477553,5360181,20)	

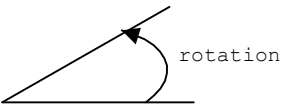
## Text Nodes

**igds\_type:** igds\_text\_node

Text nodes correspond to IGDS element type 7. Text node features are point features, and only have a single coordinate. Normally, text nodes are used to group together lines of text into a single complex element. However, such text groups are handled by the `igds_multi_text` type and not by this type, which is used only for text nodes with no attached text.

**Tip:** Free standing text nodes are often used as point features in IGDS files, with the text node number holding a key to related attribution.

Text node elements have the following attributes.

Attribute Name	Contents	
igds_node_number	The node number assigned to the text node. <b>Range:</b> 0..65535 <b>Default:</b> 0	
igds_font	The IGDS font number for the text node. For free standing text nodes, this value is relatively meaningless. <b>Range:</b> 0..254 <b>Default:</b> 25	
igds_rotation		The rotation of the text node. The rotation is measured in degrees counter clockwise up from horizontal. For free standing text nodes, this value is relatively meaningless. <b>Range:</b> -360.0..360.0 <b>Default:</b> 0
igds_justification	The justification code for the text node. <b>Range:</b> 0..14 0 is Left/Top 8 is Center/Bottom 1 is Left/Center 9 is Right Margin/Top 2 is Left/Bottom 10 is Right Margin/Center 3 is Left Margin/Top 11 is Right Margin/Bottom 4 is Left Margin/Center 12 is Right/Top 5 is Left Margin/Bottom 13 is Right/Center 6 is Center/Top 14 is Right/Bottom 7 is Center/Center <b>Default:</b> 5	
igds_text_size	The text size of the text in the node. This is stored as the text height in the element. The text size is measured in ground units. <b>Range:</b> Any real number > 0 <b>Default:</b> 20	
igds_text_width_multiplier	The text width of the text in the node. The text width is measured in ground units. If this is not supplied, then igds_text_size is used. <b>Range:</b> Any real number > 0 <b>Default:</b> Value of igds_text_size	
igds_line_spacing	The line spacing between lines of text associated with the text node. It is measured in ground units. <b>Range:</b> Any real number > 0 <b>Default:</b> 0	
igds_number_of_strings	The number of text elements associated with the text node. If the number is greater than 0, then the text node is returned as an igds_multi_text. This is only used by the IGDS reader. <b>Range:</b> integer >= 0 <b>Default:</b> Not applicable. This field is only used by the reader.	
igds_max_string_length	The maximum length of the strings associated with the text node. <b>Range:</b> integer >= 0 <b>Default:</b> 255	

Attribute Name	Contents
igds_max_used_string_length	The actual length of the strings associated with the text node. (Not included in Version 8 DGN files.) <b>Range:</b> integer $\geq 0$ <b>Default:</b> 0

**Tip:** When a text node feature is reprojected, its rotation and text size are also automatically adjusted to be correct in the new coordinate system.

## Text Strings

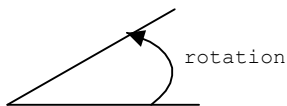
**igds\_type:** igds\_text

Text string features correspond to IGDS element type 17. Normally, text strings are grouped together into a single complex element within MicroStation by text nodes. However, such text groups are handled in the FME by the `igds_multi_text` type and not by this type, which is used only for single, free standing text strings. Text string features are point features, and only have a single coordinate. Note that V8 reader is capable of reading texts in unicode (UTF-16) in Windows only.

**Tip:** Some applications may use the graphic group field to logically group related text elements together.

Text strings have the following attributes.

Attribute Name	Contents
igds_original_justification	This attribute contains the original justification of the element when it was placed in the Design file. Once placed, all text elements are stored in the Design file using lower left corner (code 2) justification. Therefore, all text elements returned by the reader have an <code>igds_justification</code> of 2. The IGDS reader returns the original justification code in this attribute. The IGDS writer stores the value of this attribute in the justification bits of the placed text element, but it does NOT use its contents to determine placement of the text. <b>Range:</b> 0..12
igds_text_string	The text string to be output. Text strings longer than 255 characters cannot be stored in a Design file and will be broken into multiple separate text elements. <b>Range:</b> Any string <b>Default:</b> No Default
igds_font	The IGDS font number for the text string. <b>Range:</b> 0..254 <b>Default:</b> 25
igds_rotation	The rotation of the text string. The rotation is measured in degrees counterclockwise up from horizontal. <b>Range:</b> -360.0..360.0 <b>Default:</b> 0



Attribute Name	Contents
igds_justification	<p>The justification code for the text string. See the <i>Text Node</i> section for documentation on the mapping of numbers to alignments. Note that if this is specified, the IGDS writer will compute the lower left corner of the text as best it can and use that when the element is written to the Design file. Text elements in a Design file are always stored using a lower left corner.</p> <p><b>Range:</b> 0..2,6..8, and 12..14  <b>Default:</b> 2</p>
igds_text_size	<p>The text size of the text, measured in ground units. This is stored as the height of the text element.</p> <p><b>Range:</b> Any real number &gt; 0  <b>Default:</b> 20</p>
igds_text_width_multiplier	<p>The text width of the text. The text width is measured in ground units.</p> <p>If this is not supplied, then <code>igds_text_size</code> is used.</p> <p><b>Range:</b> Any real number &gt; 0  <b>Default:</b> value of <code>igds_text_size</code></p>
igds_text_num_lines	<p>If this is specified and is greater than 1, it will cause the text string to be broken into the number of lines specified, and output as that number of text elements stacked vertically.</p> <p>If this is not supplied, then <code>igds_text_size</code> is used.</p> <p><b>Range:</b> Any integer &gt; 0  <b>Default:</b> 1</p>
igds_text_horizontal_flip	<p>Indicates whether or not the text should be flipped horizontally when it is displayed. This is represented in a Design file by storing the text width as a negative number if the text should be flipped.</p> <p>(Not included in Version 8 DGN files.)</p> <p><b>Range:</b> Yes No  <b>Default:</b> No</p>
igds_text_vertical_flip	<p>Indicates whether or not the text should be flipped vertically when it is displayed. This is represented in a Design file by storing the text height as a negative number if the text should be flipped.</p> <p>(Not included in Version 8 DGN files.)</p> <p><b>Range:</b> Yes No  <b>Default:</b> No</p>
igds_insertion_x igds_insertion_y igds_insertion_z	<p>The x, y, and z location of the original insertion point for the text, before the justification was applied. Reader only.</p> <p><b>Range:</b> Any real number &gt; 0</p>
igds_lower_x igds_lower_y igds_upper_x igds_upper_y	<p>The lower left and upper right x and y coordinates of the bounding box of the text. Reader only.</p> <p><b>Range:</b> Any real number &gt; 0</p>
igds_textstyle_id	<p>The ID of the textstyle being used.</p> <p><b>Default:</b> No</p>

<b>Attribute Name</b>	<b>Contents</b>
igds_textstyle_char_spacing	Textstyle character spacing. <b>Default:</b> No
igds_textstyle_slant	Textstyle slant <b>Default:</b> No
igds_textstyle_underline_spacing	Textstyle underline spacing <b>Default:</b> No
igds_textstyle_underline_color	Textstyle underline color <b>Default:</b> No
igds_textstyle_underline_style	Textstyle underline style <b>Default:</b> No
igds_textstyle_underline_weight	Textstyle underline weight <b>Default:</b> No
igds_textstyle_overline_spacing	Textstyle overline spacing <b>Default:</b> No
igds_textstyle_overline_color	Textstyle overline color <b>Default:</b> No
igds_textstyle_overline_style	Textstyle overline style <b>Default:</b> No
igds_textstyle_overline_weight	Textstyle overline weight <b>Default:</b> No
igds_textstyle_line_offset.x igds_textstyle_line_offset.y	Coordinates of textstyle line offset <b>Default:</b> No
igds_textstyle_codepage	Textstyle codepage <b>Default:</b> No
igds_textstyle_bg_color	Textstyle background color <b>Default:</b> No
igds_textstyle_bg_style	Textstyle background style <b>Default:</b> No
igds_textstyle_bg_weight	Textstyle background weight <b>Default:</b> No
igds_textstyle_bg_border.x igds_textstyle_bg_border.y	Coordinates of textstyle background border <b>Default:</b> No
igds_textstyle_bg_fill_color	Textstyle background fill color <b>Default:</b> No
igds_textstyle_color	Textstyle color <b>Default:</b> No
igds_textstyle_font	Textstyle font <b>Default:</b> No
igds_textstyle_tnode_word_wrap_len	Textstyle word wrap length <b>Default:</b> No

Attribute Name	Contents
igds_textstyle_overrides_style1 igds_textstyle_overrides_style2	Textstyle override styles <b>Default:</b> No
igds_textstyle_txflags	Textstyle flags <b>Default:</b> No
igds_textstyle_exflags	Textstyle extended flags

**Note:**

- When a text string feature is reprojected, its rotation and text size are also automatically adjusted to be correct in the new coordinate system.
- When writing textstyles, make sure that all the textstyles are added to the seed file being used.

## Writing Levels in V8 (DEFLine Params)

In V8, the feature type is always taken as the level name. When WRITE\_TAGS is set to yes then feature type will also be the tagset name. For more information refer to section under TAGS. Levels are created with this name and the level numbers are assigned from the DEF line. However, for backward compatibility, a feature's `igds_level` and `igds_level_name` overwrite the `level_number` and `feature_type`.

The following protocol is used when processing levels:

1. If the level is already provided in the seed file, then it is left as-is.
2. If the feature type has a corresponding DEF line parameter, and if that level is not already in the seed file, then that level is created with symbology as defined on the DEF line. This allows users to create levels with the desired symbology. Note that in order to apply the level symbology to the features belonging to the level, the attributes `igds_color_set_bylevel`, `igds_style_set_bylevel` and `igds_weight_set_bylevel` must be set to Yes. If none of the above are provided, then symbology of the first feature appearing in a level is assigned as its symbology.
3. If the feature type has a corresponding DEF line parameter, and if the value of `igds_level` is undefined and the feature type (level Name) is not "Default" then the DGN writer assigns level number 1 to it.

The DEFLine Params for defining levels are as follows:

Parameter Name	Contents
<code>igds_level</code>	The level number corresponding to the feature type. Note that feature type is treated as level name.
<code>igds_level_comment</code>	The comment of the destination level.
<code>igds_level_color</code>	The color of the destination level.
<code>igds_level_style</code>	The style of the destination level.
<code>igds_level_weight</code>	The weight of the destination level.

## Example

```
DGNV8_1_DEF test2 \
  igds_level 4 \
  igds_level_comment "This is a test" \
  igds_level_color 3 \
  igds_level_style 4 \
  igds_level_weight 2 \
```

In this case, the level will be created with the level name "test2", the corresponding level number of 4 and the comment and symbology as defined above. Note that if any feature being written to this level is intended to have symbology `by_level` then the attributes `igds_color_set_bylevel`, `igds_style_set_bylevel` and `igds_weight_set_bylevel` must be set to Yes.

Note that destination feature types are treated differently for V7 and V8. Version 7 always sees the destination feature types as level numbers, whereas V8 sees them as level names. For V8, the feature attribute `igds_level_name` overwrites `feature_type`, and feature attribute `igds_level` overwrites DEF line parameter `igds_level`.

Note that a workspace originally created using a V8 seed file can only be used to write to V8. A workspace originally created using a V7 seed file can be used to write to both V7 (although there will be a difference in the way destination feature types are handled) or V8. Two additional limitations are applied to V8:

1. Limitation of level numbers from 1 to 63 will also be applied to V8.
2. Feature types will always be generated as level numbers just like V7, but those level numbers will be treated as `level_names` by V8. For instance, if you tried to write to `level_number 3`, the level would be written as level number 3 for V7, but the level name would be written as "3" for V8 (when its number may or may not be 3). This can be overcome by specifying values for `igds_level_name` and `igds_level`.

