

ESRI ArcInfo Coverage/ ESRI ArcInfo Export (E00) Reader/Writer

FORMAT NOTES:

- Coverage reading and writing is available only with FME ESRI Edition, FME Smallworld Edition and FME Oracle Edition.

The ESRI® ArcInfo® Coverage and Export (E00) Reader/Writer allows FME to read and write binary coverages, E00 files and Info tables. Full support for compressed E00 files – export option 1 or 2 – is also provided. Safe Software’s *FME ESRI Edition* also reads and writes binary ArcInfo coverages directly.

E00 Quick Facts

Format Type Identifier	E00 or ARCINFO
Reader/Writer	Both
Licensing Level	<ul style="list-style-type: none"> Reader: Base Reader + Writer: FME ESRI Edition, FME Smallworld Edition and FME Oracle Edition
Dependencies	None
Dataset Type	<ul style="list-style-type: none"> Directory for ArcInfo Coverage reader Directory for ArcInfo Table reader File for E00 reader Directory for both writers
Feature Type	<ul style="list-style-type: none"> Subdirectory base name for ArcInfo File base name for E00
Typical File Extensions	.e00 (.e01, .e02, ...)
Automated Translation Support	Yes
User-Defined Attributes	Yes
Coordinate System Support	Yes
Generic Color Support	No
Spatial Index	Never
Schema Required	Yes
Transaction Support	No
Geometry Type	e00_type

Geometry Support			
Geometry	Supported?	Geometry	Supported?
aggregate	no	point	yes
circles	no	polygon	yes
circular arc	no	raster	no

Geometry Support			
Geometry	Supported?	Geometry	Supported?
donut polygon	yes	solid	no
elliptical arc	no	surface	no
ellipses	no	text	yes
line	yes	z values	n/a
none	yes		

Overview

A single E00 file describes a complete ArcInfo coverage. The file itself is actually an archive of several smaller files, referred to here as *subfiles*. Some of these subfiles have fixed names which do not vary from coverage to coverage, and follow a pre-defined data format. These are referred to as the *standard subfiles*.

The remainder of the subfiles contained within an E00 are the *info files*. These files may contain user-defined attributes, and have names which vary from coverage to coverage. The ways in which the names vary are discussed in *Info Files* on page 585.

Reader Overview

The E00/ArcInfo reader produces FME features for all feature data contained in a single E00 file, binary ArcInfo coverage or ArcInfo table. In order to process multiple E00 files, coverages, you must invoke the FME for each E00 file, or use the Multi-Reader, described in the *Multi-Reader* chapter.

Large E00 files are often split into smaller files, named <filename>.e00, <filename>.e01, <filename>.e02, etc. The E00 reader automatically detects this and reads the set of files as if they were a single E00 file.

To read ArcInfo coverages, specify the directory that contains the coverage to the E00 reader.

To read just the ArcInfo tables, specify the "info" directory that contains the info tables to the ArcInfo reader. All info tables will be processed as data with no spatial information attached to it even though they may be part of the coverage.

Reader Directives

The suffixes listed are prefixed by the current <ReaderKeyword> in a mapping file. By default, the <ReaderKeyword> for the E00 writer is E00 or ARCINFO.

DATASET

Required/Optional: *Required*

The E00 reader processes the directive <ReaderKeyword>_DATASET, where <ReaderKeyword> is the keyword assigned to the E00 reader. By default, the reader keyword is E00 for E00 files, or ARCINFO when working with binary ArcInfo coverages.

The <ReaderKeyword>_DATASET directive specifies the data set to be read by the E00 reader. When reading E00 files, this is the name of the E00 file, including the .e00 suffix. If the E00 reader encounters the end of the E00 file <filename>.e00 while it is

expecting more data to process, it will attempt to use the file <filename>.e01 as a continuation of the input. No specific mention of <filename>.e01 is required.

When reading binary ArcInfo coverages directly, the value of <Readerkeyword>_DATASET is the path of the directory containing the files that make up the coverage. (Note that it is not the directory containing the individual coverage directories, but rather one of the coverage directories itself.) No additional configuration is required to tell the reader to process a binary coverage instead of an E00 file. If the supplied argument is a directory, the reader will assume the data set is a binary coverage.

When reading binary ArcInfo tables directly, the value of <Readerkeyword>_DATASET is the path to the "info" directory which contains info tables.

[Workbench Parameter: <WorkbenchParameter>](#)

TEXT_CURVE

Required/Optional: *Optional*

There is an additional directive that tells the E00 reader how to deal with text elements which follow a splined curve in ArcInfo. In the past, the FME has simply drawn a straight line from the first point of the curve to the last point, and placed the text along that line. The default behavior now is to space the characters along the original curve, and generate a separate character for each (non-whitespace) character of the text. The directive <ReaderKeyword>_TEXT_CURVE allows one to change the FME's interpretation of curved text features. The default value for the directive is FOLLOW; a value of IGNORE will revert the FME to its traditional behavior; and a value of FIT will tell FME to evenly space out the characters of the text along the curve, so that the left edge of the first character is on the first point of the curve, and the right edge of the last character is on the last point.

Value: FIT | FOLLOW | IGNORE

Default Value: FOLLOW

[Workbench Parameter: <WorkbenchParameter>](#)

SINGLE_BYTE_TEXT

Required/Optional: *Optional*

If the E00 reader is placing the text characters along their curve (i.e., if the <ReaderKeyword>_TEXT_CURVE was not given a value of FIT or FOLLOW), it normally inspects the text content to try to detect whether any are multi-byte representations of "international" characters. If it finds a pair of bytes that it thinks define a single character, it will use those two bytes in a single feature representing that character of text. This behavior might lead to incorrect representation of some character strings, if they happen to be composed of single-byte characters that look like multi-byte characters. This automatic detection of multi-byte characters may be disabled by providing the SINGLE_BYTE_TEXT directive a value of YES.

Value: YES | NO

Default Value: NO

[Workbench Parameter: <WorkbenchParameter>](#)

INCLUDE_BND

Required/Optional: *Optional*

ArcInfo coverages typically include an info file named BND, which defines the extents of the coverage. FME will normally ignore the contents of this file. If the <ReaderKeyword>_INCLUDE_BND keyword is specified with a value of YES, FME will create a single feature representing the coverage extent information.

The extent is defined on the resulting feature with the attributes XMIN, YMIN, XMAX, and YMAX. In FME, the feature will have a polygon geometry corresponding to these attributes' values; in FME Objects, the BND feature will have no geometry.

Value: YES | NO

Default Value: NO

Example:

```
E00_INCLUDE_BND Yes
```

[Workbench Parameter: <WorkbenchParameter>](#)

INCLUDE_TIC

Required/Optional: *Optional*

ArcInfo coverages typically include an info file named TIC, which defines the tic points for the coverage. FME will normally ignore the contents of this file. If the <ReaderKeyword>_INCLUDE_TIC is specified with a value of YES, FME will create a feature for each TIC point.

The features resulting from reading the TIC file will have the IDTIC, XTIC, and YTIC attributes as defined in the TIC file, and will have a point geometry corresponding to (XTIC,YTIC).

Value: YES | NO

Default Value: NO

Example:

```
E00_INCLUDE_TIC Yes
```

[Workbench Parameter: <WorkbenchParameter>](#)

HYPHENS_ARE_VALID

Required/Optional: *Optional*

When set to "Yes" the reader will not convert hyphens in attribute names to underscores. The one exception to this rule is that if the attribute name ends in "-ID" – in this case, it will be converted to _ID no matter what the directive is set to.

So, for example, if the attribute name is "HYPH-ENS-ID", then when the directive is set to "Yes", it will be read as "HYPH-ENS_ID" and if the directive is set to "No", it will be read as "HYPH_ENS_ID".

If this directive is missing, then it will have an implied value of "No". This is to ensure backwards compatibility, so workspaces created with a previous version of FME (one that does not yet support this directive) continue to work as they always have.

Value: YES | NO

Default Value: Yes

Example:

```
E00_HYPHENS_ARE_VALID Yes
```

Workbench Parameter: [<WorkbenchParameter>](#)

GENERATE_NODE_FEATURES

Required/Optional: *Optional*

Traditionally the ArcInfo reader reads the NAT table and outputs the NODE attributes as a plain set of attributes. If the value is YES, the endpoints of the ARC features are turned into NODE features, which are then joined with the NAT table attributes to provide fully formed point features.

Value: YES | NO

Default Value: NO

Example:

```
E00_GENERATE_NODE_FEATURES NO
```

Workbench Parameter: [N/A \(settings box only\)](#)

There are no other directives processed by the E00 reader, meaning there are no DEF lines for reading E00 features. The features obtained from the specified data set take the form described in the remainder of this chapter.

Writer Overview

The ArcInfo writer provides the ability to generate, in a single invocation of the FME, several E00 files or binary coverage directories. All E00 files or coverages created in a single run of the FME will be placed into a single directory.

Unlike the reader, the writer requires DEF lines to define the attributes of the output E00 files or binary coverage directories. The writer provides a mechanism to apply a single set of attributes to all geometric features placed into a file, as well as allowing each specific info file to have a unique set of attributes.

The E00 writer can create compressed or uncompressed E00 files, or, with *FME ESRI Edition*, binary ArcInfo coverages.

Writer Directives

The suffixes shown are prefixed by the current `<WriterKeyword>` in a mapping file. By default, the `<WriterKeyword>` for the E00 writer is `E00` or `ARCINFO`.

DATASET

Required/Optional: *Required*

Unlike the E00 reader, which reads a single E00 file, the writer may create several files. The `DATASET` keyword for the E00 writer is the directory into which the created E00 files (or coverages) will be written.

Workbench Parameter: [<WorkbenchParameter>](#)

DEF

Required/Optional: *Required*

Workbench Parameter: [<WorkbenchParameter>](#)

An E00 file or coverage must be completely defined before it may be written. The definition specifies the base name of the file and the names and types of all attributes on all info files within the E00 file. The syntax of an E00 `DEF` line is:

```
<ReaderKeyword>_DEF <baseName> \
  [E00_FORCE_OUTPUT <subfileName>[, <subfileName>]+] \
  [<attrName> <attrType>[, INDEX]]+
```

The file name of the physical E00 file is constructed by appending the `.e00` suffix to its `baseName`. Binary coverages are written as a directory named `baseName`. Coverage names are always truncated to contain thirteen or fewer characters, and converted to lowercase letters, when forming output directory names.

The `E00_FORCE_OUTPUT` keyword is used to force the named subfiles to appear in the output E00 file or binary coverage, even if no contents are specified for them. The value for `subfileName` can either be the name of a one of the standard subfiles `LAB` or `TOL`, or can be any info file. (The naming of info files follows the same pattern as in the `<infoFile>:<attrName>` notation discussed below.) The `TOL` and `LAB` files are normally generated by the E00 writer, but this behavior can be overridden by specifying an `E00_FORCE_OUTPUT` keyword that does not list the `TOL` or `LAB` subfile: if a list of files are forced to be output, and no `TOL` file is specified in the list, then a `TOL` file will be generated only if specific tolerances are defined as discussed in the section titled *Tolerances*; if the `LAB` subfile is not specified in the list, a `LAB` file will be generated only if features are written out to define the contents of the `LAB` file.

The attribute names for an E00 file must be uppercase, and must not exceed 16 characters in length. One should be aware that there are a few peculiar character sequences used by the E00 writer, for historical purposes. A trailing underscore character on an attribute name is replaced by a hash character ("`#`"), and a trailing sequence of "`_ID`" is replaced with "`-ID`". Thus, attributes named `JOES_` and `JOES_ID` on the `E00_DEF` line of a mapping file would be called `JOES#` and `JOES-ID` in the resulting info file.

One can work around this limitation by inserting single quotes around the underscore in the name of the attribute on the `DEF` line. For example, an attribute listed as

JOES'_'ID on an E00_DEF line would result in an attribute named JOES_ID appearing in the info file, instead of an attribute named JOES-ID.

The following table shows the attribute types that are supported.

Attribute Type	Description
char(<width>)	Fixed-length character string. The <code>width</code> parameter controls the maximum number of characters that can be stored in the field. When a character field is written, it is right-padded with blanks, or truncated, to fit the width. <code>width</code> must be between 1 and 320, inclusive.
char	Character string with a default maximum length (currently set to 320). This type should be used only for testing purposes, and not for production mapping files; for most cases, use the <code>char(<width>)</code> form above.
date	Character string representing a date. Attributes of type <code>date</code> must have exactly eight characters, and be of the form YYYYMMDD, where YYYY is the year, MM is the month (01-12), and DD is the day of the month (01-31).
int int(<width>)	Integer field. The optional <code>width</code> parameter specifies the display width of the field within ArcInfo. <code>width</code> must be between 1 and 16, inclusive. Representable numbers are those in the range of -999,999,999,999 to 9,999,999,999,998, inclusive.
number (<width>, <dec>)	Numeric data displayed with a field <code>width</code> of <code>width</code> and <code>dec</code> decimal positions. The value of <code>width</code> must allow for any minus sign and decimal point in the number, and must be in the range of 1 to 16, inclusive. The value of <code>dec</code> must be between 0 and 14, inclusive.
binint binint(<size>) bin- int(<size>, <width>)	Integer value, to be stored in ArcInfo as a binary number instead of character data. If the optional <code>size</code> parameter is specified, it specifies the number of bytes of storage (2 or 4 bytes) ArcInfo will use to store the value. The optional <code>width</code> parameter specifies the display width for ArcInfo to use. The <code>size</code> will default to 4bytes, and the display <code>width</code> will default to 5 for a 4-byte integer, or 4 for a 2-byte integer. <code>size</code> must be either 2 or four. <code>width</code> may be any integer between 1 and 6 when <code>size</code> is 2, or between 1 and 11 if <code>size</code> is 4.
float float(<width>, <prec>)	Floating point number, to be stored in ArcInfo as a four-byte binary number instead of as character data. The <code>width</code> and <code>prec</code> parameters define the display width and number of decimals for ArcInfo to use. A <code>float</code> field retains up to 9 digits of precision; only zero and numbers with a magnitude between 1.175494351e-38 and 3.402823466e+38 may be represented as <code>float</code> values.

Attribute Type	Description
double double(<width>, <prec>)	Floating point number, to be stored in ArcInfo as an eight-byte binary number instead of as character data. The <code>width</code> and <code>prec</code> parameters define the display width and number of decimals for ArcInfo to use. A <code>double</code> field retains up to 17 digits of precision; only zero and numbers with a magnitude between 2.225073858507201e-308 and 1.7975931348623158e+308 may be represented as <code>double</code> values.
redefined(<offset>, <length>, <fieldName>)	A redefined field specifies a subfield of another field within the same info file. The features written to an info file with redefined fields do not actually have attributes named for the redefined fields; the resulting E00 file defines the field in such a way that ArcInfo interprets the value of the redefined field as byte positions <code>offset</code> to <code>(offset + length - 1)</code> of the specified <code>fieldName</code> . Offsets are zero-relative, so an offset value of 1 actually refers to the second character of the named field.

An attribute's type parameter may optionally be followed by the literal string `, INDEX`, such as in:

```
E00_DEF ROADS \
  NAME      char(16) \
  .ARC:IDENT binint, INDEX
```

This indicates that particular field is an index in the ArcInfo info file.

Normally, any attributes provided on the `DEF` line will be applied to all info files created with the exception of the `.BND` and `.TIC` files, which have a specific format. However, the `DEF` line supports an attribute naming convention which allows a particular attribute to be applied to a specific info file. If the attribute name in the `DEF` line is of the form `<infoFile>:<attrName>`, then the attribute `attrName` will be added only to the info file specified by `infoFile`. There are actually three ways to specify an `infoFile`, as outlined by the following table.

infoFile Specification	Application
<baseName>.<suffix>	Attribute belongs to the precise info file named, and no others, for example, <code>HYDRO.TATANNO</code> .
.<suffix>	Attribute belongs to all info files with the specified suffix, for example, <code>.TATANNO</code> , <code>.PAT</code> .
.TAT	Attribute applies to all info files with a suffix that starts with <code>.TAT</code> . This is a special case provided solely for text attributes. Since text attribute files have the suffix <code>.TAT<annoLayer></code> , with the possibility of having several <code>annoLayers</code> in a single E00 file, this syntax allows the definition of attributes to be applied to the text attribute files for all annotation layers.

If an info file has specific attributes defined on it using the above `<infoFile>:<attrName>` syntax, then it will not have any of the attributes listed with the normal syntax, that is, `<attrName>` only.

Furthermore, the special info files `.TIC` and `.BND` always have the same predefined attributes, no matter what the contents of the `DEF` line for the E00 file are. Each of the special geometry-related attribute files (`.AAT`, `.PAT`, and `.TAT`) also has a default set of attributes which will always be present in the info file, but in these cases, the set of attributes will be supplemented with the appropriate attributes specified on the `DEF` line.

If an attribute name starts with a "-" character, then the specified attribute is removed. For example, if an attribute named `.TAT:-OFFSETX` is specified on the `DEF` line, then the specified attribute is not included in the resulting info file. This allows you to remove default attributes from standard info files.

If an attribute name starts with a "+" character, then its type will override the type of any of the standard attributes with the same name.

Controlling E00 Output on page 589 describes what info files will be automatically created when geometric entities are directed at E00 files, and how to generate custom info files.

TOLERANCES

Required/Optional: *Optional*

An E00 file or coverage can contain a subfile which defines tolerances used within ArcInfo. There are exactly ten tolerances defined in this file. Each tolerance has a value and a state. The FME refers to these tolerances by name or by number, as described in the *Tolerance Values* on page 583.

The `TOLERANCES` keyword tells the FME to create a `TOL` subfile with a specific value or state for a particular tolerance value. The syntax of a tolerance specification is:

```
<WriterKeyword>_TOLERANCES <id>=<val>[,<state>][[:<id>=val[,<state>]]+</pre>

```

where `id` is a valid tolerance number or name from the table in *Tolerance Values* on page 583, `val` is the specified tolerance's new value, and the optional `state` parameter is either 1 or 2, to specify whether the tolerance has been verified. If the state is not specified, a default value of 1 is used.

The E00 writer always generates a `TOL` file in each generated E00 file or binary coverage unless the `E00_FORCE_OUTPUT` option is specified in the coverage's `DEF` line, and `TOL` does not appear in the list of subfiles to force. If a `TOL` file is generated, any tolerances not specified by the `TOLERANCES` keyword will take the following default values and states:

Tolerance Identifier	Default Value	Default State
1 (FUZZY)	1.0e-20	1
2 (GENERALIZE)	0	2
3 (NODE_MATCH)	0	2
4 (DANGLE)	0	1

Tolerance Identifier	Default Value	Default State
5 (TIC_MATCH)	0	2
6 (EDIT)	1.0e-18	2
7 (NODESNAP)	1.0e-19	2
8 (WEED)	1.0e-19	2
9 (GRAIN)	1.0e-19	2
10 (SNAP)	1.0e-19	2

PRECISION

Required/Optional: *Optional*

All subfiles in a given E00 file or coverage are written out with either single-precision numbers or double-precision numbers. The `PRECISION` keyword takes a value of `single` or `double`, and specifies the precision used for all subfiles of all E00 files written.

Value: `SINGLE` | `DOUBLE`

Default Value: `DOUBLE`

Workbench Parameter: [<WorkbenchParameter>](#)

COMPRESSION

Required/Optional: *Optional*

E00 files may be written out uncompressed or they may have one or two levels of compression applied to them. Compressed files take up far less space than uncompressed files but they are impossible to inspect manually and are not readable by many systems.

The `COMPRESSION` directive allows you to specify how much compression to apply to a file.

Note: *FME ESRI Edition* will also generate binary ArcInfo coverages directly, if the `COMPRESSION` keyword is given a value of `BINARY`.

Value: `NONE` | `PARTIAL` | `FULL`

Default Value: `NONE`

Workbench Parameter: [<WorkbenchParameter>](#)

MAX_OUTPUT_SIZE

Required/Optional: *Optional*

When writing out to an E00 file, the user might want to break the file into an `.e00`, `.e01`, `.e02`, etc., based on the size of the file being written. This may be done by specifying `MAX_OUTPUT_SIZE` directive. The argument is the maximum size of each file produced.

The argument is specified as a number, optionally suffixed with a lower case "b" (bytes), "k" (kilobytes), or "m" (megabytes). If there is no such suffix, "bytes" will be assumed. For example:

```
E00_MAX_OUTPUT_SIZE 1024k
```

will produce no file greater than a megabyte.

When expressed in kilobytes or megabytes, the size is measured using base-2 measurements, hence a kilobyte 1024 bytes, and a megabyte is 1024 kilobytes.

The default behaviour is not to limit the size of output E00 files.

Workbench Parameter: [<WorkbenchParameter>](#)

BYPASS_LINEAR_TOPOLOGY

Required/Optional: *Optional*

Normally, the FME will compute a line-node topology whenever it writes out an E00 file or binary coverage. This can be a very expensive operation, and is not always needed.

For instances where a fully built linear topology is not needed in the resulting coverage, one may use the `BYPASS_LINEAR_TOPOLOGY` keyword to disable this feature. A value of `YES` will disable the topology computation, and a value of `NO` will leave it enabled.

For example,

```
E00_BYPASS_LINEAR_TOPOLOGY Yes
```

will produce output files with no linear topology.

Note that this affects linear output only. It is not possible to create a coverage of polygons which do not have topology connecting them to their linear boundaries. (Such representation is not possible in E00 files.)

Value: `YES` | `NO`

Default Value: `NO` (FME will compute a linear topology)

Workbench Parameter: [<WorkbenchParameter>](#)

PRESERVE_CASE

Required/Optional: *Optional*

When set to "Yes", the output files will have the same case as the feature types specified on the DEF line. When set to "No" it will demote the feature type to lowercase. For example, if the feature type is `POINT`, it will be output as `POINT` if the directive is set to "Yes" and as `point` if the directive is set to "No".

If this directive is missing, then it will have an implied value of "No". This is to ensure backwards compatibility, so workspaces created with a previous version of FME (one that does not yet support this directive) continue to work as they always have.

Example:

```
E00_PRESERVE_CASE Yes
```

Value: YES | NO

Default Value: Yes

Workbench Parameter: <WorkbenchParameter>

Feature Representation

This section discusses the way geometry and attributes are defined on features which represent the records in the various files within an E00 file. There is quite a difference between the features that the E00 reader emits, and those formed from the generic, automatically-generated mapping file. This discussion focuses primarily on the raw output from the E00 reader. *Generated Mapping Files* on page 585 provides a description of the feature that the generated mapping file creates.

There is also a difference between the features that the reader emits and those which the writer expects to be given to write out. Most notably, the reader uses the name of the subfile as the feature type of each FME feature read from an E00 file, whereas the writer uses the FME feature type to determine the name of the E00 file to which the feature will be written. In addition, the reader defines certain attributes on features read from E00 files – such as, `E00_FEAT_ROLE` and `E00_RECORD_NUM` – which are unused by the E00 writer.

In addition to the generic FME feature attributes that FME Workbench adds to all features (see *About Feature Attributes* on page 7), this format adds the format-specific attributes described in this section.

Geometry Composition

There are essentially four types of geometry defined in E00 files:

- arcs (lines)
- points
- polygons
- text

The geometries are formed by forming relations between certain standard subfiles and certain info files. The reader itself does not form these relations, but provides the attributes on the features allowing the mapping file to form the relations. Mapping files which have been automatically generated to read from E00 files form the necessary joins between the subfiles and the info files, and are a good starting point when creating custom mapping files to read E00 data. A description of the features output from automatically generated mapping files is given in the subsection titled *Generated Mapping Files*.

The following table summarizes the geometry types and lists the additional attributes required to fully define the geometry, as is the case for text features.

Geometry Type	Description	Additional Attributes Required
e00_arc	A string of geographic points that does not join or cross with itself. Features read from the ARC subfile contain arcs as their geometry.	None
e00_poly	A solid area, with an outer boundary and zero or more holes. No features is given a polygon geometry by the reader. They must be formed by factories in the mapping file.	None
e00_point	A simple (x, y) coordinate. The E00 reader creates points for features read from CNT and LAB standard subfiles.	None
e00_text	Defines annotation text at a particular location, with a rotation measured in degrees counterclockwise from the horizontal and text height measured in ground units. The geometry portion of a text feature is the single (x, y) point that defines its position.	e00_text_string e00_rotation e00_text_height
e00_no_geom	Feature has no associated geometry.	None

Feature Types

The E00 reader and the E00 writer view feature types somewhat differently. The features emitted from the E00 reader have a feature type dependent on the subfile or the info file from which the feature originated, whereas the E00 writer uses a feature type based on the name of the E00 file.

The following table summarizes the feature types generated for each subfile. If a subfile name in the table below contains an asterisk, then it is really a pattern to match info file names. This convention is required because the names of info files vary from coverage to coverage. The + symbol is used for an alternate asterisk for files containing two wildcard expressions. Therefore, the info file defining text attributes for the ERR annotation layer of the HYDR_SUR coverage is named HYDR_SUR.TATERR, and is referred to as *.TAT+ in the following table. References in the rest of the table row expand * to HYDR_SUR and + to ERR.

Note that the Reader Feature Type listed in the table applies only when reading E00 data, not when writing it. As discussed above, the E00 interprets the feature type as

the name of the target E00 file. The subsection titled *Controlling E00 Content* describes a method for redirecting features to a particular subfile.

Subfile Name	Reader Feature Type	Geometry	Additional Attributes
ARC	e00_arcdef	e00_arc	E00_FEAT_ROLE = "e00_arc_def" LPOLY = <id of left polygon> RPOLY = <id of right polygon> FNODE = <id of start node> TNODE = <id of end node> cover_id = <id of arc in coverage> cover_num = <sequence # of arc in coverage>
CNT	e00_centroid	e00_point	E00_FEAT_ROLE = e00_poly_cnt
LAB	e00_label	e00_point	E00_FEAT_ROLE = "e00_label" poly_id = <id of polygon containing label> boundingBoxMin.x = <min x of bounding box> boundingBoxMin.y = <min y of bounding box> boundingBoxMax.x = <max x of bounding box> boundingBoxMax.y = <max y of bounding box>
LOG	e00_log	e00_no_geom	text = <whole line of text from log file>
MTD	e00_mtd	e00_no_geom	FME currently skips this subfile
PAL	e00_polyarc	e00_no_geom	E00_FEAT_ROLE = "e00_poly_arc" arcnum{n} = <record number of ARC record for segment #n> arc{n}.arcnum = <record number of ARC record for segment #n> arc{n}.fnode = <start node of ARC record for segment #n> arc{n}.lpoly = <left polygon id of ARC record for segment #n> boundingBox.minX = <min x coordinate of bounding box> boundingBox.minY = <min y coordinate of bounding box> boundingBox.maxX = <max x coordinate of bounding box> boundingBox.maxY = <max y coordinate of bounding box>

Subfile Name	Reader Feature Type	Geometry	Additional Attributes
PRJ	e00_projection	e00_no_geom	<p>E00_FEAT_ROLE = "e00_proj"</p> <p>datum = <Name of datum></p> <p>projection = <Name of projection></p> <p>units = <Unit type></p> <p>xshift = <Shift in x coordinate></p> <p>yshift = <Shift in y coordinate></p> <p>zunits = <YES/NO></p> <p>zone = <UTM zone number></p> <p>unknown_param{n}.name = <name of non-standard parameter #n></p> <p>unknown_param{n}.value = <value of non-standard parameter #n></p>
RPL	e00_polyarc	e00_no_geom	<p>E00_FEAT_ROLE = "e00_region_arc"</p> <p>arcnum{n} = <record number of ARC record for segment #n></p> <p>arc{n}.arcnum = <record number of ARC record for segment #n></p> <p>arc{n}.fnode = <start node of ARC record for segment #n></p> <p>arc{n}.lpoly = <left polygon id of ARC record for segment #n></p> <p>boundingBox.minX = <min x coordinate of bounding box></p> <p>boundingBox.minY = <min y coordinate of bounding box></p> <p>boundingBox.maxX = <max x coordinate of bounding box></p> <p>boundingBox.maxY = <max y coordinate of bounding box></p> <p>e00_region_subclass = <name of region subclass></p> <p>(See the subsection titled <i>Region Support</i> for a description of region-related records.)</p>
RXP	e00_regionxref	e00_no_geom	<p>E00_FEAT_ROLE = "e00_region_xref"</p> <p>name = <name of tolerance type></p> <p>id = <numeric id of tolerance type></p> <p>state = <state of tolerance></p> <p>value = <value of tolerance></p> <p>(See the subsection titled <i>Region Support</i> for a description of region-related records.)</p>
TOL	e00_tolerance	e00_no_geom	<p>E00_FEAT_ROLE = "e00_tolerance"</p> <p>name = <name of tolerance type></p> <p>id = <numeric id of tolerance type></p> <p>state = <state of tolerance></p> <p>value = <value of tolerance></p> <p>(See the subsection titled <i>Tolerances</i> for a description of tolerance records.)</p>

Subfile Name	Reader Feature Type	Geometry	Additional Attributes
TXT	e00_text	e00_text	E00_FEAT_ROLE = "e00_text_def" e00_anno_name = "" e00_anno_id = <record number within TXT file> <Attributes for text geometry> (See the <i>Text Representation</i> subsection for a discussion about text geometry.)
TX6 or TX7	e00_text	e00_text	E00_FEAT_ROLE = "e00_text_def" e00_anno_name = <name of anno subclass> e00_anno_id = <position within anno subclass> e00_anno_level = <level number of text feature> e00_num_coords = <number of coordinates defining text position> parameter{} = <unnamed TX6/TX7 parameters> <Attributes for text geometry> (See the <i>Text Representation</i> subsection for a discussion about text geometry.)
	e00_textarrow	e00_arc	E00_FEAT_ROLE = "e00_text_arrow" e00_anno_name = <name of anno section> e00_anno_id = <position within anno section>
LNK	LNK	e00_point	E00_FEAT_ROLE = "LNK"
*.AAT	*_arcattr	e00_no_geom	E00_FEAT_ROLE = "e00_arc_attr" FNODE_ = <Start node cover#> TNODE_ = <End node cover#> LPOLY_ = <Left polygon cover#> RPOLY_ = <Right polygon cover#> *_ID = <arc identifier> *# = <coverage # of arc> LENGTH = <length of arc> <User-defined attributes>
*.BND	*_bounds	e00_no_geom	E00_FEAT_ROLE = "e00_bounds" XMIN = <min x of bounding box> YMIN = <min y of bounding box> XMAX = <max x of bounding box> YMAX = <max y of bounding box>
*.PAT	*_polyattr	e00_no_geom	E00_FEAT_ROLE = "e00_poly_attr" AREA = <area of polygon> PERIMETER = <perimeter of polygon> *_ID = <id of polygon> *# = <coverage # of polygon> <User-defined attributes>

Subfile Name	Reader Feature Type	Geometry	Additional Attributes
	*_pointattr	e00_no_geom	E00_FEAT_ROLE = "e00_point_attr" AREA = 0.0 PERIMETER = 0.0 *_ID = <id of point> *# = <coverage # of point> <User-defined attributes>
*.TIC	*_tic	e00_no_geom	E00_FEAT_ROLE = "e00_tic_point" IDTIC = <TIC point identifier> XTIC = <TIC point x coordinate> YTIC = <TIC point y coordinate>
*.TAT+	*+_textattr	e00_no_geom	E00_FEAT_ROLE = "e00_text_attr" e00_anno_name = <name of annotation layer> *# = <coverage number for text>
*.XCODE	*_textattr	e00_no_geom	E00_FEAT_ROLE = "e00_text_attr" e00_anno_name = "" *# = <coverage number for text>
*.+	*.+	e00_no_geom	E00_FEAT_ROLE = ".+" <User-defined attributes>

In addition to the attributes shown in this table, all features read from an E00 file have an attribute named `E00_RECORD_NUM`, whose value corresponds to the position within the subfile of the record defining the feature. The record numbers start at 1 for each file, and are incremented for each record. This number provides the positional information required to define the relationships between certain geometries and their attributes.

Note that the numbering of the text features is somewhat special. See *Text Representation* on page 581 for further details. Note that the reader also assigns features of most feature types an `E00_FEAT_ROLE` attribute, which defines the role of the feature within the coverage. This is required to make it easier to create a generic mapping file, when different files processed by that mapping file might have different info file names. For example, the file `BART.E00` might have an info file named `BART.TIC` where `JOSIE.E00` has an info file named `JOSIE.TIC`. The features emitted for these two info files would have a type of `BART_tic` and `JOSIE_tic`, respectively, but the features for both info files would have the value of `e00_tic_point` for their `E00_FEAT_ROLE` attribute. The role is given to features from the standard subfiles, as well as the info files which have one of the known suffixes - `.AAT`, `.BND`, `.PAT`, `.TIC`, `.TAT+`.

If features from a subfile have a particular type of geometry, then they will have an attribute named `e00_type`, whose value is the geometry type. For example, features from the `ARC` subfile will have line geometry attached, and will have an `e00_type` attribute with the value `e00_arc`.

Text Representation

The main geometry for text features are defined from records in the `TX6`, `TX7`, or `TXT` subfiles of the E00 coverage. This geometry consists of a text string, a location at which to draw the text, and optionally a string of points that form a curved line along which

to place the characters. Additionally, text features from the TX6 or TX7 subfile might have arrows associated with them¹.

When these features are read into the FME, the form changes slightly. If the keyword `<ReaderKeyword>_TEXT_CURVE` has been given the `IGNORE` value, the start and end points of the text line are used to compute the average rotation of the characters, and the first point in the line becomes the text's position. The text feature's geometry is a point which defines the position, along with the following attributes to define the rest of the feature.

Attribute Name	Description
e00_anno_name	Name of annotation layer (subclass) containing text.
e00_anno_id	Sequence number of text features within its annotation layer.
e00_rotation	Rotation of text display, measured in degrees counterclockwise from horizontal.
e00_text_height	Height of one line of text, measured in ground units.
e00_text_width	Height of the line of text, measured in ground units. When features contain multiple lines of text, this will be the width of the longest line of the text.
e00_tbox_height	Height of entire text block, measured in ground units. If the text contains carriage return characters, and thus spans multiple lines, this number will be greater than the value for <code>e00_text_height</code> ; otherwise the two will have the same value.
e00_text_string	String of text being displayed.
e00_text_just	(Optional) Justification of text feature relative to its baseline. This is an integer with a value between 1 and 12, inclusive. The default justification value will be "1", indicating that the bottom of the text character is aligned with the first point of its defining arc.
e00_num_coords	(Optional) When writing TX6 or TX7 features, this attribute defines how many coordinates are to be used to define the text's position. Its value is an integer between 1 and 3 (inclusive); if no <code>e00_num_coords</code> attribute is present, three coordinates will be used to represent the text location.
e00_text_level	(Optional) Numeric value representing the level of the text feature.
e00_text_symbol	(Optional) Numeric value representing the symbology ArcInfo will use to render the text.

If the keyword `<ReaderKeyword>_TEXT_CURVE` has been given the `FIT` value, and the text feature is defined by more than two coordinates, the FME computes a position and

1. Note that the E00 writer does not currently support any form of text arrow associated with text features. This capability may be added in a later release. Neither does the writer support TXT-style text records; only TX6-style text may be generated.

rotation of each character of text, generating a separate feature for each character. (No features are generated for whitespace characters.) In this case, all features corresponding to a given ArcInfo text element will have identical values for the `e00_anno_name` attribute, and for the `e00_anno_id` attribute, and will also contain two additional attributes:

Attribute Name	Description
<code>e00_whole_text_string</code>	The original text string, from which this feature's single character was taken.
<code>e00_pos_in_whole_text</code>	The position of this feature's character within the original text string. The first character has a position of "1", the second has a position of "2", and so on.

The contents of a `TX6` or `TX7` subfile within an E00 coverage may contain annotation in several different annotation layers (subclasses). Each feature belongs to one subclass, and this subclass' name is contained in the feature's `e00_anno_name` attribute. The features within a given subclass are numbered as they are read and the `e00_anno_id` attribute is assigned the feature's sequence number, starting at 1, within the layer.

If there are no named annotation subclasses in the coverage – as is always the case with annotation from the `TXT` subfile – all text features will have an empty string ("") as the value for their `e00_anno_name` attribute.

If the text has an associated arrow, a separate line feature is generated. This feature is type `e00_textarrow`, and contains the same values for its `e00_anno_name` and `e00_anno_id` attributes as the associated `e00_text` feature.

Every text feature defined in a `TX6` or `TX7` subfile of an E00 coverage has an associated set of user-defined attributes from a particular info file. Each record of the info file is returned from the E00 reader as a feature with the attributes defined on it. The features have an `E00_FEAT_ROLE` attribute of `e00_text_attr` and a feature type of `<prefix>_<anno_name>_textattr`, where `<prefix>` is an arbitrary prefix, and `<anno_name>` is the name of the annotation layer containing the feature. If the annotation layer is unnamed, the features defining the user attributes simply have a feature type of `<prefix>_textattr`.

The text geometries are associated with their user-defined attributes according to their position within the file. In other words, there is a one-to-one relationship between the text geometries and the features defining the user attribute. This relationship is formed by joining the text feature's `e00_anno_name` and `e00_anno_id` attributes with the attribute feature's `e00_anno_name` and `E00_RECORD_NUM` attributes.

Text features from `TXT` subfiles do not have named annotation subclasses, and consequently behave like text features from `TX6` or `TX7` files whose `e00_anno_name` contains an empty string. Note that the user attributes for text defined in the `TXT` subfile come from a different info file than those for text from the `TX6/TX7` subfile – `*.XCODE` rather than the `*.TAT+` info file – but the E00 reader forms the features generated from the two info files identically.

Tolerance Values

E00 coverages contain a list of ten tolerance values, which have a specific meaning within ArcInfo. Each tolerance has a numerical identifier in the range 1..10, a state, and a floating point value.

The E00 reader generates ten features from the `TOL` subfile. Each feature contains the following attributes:

Attribute Name	Description
<code>id</code>	Original numerical id given to the tolerance.
<code>name</code>	A standard name given to the tolerance record. This name provides a description of the tolerance in question, and is really just a textual version of the above ID. (1=>FUZZY, 2=>GENERALIZE, 3=>NODE_MATCH, 4=>DANGLE, 5=>TIC_MATCH, 6=>EDIT, 7=>NODESNAP, 8=>WEED, 9=>GRAIN, 10=>SNAP)
<code>state</code>	The state of the tolerance. (1=>tolerance has been verified, 2=>tolerance has not been verified)
<code>value</code>	The size of the tolerance. This is a floating point number, typically smaller than 1.0.

Projections

An E00 coverage may contain a subfile named `PRJ`, that defines the geographic projection of the coordinates within the coverage. The E00 reader gathers all of the information in this subfile into a single feature of the type `e00_projection`. The attributes of this feature are listed above, in the subsection titled *Feature Types*.

The `PRJ` subfile contains a list of named parameters, followed by a list of apparently unnamed parameters. Any of the named parameters, whose names are recognized, are defined as standard attributes - `datum`, `projection`, `units`, etc. on the `e00_projection` feature. Named parameters, whose names are not recognized by the reader, are placed into the attributes `unknown_parameter{}.name` and `unknown_parameter{}.value`. Unnamed parameters are placed into the attribute list `param{}.value`.

The E00 attempts to interpret the coordinate system information from the projection feature. If the parameters in the E00 file are from a known coordinate system or projection, the coordinate system information will be passed on the rest of the FME for normal processing. Otherwise a warning message will be logged in the log file. In either case, a feature describing the `PRJ` file will be passed-in to the FME feature stream.

Likewise, the E00 writer attempts to create a `PRJ` record from the coordinate system information attached to the features it is writing. The features must all belong to a single coordinate system for this to work. If no mapping can be found from the FME coordinate system to an E00 projection record, a warning will be written to the log file.

Region Support

The E00 reader provides complete support for reading ArcInfo regions, while the E00 writer currently has a very limited form of support for writing ArcInfo regions. An ArcInfo region is essentially a set of non-overlapping polygons with a logical connection. They are represented in the RPL and RXP subfiles, and their attributes are defined in the .PAT<subclass> info files.

The RPL file is virtually identical in structure to the PAL file. It defines the polygons which make up the various regions by listing the arcs that make up the polygons' boundary and holes. The only real difference in structure between the PAL and the RPL is that regions may be divided into subclasses, so the contents of the RPL file are likewise divided into subclasses, with all polygons belonging to regions of a single subclass appearing together within the RPL file.

The RXP file defines the actual regions by cross-referencing the polygons defined in the RPL file with region IDs. Each data line of the RXP file contains a region ID and an RPL polygon ID. The records of the RXP file are grouped by subclass.

Regions' attributes are stored in info files named <baseName>.PAT<subclass>, much as the text attributes are stored in <baseName>.PAT<subclass>. The records of the region attribute tables contain <baseName>-ID attribute that relates them to the regions defined in the RXP file.

Mapping files generated by FME to read E00 include the necessary factories to fully recreate complete regions and output them as area features.

To write regions out to an E00 file, the mapping file must define FME features that represent the RPL, RXP, and .PAT<subclass> records as they are to appear in the output file. This means that the every aspect of the regions, from the topology down to the assignment of IDs, and the ordering of the records with each subfile, must be performed in the mapping file; the E00 writer will simply write out whatever information it is given. In most cases it will be quite difficult to define this information in the mapping file. At some point in the future, the E00 writer will actually take care of computing the various regions and defining the contents of these records, but for the time being it simply provides a way to format the information computed elsewhere.

Info Files

Unlike the *standard* subfiles, whose names and formats are common to all E00 files, the info files' names and data structures vary from one coverage to another. Each info file starts with a header that defines its name and attributes on each record of the file.

The name of the info file is in the form <prefix>.<extension>, where <prefix> is arbitrary and <extension> defines the role of the records of the info file. Typically, all info files within a single E00 coverage have the same <prefix>. The <extension> is usually from a standard set, which includes the AAT (Arc Attribute Table), PAT (Point or Polygon Attribute Table), and BND (coverage bounding box). The E00 reader uses the extension to determine a role for the content of this info file.

Each record of the info file is interpreted by the E00 reader as an FME feature with no geometry. The <extension> of the info file's name is used to define the feature type and the value of the E00_FEAT_ROLE attribute of these features. The attributes defined on the record as specified in the info file's header are defined verbatim on the output feature.

Generated Mapping Files

Mapping files generated by the FME to read E00 files manipulate and join the features output from the E00 reader to form fully-formed, fully-attributed features with arc, point, polygon, or text geometry. The following sections explain each type of output feature and how it is put together.

Each coverage also contains a single polygon feature defining the bounding box of the coverage, and usually a set of four point features representing the TIC points. These features have polygon and point geometries, respectively, with the feature types `<prefix>_bounds` and `<prefix>_tic`.

Mapping files generated by the FME to write E00 files will only write one type of geometry – point, text, arc, or polygon – to each E00 output file. It will also calculate a bounding box of all features for each E00 file's BND subfile, and use the corners of this bounding box to define the TIC points.

Arc Features

In ArcInfo, arcs are simply polyline features with attributes to define a topology, as well as user-defined attributes. The geometry comes from the `e00_arcdef` features, originating from the ARC subfile and the attributes come from the `e00_arc_attr` features, originating from the `<prefix>.AAT` info file. Typically, the attributes defining the topology – left polygon, right polygon, from node, to node – are also defined in the info file, and will appear as attributes on the resulting arc features.

The arc features have a feature type of `<prefix>_arc`, where `<prefix>` is the prefix from the info file name. The attributes defined on `<prefix>_arc` features are summarized in the following table.

Attribute Name	Attribute Value
<code>e00_type</code>	<code>e00_arc</code>
<code><prefix>-ID</code>	Numerical identifier for arc feature.
<code><prefix>_</code>	Sequence number of arc feature within the E00 file.
<code>LENGTH</code>	Length of the line, measured in ground units.
<code>FNODE_</code>	Sequence number of starting node of the line.
<code>TNODE_</code>	Sequence number of ending node of the line.
<code>LPOLY_</code>	Sequence number of the polygon that lies to the left of the line when travelling from <code>FNODE</code> to <code>TNODE</code> .
<code>RPOLY_</code>	Sequence number of the polygon that lies to the right of the line when travelling from <code>FNODE</code> to <code>TNODE</code> .

In addition, any other attributes defined in the `<prefix>.AAT` info file are defined on the `<prefix>_arc` features generated with this mapping file.

Point Features

Point features are generated when the E00 coverage contains a LAB subfile, but no PAL subfile. In this case, the e00_label features originating from the LAB subfile are joined with the attributes of the e00_point_attr features originating from the <prefix>.PAT info file. The resulting point features have a type of <prefix>_point and the attributes from the following table.

Attribute Name	Attribute Value
e00_type	e00_point
<prefix_ID>	Numerical identifier for point feature.
<prefix>_	Sequence number of the point feature.
PERIMETER	0.0

In addition, any other attributes defined in the <prefix>.PAT info file are defined on the <prefix>_point features generated with this mapping file.

Polygon Features

Polygon features are the most complex of the features created by the generated mapping files. The polygon features result from combining four different types of features output from the E00 reader: e00_arcdef, e00_centroid, e00_polyarc, and e00_poly_attr. A combination of these features is performed as follows.

- The polylines defined by the e00_arcdef features in the ARC subfile form the edges of the polygons. They are combined to form each polygon and its holes, according to the contents of the arcnum{} attributes on each e00_polyarc feature.
- The point geometry from each e00_centroid feature is attached to the corresponding polygon, providing the values for the attributes e00_centroid_x and e00_centroid_y.
- The attributes from the e00_poly_attr features originating in the <prefix>.PAT info file are added to the formed polygon features.

The resulting polygon features have a type of <prefix>_poly and the attributes from the following table.

Attribute Name	Attribute Value
e00_type	e00_poly
<prefix_ID>	Numerical identifier for polygon feature.
<prefix>_	Sequence number of the polygon feature within the E00 file.
e00_centroid_x	X coordinate of polygon's centroid.
e00_centroid_y	Y coordinate of polygon's centroid.
PERIMETER	Outer perimeter of polygon.
AREA	Area of the polygon, measured in square ground units.

In addition, any other attributes defined in the <prefix>.PAT info file are defined on the <prefix>_poly features generated with this mapping file.

Text and Textarrow Features

There are two ways text features are formed in the automatically generated mapping files. The first, and most common, is by combining the text geometries from the TX6 or TX7 subfile with the attributes from the <prefix>.TAT<annoLayer> info file. In this case, the resulting text features have a feature type of <prefix>_<annoLayer>_text, or <prefix>_text if the annotation layer is unnamed. See *Text Representation* on page 581 for an explanation of annotation layers.

Some E00 coverages have their annotation defined in a TXT subfile rather than in a TX6 or TX7 subfile. These features are combined with the attributes of the <prefix>.XCODE info file instead of a <prefix>.TAT<annoLayer> subfile, and will always be contained in an unnamed annotation layer.

In either case, text features will have a feature type of <prefix>_text or <prefix>_<annoLayer>_text, depending on whether they are contained in a named annotation layer, and will have the attributes shown in the following table.

Attribute Name	Attribute Value
e00_type	e00_text
<prefix_ID>	Numerical identifier for text feature.
<prefix>_	Sequence number of the text feature within the E00 file.
e00_anno_name	Name of annotation layer containing text feature. This will be "" if it is in an unnamed annotation layer.
e00_anno_id	Sequence number of text feature within its annotation layer. This number starts at 1 for the first feature in each annotation layer and is incremented for every other feature.
e00_rotation	Rotation at which to display text, measured in degrees counterclockwise from horizontal.
e00_text_string	Textual portion of feature.
e00_text_height	Height of text, measured in ground units.
e00_text_level	Number indicating level of text.

In addition, any other attributes defined in the <prefix>.TAT<annoLayer> or <prefix>.XCODE info file are defined on the <prefix>_text features generated with this mapping file.

If the text geometry originates in the TX6 or TX7 subfile – as opposed to the TXT subfile – it might have a separate linear portion that acts as an arrow pointing from the text to another location. These lines are written out as features with a feature type of <prefix>_<annoLayer>_textarrow or <prefix>_textarrow, and attributes e00_anno_name and e00_anno_id which take the same values as the corresponding <prefix>_<annoLayer>_text or <prefix>_text features.

Occasionally, an E00 file will have `e00_text` features for which there are no corresponding attributes in the info files. In this case, the feature types of the corresponding text features generated are simply `text` and `textarrow`.

The E00 writer is not capable of generating `TEXT` features. Text output from the FME takes place with `TX6` or `TX7` records. See *Controlling E00 Output* on page 589 for a description of how geometry is formed on output E00 files.

Controlling E00 Output

The E00 writer allows easy generation of E00 files, but also provides a high level of customization to the format and content of the resulting E00 files. In its simplest form, the E00 writer takes FME features defining line, point, text, and polygon features, and writes them to the appropriate subfiles of the E00 file. This mode of operation makes it very easy to write a mapping file which creates E00 files:

- Decide on the names of the E00 files.
- For each file, decide on the type of geometry to go into the file, and the names and types of the attributes.
- Create a `DEF` line to define the attributes.
- Create the correlation line to direct the features at the appropriate files. Features going to an E00 file must contain an attribute named `e00_type`, with one of the values `e00_point`, `e00_arc`, `e00_text`, or `e00_poly`. The writer uses this attribute to determine how the features' geometry is written out. Some geometry might require additional attributes to be defined on the features being written – see the subsection titled *Geometry Composition* for more information.

This will generate all of the `ARC`, `LAB`, `CNT`, `PAL`, `TX6` or `TX7`, `*.BND`, `*.TIC`, `*.AAT`, `*.PAT`, and `*.TAT+` records needed to describe the features passed-in. This is normally all one will need to create E00 files. However, the E00 writer also contains mechanisms to allow the advanced user to:

- Define specific label locations for polygons.
- Explicitly direct a particular feature to a particular subfile.
- Create multiple info files within a single E00 file, each with its own set of attributes.

These mechanisms are described in the following subsections.

Specific Label Positions

When the E00 writer generates a `PAL` record to define a polygon, it also generates a centroid (`CNT`) record and a label (`LAB`) record. The position of the centroid and the label are normally computed to be some point within the polygon. (The computed location will always be inside the polygon, but not inside any hole of the polygon.) A specific location will be used for the location of the `CNT` and `LAB` records, instead of the computed location, if the E00 writer is given a feature with a point-in-polygon (PIP) geometry instead of a polygon or donut geometry. In this case, the point contained in the PIP will define the location of both the label and the centroid.

Explicit Subfile Selection

The E00 writer normally looks for the `e00_type` attribute to decide how to write out the features. If this attribute is not present, or has the value of `e00_no_geom`, the writer

will look for an attribute named `E00_SUBFILE` on the input feature. This tells the writer the subfile in which to write the record. This can be one of the standard subfiles (`ARC`, `LAB`, `TOL`, etc.) or any one of the info files.

This provides a powerful tool to the mapping file author. By changing a few attributes on an FME feature, that feature can be directed to almost any part of the E00 file.

There are two warnings associated with explicit subfile selection:

- The first is that the author of the mapping file must ensure that all of the attributes necessary to write the feature are present on the FME feature when it is given to the E00 writer. This is generally true in FME mapping files, but is of particular importance to this option. For the standard files, the attributes defined in the table shown in *Feature Types* on page 577 must be present on the features. Features destined for info files must provide values to all attributes defined on the info file. The following section, *Info File Creation*, explains how info file attribution works.
- The second is that the mapping file **must not** write features directly to a standard subfile or info file that is also being written to with the normal geometry writing. The writer contains an internal state to keep track of which geometries have been written to each subfile, and can become easily confused if other features are manually inserted into the same subfiles.

Info File Creation

The E00 normally assigns all attributes specified on the `DEF` line to all info files. This is usually not a problem, since an E00 file will typically just define some geometry of a single type, and associate attributes with each piece of geometry.

However, there are instances where the mapping file author will want more control over the format of the info file. For example, an input file of tabular data can be placed into a specific info file using the explicit subfile selection described in *Explicit Subfile Selection* on page 589. The normal means of specifying info file attributes on the `DEF` line will place the *same* attributes on every info file in the E00 file. If there is other information to be placed into the same E00 file, such as annotation (text features) and linear geometry (arc features). The other info files have to carry attributes from the tabular data's info file and vice-versa.

To overcome this, the `DEF` line can contain attribute definitions specific to each info file. The forms of syntax for this are listed in the *Writer Keywords* section titled *DEF*.

Suppose the tabular data in the example above is a simple table listing street names, and the minimum and maximum street numbers for each street. Using the normal `DEF` line syntax, the `DEF` line for the E00 file might normally be something like:

```
E00_DEF STREET \
  STREET_NAME      char(32) \
  MIN_ADDR         binint   \
  MAX_ADDR         binint
```

If the E00 writer were to write lines and text to the above E00 file, as well as directing tabular data to the `STREETS.TAB` info file using the mechanism described in *Explicit Subfile Selection* on page 589, the resulting E00 file would contain lines, text, and `STREETS.TAB` records, all with the attributes `STREET_NAME`, `MIN_ADDR`, and `MAX_ADDR`.

In contrast, the following DEF line would apply the STREET_NAME, MIN_ADDR, and MAX_ADDR attributes only to the STREETS.TAB records, leaving no attribution on the line or text records:

```
E00_DEF STREET \
  STREETS.TAB:STREET_NAME char(32) \
  STREETS.TAB:MIN_ADDR binint \
  STREETS.TAB:MAX_ADDR binint
```

This DEF line provides us with a more useful description of the data, but it does not give the line and text features any attributes which can be used to relate them to the STREETS.TAB attributes. The following DEF line will attach a STREET_NAME attribute to each line and text feature, as well as generating the same STREETS.TAB file listed above:

```
E00_DEF STREET \
  STREETS.TAB:STREET_NAME char(32) \
  STREETS.TAB:MIN_ADDR binint \
  STREETS.TAB:MAX_ADDR binint \
  STREET_NAME char(32)
```

Note: If you specify any attributes for a specific info file, then none of the "general" attributes will be added (other than the # and -ID attributes).

Finally, the following DEF line would create the same E00 file, except that the text features would be left with no attribution at all.

```
E00_DEF STREET \
  STREETS.TAB:STREET_NAME char(32) \
  STREETS.TAB:MIN_ADDR binint \
  STREETS.TAB:MAX_ADDR binint \
  .AAT:STREET_NAME char(32)
```

The E00 writer uses the following heuristic to decide which attributes are defined on a given info file:

1. The .BND and .TIC info files each have a predefined set of attributes and corresponding types which are **always** used, and never supplemented.
2. If the info file is one of the geometry-related info files – .AAT, .PAT, or .TAT+ – have a predefined set of attributes which are always present, but are supplemented by any attributes from **3** or **4** below.
3. If there are any attributes that were specified using the <infoFile>:<attrName> syntax, for this particular info file, they are appended to the info file definition.
4. If there were no info file-specific attributes in **3**, then any attributes which were specified with the normal <attrName> syntax will be appended to the info file definition.

