
Stream Format

Introduction	A-1
Stream File Description	A-2
Stream Data Types	A-3
Stream Records	A-7
Stream Syntax	A-24
Example of a Stream Format File	A-26

Introduction

When you want to transfer Virtuoso data to other systems, the best format to translate your data to is Stream format. This appendix describes the Stream format used in the files you can produce and read in using the *pipo* utility from a Unix command line, or the *Stream In* and *Stream Out* commands from the Translators menu in the CIW.

You can easily transfer libraries preserved in Stream format to other systems for processing. In addition, Stream format is upward compatible, which means newer releases of the Stream translators can read libraries produced with an older release.

For complete descriptions of the options and arguments for *Stream In*, *Stream Out*, and *pipo* commands, refer to [Chapter 3, “Translating Stream Files.”](#)

Although you can use the *Stream In* and *Stream Out* commands to translate your Stream data, you might want to directly edit your Stream file, or write programs or scripts that manipulate your Stream data. To help you understand the type and format of information your Stream file contains, this appendix describes the following:

- The composition of a Stream record
- Stream syntax
- The Stream format file

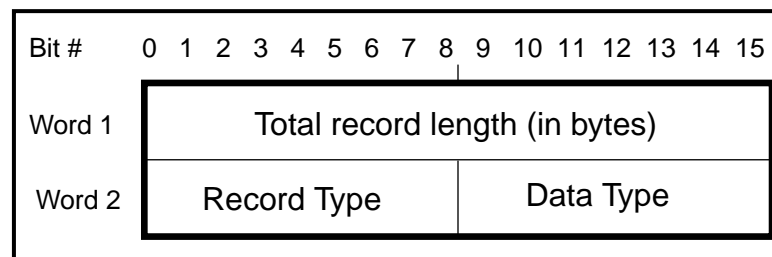
Note: Portions of this appendix describe features and data types that are applicable to the GDSII and Construct systems. These features might not apply to the Virtuoso system.

Stream File Description

The information stored in a Stream file is coded in variable length records. The length of a record is measured in eight-bit bytes. The minimum record length is four bytes. There is no limit on record length. Within the record, two bytes (16 bits) is a *word*. The 16 bits in a word are numbered 0 to 15, left to right.

The first four bytes of a record are the header. The first two bytes of the header specifies how many eight-bit bytes the record contains. The third byte of the header specifies the record type. The fourth byte of the header specifies the type of data contained within the record. The fifth through last bytes of the record are data. The next record begins immediately after the last byte in the record.

The following figure shows a typical record header.



If the Stream file is on a magnetic tape, the records of the library are usually divided in 2048-byte physical blocks. Records can overlap physical block boundaries; a record is not required to be wholly contained in a single physical block.

Two consecutive zero bytes are a *null word*. You can use null words to fill the space between the last record of a library and the end of its physical block.

Stream records are always an even number of bytes. If a record contains ASCII string data and the ASCII string is an odd number of bytes, the last character is a null character.

Stream Data Types

The following table lists the Stream data types and their values. The data type is specified by the fourth byte of the record.

Data type	Value
No data present	0
Bit Array	1
Two-Byte Signed Integer	2
Four-Byte Signed Integer	3
Four-Byte Real	4 (not used)
Eight-Byte Real	5
ASCII String	6

The following paragraphs describe these Stream data types. As a reminder, a word consists of 16 bits, numbered 0 to 15, left to right.

- *Bit Array (1)*

A bit array is a word which contains bits or group of bits that represent data. A bit array allows one word to contain more than one piece of information.

- *Two-Byte Signed Integer (2)*

2-byte integer = 1 word 2s-complement representation

The range of two-byte signed integers is -32,768 to 32,767.

The following is a representation of a two-byte integer, where *S* is the sign and *M* is the magnitude.

SMMMMMMM MMMMMMMM

The following are examples of two-byte integers:

```
00000000 00000001 = 1
00000000 00000010 = 2
00000000 10001001 = 137
11111111 11111111 = 1
11111111 11111110 = -2
11111111 01110111 = -137
```

- **Four-Byte Signed Integer (3)**

4-byte integer = 2 word 2s-complement representation

The range of four-byte signed integers is -2,147,483,648 to 2,147,483,647.

The following is a representation of a four-byte integer, where *S* is the sign and *M* is the magnitude.

SMMMMMMM MMMMMMMM MMMMMMMM MMMMMMMM

The following are examples of four-byte integers:

```
00000000 00000000 00000000 00000001 = 1
00000000 00000000 00000000 00000010 = 2
00000000 00000000 00000000 10001001 = 137
11111111 11111111 11111111 11111111 = -1
11111111 11111111 11111111 11111110 = -2
11111111 11111111 11111111 01110111 = -137
```

- **Four-Byte Real (4) and Eight-Byte Real (5)**

4-byte real = 2-word floating point representation

8-byte real = 4-word floating point representation

For all non-zero values:

A floating point number has three parts: the sign, the exponent, and the mantissa.

- The value of a floating point number is defined as:
(Mantissa) x (16 raised to the true value of exponent field).
- The exponent field (bits 1-7) is in Excess-64 representation. The field shows a number 64 greater than the actual exponent.
- The mantissa is always a positive fraction greater than or equal to 1/16 and less than 1. For a 4-byte real, the mantissa is bits 8 through 31. For an 8-byte real, the mantissa is bits 8 through 63. The binary point is just to the left of bit 8. Bit 8 represents the value 1/2, bit 9 represents 1/4, and so on.

To keep the mantissa in the range of 1/16 to 1, the results of floating point arithmetic are *normalized*. Normalization is a process that shifts the mantissa left one hex digit at a time until its left FOUR bits represent a non-zero quantity. For every hex digit shifted, the exponent is decreased by one. Since the mantissa is shifted four bits at a time, it is possible for the left three bits of the normalized mantissa to be zero. A zero value, also called *true zero*, is represented by a number with all bits zero.

The following are representations of 4-byte and 8-byte reals, where *S* is the sign, *E* is the exponent, and *M* is the magnitude. Examples of 4-byte reals are included on the following pages, although 4-byte reals are not used currently. The representation of the negative values of real numbers is exactly the same as the positive, except that the highest order bit is 1, not 0.

In the eight-byte real representation, the first four bytes are exactly the same as in the four-byte real representation. The last four bytes contain additional binary places for higher resolution.

4-byte real:

```
SEEEEEEE MMMMMMMM MMMMMMMM MMMMMMMM
```

8-byte real:

```
SEEEEEEE MMMMMMMM MMMMMMMM MMMMMMMM  
MMMMMMMM MMMMMMMM MMMMMMMM MMMMMMMM
```

Examples of 4-byte real:

In the first six lines of the following example, the 7-bit exponent field is 65. The actual exponent is $65-64=1$.

```
01000001 00010000 00000000 00000000 = 1
01000001 00100000 00000000 00000000 = 2
01000001 00110000 00000000 00000000 = 3
11000001 00010000 00000000 00000000 = -1
11000001 00100000 00000000 00000000 = -2
11000001 00110000 00000000 00000000 = -3
01000000 10000000 00000000 00000000 = 0.5
01000000 10011001 10011001 10011001 = 0.6
01000000 10110011 00110011 00110011 = 0.7
01000001 00011000 00000000 00000000 = 1.5
01000001 00011001 10011001 10011001 = 1.6
01000001 00011011 00110011 00110011 = 1.7
00000000 00000000 00000000 00000000 = 0
01000001 00010000 00000000 00000000 = 1
01000001 10100000 00000000 00000000 = 10
01000010 01100100 00000000 00000000 = 100
01000011 00111110 00000001 00000000 = 1000
01000100 00100111 00010000 00000000 = 10000
01000101 00011000 01101010 00000000 = 100000
```

- **ASCII String (6)**

A collection of bytes representing ASCII characters. All odd length strings are padded with a null character (the number zero), and the byte count for the record containing the ASCII

string includes this null character. If you write a program to read Stream data, the program must check for the null character and, if present, decrease the length of the string by one.

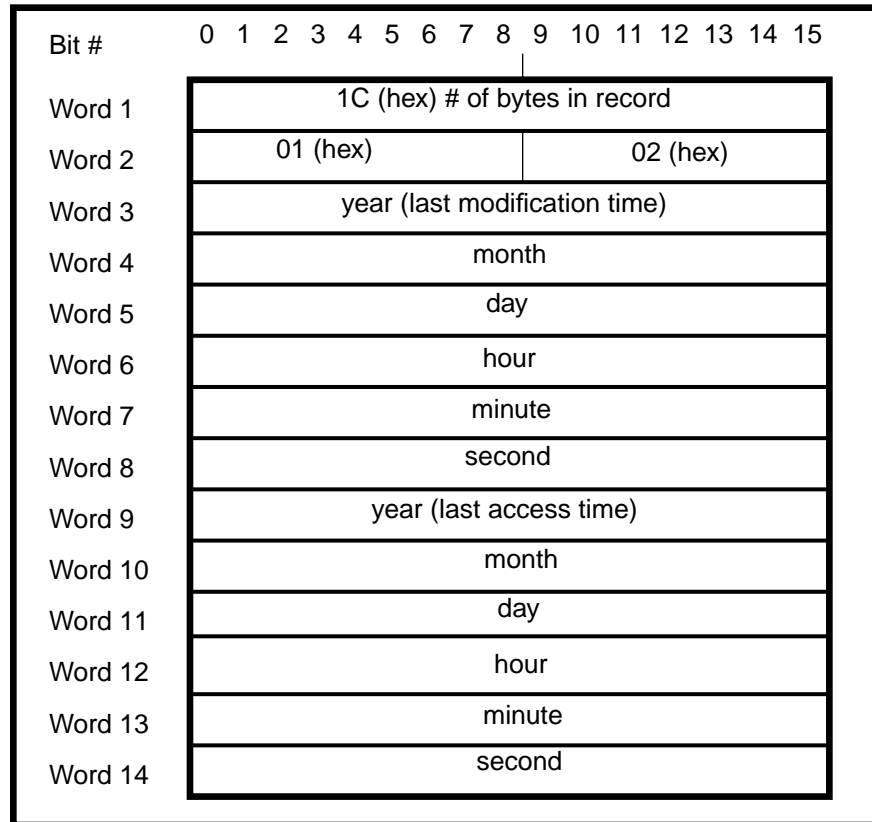
Stream Records

This section describes the records that make up a Stream file. The descriptions include the following:

- Record number
- Record name
- Data type
- The four digits which make up the second word in the record, divided as follows:
 - The first two digits are the hexadecimal value of the record number.
 - The second two digits are the numeric value for the data type.
- A description of the information contained in the record.

0	HEADER	Two-Byte Signed Integer [0002] Contains the Stream version number.
1	BGNLIB	Two-Byte Signed Integer [0102] Contains the last modification time of a library (one word each for year, month, day, hour, minute, and second), the time of last access (same format). This record type marks the beginning of a library.

The following figure shows the meaning of each word.



2 LIBNAME ASCII String

[0206] Contains the library name. The library name must follow UNIX filename conventions for length and valid characters. The library name can include the file extension (*.sf* or *.db* in most cases).

3 UNITS Eight-Byte Real

[0305] Contains two numbers indicating the number of database units in a user unit and the size of a database unit in meters. Typically, the number of database units in a user unit is less

than 1, because you use more than 1 database unit per user unit. To calculate the size of a user unit in meters, divide the second number by the first.

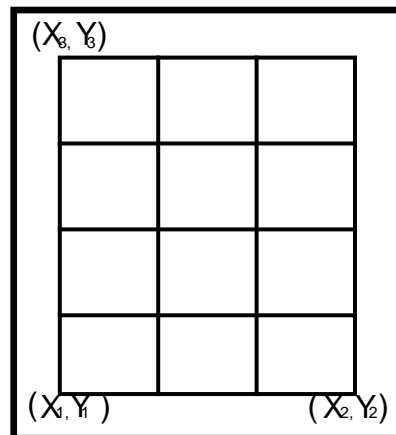
For example, if you create a library using the default unit (user unit = 1 micron and 1000 database units per user unit), the first number is .001 and the second number is 1E-9.

- | | | |
|---|----------|---|
| 4 | ENDLIB | No Data Present
[0400] Marks the end of a library. |
| 5 | BGNSTR | Two-Byte Signed Integer
[0502] Contains the creation time and last modification time of a structure (in the same format as the BGNLIB record), and marks the beginning of a structure. |
| 6 | STRNAME | ASCII String
[0606] Contains the structure name. A structure name can be up to 32 characters. Legal structure name characters are:
<ul style="list-style-type: none"> A through Z a through z 0 through 9 Underscore (_) Question mark (?) Dollar sign (\$) |
| 7 | ENDSTR | No Data Present
[0700] Marks the end of a structure. |
| 8 | BOUNDARY | No Data Present
[0800] Marks the beginning of a boundary element. |
| 9 | PATH | No Data Present |

		[0900] Marks the beginning of a path element.
10	SREF	No Data Present [0A00] Marks the beginning of an SREF (structure reference) element.
11	AREF	No Data Present [0B00] Marks the beginning of an AREF (array reference) element.
12	TEXT	No Data Present [0C00] Marks the beginning of a text element.
13	LAYER	Two-Byte Signed Integer [0D02] Specifies the layer number. The value of the layer must be in the range of 0 to 255.
14	DATATYPE	Two-Byte Signed Integer [0E02] Specifies the data type. The value of the datatype must be in the range of 0 to 255.
15	WIDTH	Four-Byte Signed Integer [0F03] Specifies the width of a path in database units. If the value for the width is negative, the width is absolute and is not affected by the magnification factor of any parent reference. If this record type is omitted, the default is zero.
16	XY	Four-Byte Signed Integer [1003] Contains an array of the XY coordinates for path, boundary, text, contact, SREF, node, box, and AREF elements. The coordinates are in database units. Each X or Y coordinate is four bytes long.

The elements have the following number of coordinates:

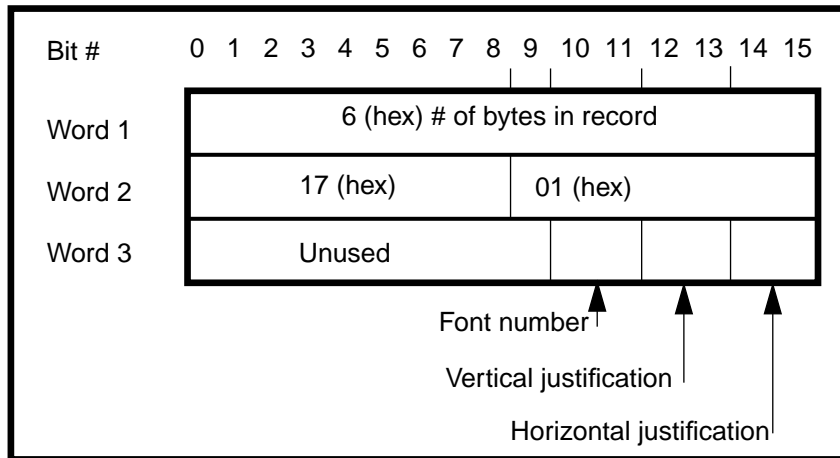
- Path elements have a minimum of 2 and a maximum of 200 coordinates.
- Boundary elements can have a minimum of 4 and a maximum of 600 coordinates. The first and last coordinates must coincide.
- A text, contact, or SREF element can have only one coordinate.
- A node can have from one to 50 coordinates.
- A box must have five coordinates, with the first and last coordinates coinciding.
- An AREF has exactly three coordinates. In an AREF, the first coordinate is the array reference point (origin point). The second coordinate locates a position that is displaced from the reference point by the inter-column spacing times the number of columns. The third coordinate locates a position that is displaced from the reference point by the inter-row spacing times the number of rows. The following is an example of an array lattice.



17 ENDEL No Data Present
 [1100] Marks the end of an element.

- 18 SNAME ASCII String
[1206] Contains the name of a referenced structure. See also STRNAME.
- 19 COLROW Two-Byte Signed Integer
[1302] Contains four bytes. The first two bytes contain the number of columns in the array. The third and fourth bytes contain the number of rows. The number of columns and the number of rows must be in the range 0 to 32,767 (decimal).
- 20 TEXTNODE No Data Present (Not currently used)
[1400] Marks the beginning of a text node.
- 21 NODE No Data Present
[1500] Marks the beginning of a node.
- 22 TEXTTYPE Two-Byte Signed Integer
[1602] Specifies the text type. The value of the text type must be in the range 0 to 255.
- 23 PRESENTATION Bit Array
[1701] Specifies how text is presented.
The bits in the bit array have the following values:
- Bits 10 and 11, used together as a binary number, specify the font (00 is font 0, 01 is font 1, 10 is font 2, and 11 is font 3).
 - Bits 12 and 13 specify the vertical justification (00 means top, 01 means middle, and 10 means bottom).
 - Bits 14 and 15 specify the horizontal justification (00 means left, 01 means center, and 10 means right).
 - Bits 0 through 9 are reserved for future use and must be cleared.

If this record is omitted, top-left justification and font 0 are the default values. The following shows a PRESENTATION record.



24 SPACING Not currently used

25 STRING ASCII String

[1906] Contains up to 512 characters of text to present.

26 STRANS Bit Array

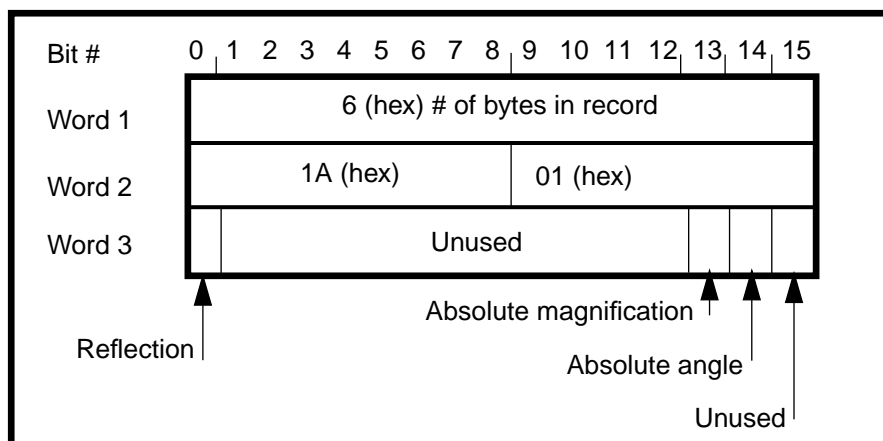
[1A01] Contains two bytes of bit flags for SREF, AREF, and text transformation. Bit 0 (the leftmost bit) specifies reflection.

The bits in the bit array have the following values:

- If bit 0 is set, the element is reflected about the X-axis before angular rotation. For an AREF, the entire array is reflected with the individual array members rigidly attached.
- Bit 13 flags absolute magnification.
- Bit 14 flags absolute angle.
- Bits 1 to 12 and 15 are reserved for future use and must be clear.

Note: Absolute magnification and absolute angle are not supported by Construct 1.0, and are interpreted as non-absolute values.

If this record is omitted, the defaults for the element are no reflection, non-absolute magnification, and non-absolute angle. The following shows an STRANS record.



27 MAG Eight-Byte Real

[1B05] Contains the magnification factor. If this record is omitted, the default magnification factor is one.

28 ANGLE Eight-Byte Real

[1C05] Contains the angular rotation factor. The angle of rotation is measured in degrees and in the counterclockwise direction.

For an AREF, the ANGLE rotates the entire array (with the individual array members rigidly attached) about the array reference point. If this record is omitted, the default angle is zero degrees.

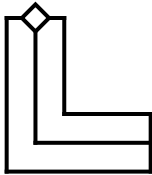
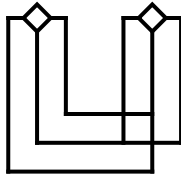
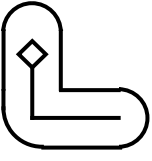
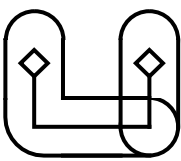
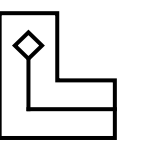
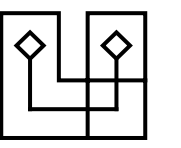
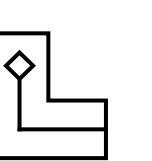
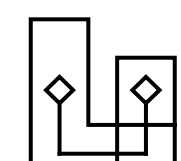


29 UNINTEGER Not currently used

User Integer data was used in GDSII Release 2.0 only.

- 30 USTRING Not currently used
- User String data, formerly called character string data (CSD), was used in GDSII Releases 1.0 and 2.0.
- 31 REFLIBS ASCII String
- [1F06] Contains the names of the reference libraries. This record must be present if any reference libraries are bound to the working library. The name of the first reference library starts at byte 5 (immediately following the record header) and continues for 44 bytes. The next 44 bytes contain the name of the second library. The record is extended by 44 bytes for each additional library (up to 15) which is bound for reference. The reference library names may include directory specifiers (separated with "/") and an extension (separated with "."). If either the first or second library is not named, its place is filled with nulls.
- 32 FONTS ASCII String
- [2006] Contains the names of the text font definition files. This record must be present if any of the four fonts have a corresponding text font definition file. This record must not be present if none of the fonts have a text font definition file. The filename for text font 0 starts the record, followed by the filenames for the remaining three fonts. Each filename is 44 bytes long. The filename is padded with nulls if the name is shorter than 44 bytes. The filename is null if no filename corresponds to the font. The filenames may include directory specifiers (separated with "/") and an extension (separated with ".").
- 33 PATHTYPE Two-Byte Signed Integer
- [2102] Contains a value indicating the type of path endpoints. The value is 0 for square-ended paths that end flush with their

endpoints, 1 for round-ended paths, 2 for square-ended paths that extend a half-width beyond their endpoints, and 4 for paths with variable square-ended extensions (see records 48 and 49). If not specified, assumes a type of 0.

The following illustration shows the path types.

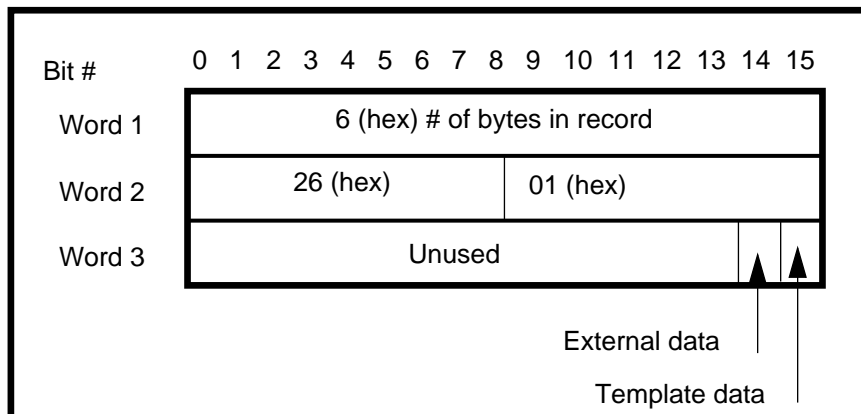
<p>Path type 0</p> 		<p>Path type 0</p> <p>Path type 0 produces a square-ended path, ending flush with the digitized endpoints. This is the default path type if none is specified.</p>
<p>Path type 1</p> 		<p>Path type 1</p> <p>Path type 1 produces a round-ended path. The two ends are semicircular with center at the digitized endpoints.</p>
<p>Path type 2</p> 		<p>Path type 2</p> <p>Path type 2 produces a square-ended path. The ends of the path extend beyond the digitized endpoints by one-half the path width.</p>
<p>Path type 4</p> 		<p>Path type 4</p> <p>Path type 4 produces a square-ended path. The ends of the path extend beyond the digitized endpoints or endpoints extend beyond the path by a variable, user-definable distance. The two possibilities are shown here:  </p>

- 34 GENERATIONS Two-Byte Signed Integer
[2202] Contains the number of copies of deleted or back-up structures to retain. This number must be at least 2 and not more than 99. If the GENERATIONS record is omitted, the default value is 3.
- 35 ATTRTABLE ASCII String
[2306] Contains the name of the attribute definition file. This record is present only if an attribute definition file is bound to the library. The attribute definition filename can include directory specifiers (separated with "/") and an extension (separated with "."). The maximum record size is 44 bytes.
- 36 STYPTABLE ASCII String (Unreleased feature)
[2406]
- 37 STRTYPE Two-Byte Signed Integer (Unreleased feature)
[2502]

38 ELFLAGS Bit Array

[2601] Contains two bytes of bit flags. Bit 15 (the rightmost bit) specifies Template data. Bit 14 specifies External data (also referred to as Exterior data). All other bits are currently unused and must be cleared to 0. If this record is omitted, the default value for all bits is 0.

The following illustration shows an ELFLAGS record.



39 ELKEY Four-Byte Signed Integer (Unreleased feature)

[2703]

40 LINKTYPE Two-Byte Signed Integer (Unreleased feature)

[28]

41 LINKKEYS Four-Byte Signed Integer (Unreleased feature)

[29]

42 NODETYPE Two-Byte Signed Integer

[2A02] Contains a value indicating the node type. The value of the node type must be in the range of 0 to 255.

- 43 PROPATTR Two-Byte Signed Integer
- [2B02] Contains the attribute number. The attribute number is an integer from 1 to 127. Attribute numbers 126 and 127 are reserved for the user integer and user string (CSD) properties which were used prior to Release 3.0.
- 44 PROPVALUE ASCII String
- [2C06] Contains the string value associated with the attribute named in the preceding PROPATTR record. The maximum string length is 126 characters. The attribute-value pairs associated with any one element must all have distinct attribute numbers. Also, the total amount of property data that can be associated with any one element is limited: the total length of all the strings, plus twice the number of attribute-value pairs, must not exceed 128. (Or, if the element is an SREF, AREF, contact, nodeport, or node, the length must not exceed 512.)
- For example, if a boundary element uses property attribute 2 with property value “metal,” and property attribute 10 with property value “property,” the total amount of property data is 18 bytes. This is 6 bytes for “metal” (odd-length strings must be padded with a null) plus 8 for “property” plus 2 times the 2 attributes (4), which equals 18.
- 45 BOX No Data Present
- [2D00] Marks the beginning of a box element.
- 46 BOXTYPE Two-Byte Signed Integer
- [2E02] Contains a value indicating the box type. The value of the box type must be in the range of 0 to 255.

- 47 PLEX Four-Byte Signed Integer
[2F03] A unique positive number which is common to all elements of the plex to which this element belongs. The head of the plex is flagged by setting the seventh bit; therefore, plex numbers should be small enough to occupy only the rightmost 24 bits. If this record is not present, the element is not a plex member.
- 48 BGNEXTN Four-Byte Signed Integer
[3003] Applies to Pathtype 4. Contains four bytes which specify in database units the distance a path outline begins before or after the first point of the path. This value can be negative.
- 49 ENDEXTN Four-Byte Signed Integer
[3103] Applies to Pathtype 4. Contains four bytes which specify in database units the distance a path outline begins before or after the last point of the path. This value can be negative.
- 50 TAPENUM Two-Byte Signed Integer
[3202] Contains the number of the current reel of tape for a multi-reel Stream file. For the first tape, the tape number is 1; for the second tape, the tape number is 2. For each additional tape, increment the tape number by one.
- 51 TAPECODE Two-Byte Signed Integer
[3302] Contains a unique, 6-integer code common to all reels of a multi-reel Stream file. It verifies that the correct reels are being read.
- 52 STRCLASS Two-Byte Bit Array (Cadence internal use only)
[3401] If Stream tapes are produced by non-Cadence programs, this record should either be omitted or cleared to zero.

53 RESERVED Not currently used

[3503] This record type was used for NUMTYPES but was not required.

54 FORMAT Two-Byte Signed Integer

[3602] This optional record defines the format of a Stream tape in two bytes. The possible values are:

- 0 for GDSII Archive format
- 1 for GDSII Filtered format
- 2 for EDSIII Archive format
- 3 for EDSIII Filtered format

An Archive Stream file contains elements for all the layers and data types. In an Archive Stream file, the FORMAT record is followed immediately by the UNITS record. A file that does not have the FORMAT record is assumed to be an Archive file.

A Filtered Stream file contains only the elements on the layers and with the datatypes you specify during creation of the Stream file. The list of layers and datatypes specified appear in MASK records. At least one MASK record must immediately follow the FORMAT record. The MASK records are terminated with the ENDMASKS record.

55 MASK ASCII String

[3706] This record is required for and present only in Filtered Stream files. It contains the list of layers and data types specified when the file was created. At least one MASK record must immediately follow the FORMAT record. More than one MASK record can occur. The last MASK record is followed by the ENDMASK record.

In the MASK list, datatypes are separated from the layers with a semicolon. Individual layers or datatypes are separated with a

space. A range of layers or datatypes is specified with a dash. An example MASK list looks like this:

```
1 5-7 10 ; 0-255
```

- | | | |
|----|------------|---|
| 56 | ENDMASKS | No Data Present |
| | | [3800] This record is required for and present only in Filtered Stream files. It marks the end of the MASK records. The ENDMASKS record must follow the last MASK record. ENDMASKS is immediately followed by the UNITS record. |
| 57 | LIBDIRSIZE | Two-Byte Signed Integer |
| | | [3902] Contains the number of pages in the library directory. |
| 58 | SRFNAME | ASCII String |
| | | [3A06] Contains the name of the spacing rules file, if one is bound to the library. |
| 59 | LIBSECUR | Two-Byte Signed Integer |
| | | [3B02] Contains an array of Access Control List (ACL) data. Each ACL entry consists of a group number, a user number, and access rights. Up to 32 ACL entries can be present. |
| 60 | BORDER | No Data Present |
| | | [3C00] Marks the beginning of a border element. |
| 61 | SOFTFENCE | No Data Present |
| | | [3D00] Marks the beginning of a soft fence element. |
| 62 | HARDFENCE | No Data Present |
| | | [3E00] Marks the beginning of a hard fence element. |

63	SOFTWIRE	No Data Present [3F00] Marks the beginning of a soft wire element.
64	HARDWIRE	No Data Present [4000] Marks the beginning of a hard wire element.
65	PATHPORT	No Data Present [4100] Marks the beginning of a path port element.
66	NODEPORT	No Data Present [4200] Marks the beginning of a node port element.
67	USERCONSTRAINT	No Data Present [4300] Marks the beginning of a user constraint.
68	SPACER ERROR	No Data Present [4400] Marks the beginning of a spacer error.
69	CONTACT	No Data Present [4500] Marks the beginning of a contact element.

Stream Syntax

This section contains a Bachus Naur representation of the Stream syntax. Bachus Naur uses ALL CAPS to represent the name of an actual record type and lower case for names that can be further broken down into a set of actual record types. The following table provides descriptions of the Bachus Naur symbols.

Symbol	Symbol	Meaning
Double colon	::	“Is composed of”
Square brackets	[]	An element that can be absent or occur one time.
Braces	{ }	One of the elements within the braces can occur.
Braces with an asterisk	{ }*	The elements within the braces can be absent or occur one or more times.
Braces with a plus	{ }+	The elements within braces must occur one or more times.
Angle brackets	< >	These elements are further defined in the Stream syntax list.
Vertical bar		“Or”

```
<stream format> ::= HEADER BGNLIB [LIBDIRSIZE] [SRFNAME] [LIBSECUR]
LIBNAME [REFLIBS] [FONTS] [ATTRTABLE] [GENERATIONS]
[<FormatType>] UNITS {<structure>}* ENDLIB
```

```

<FormatType> ::= FORMAT | FORMAT {MASK}+ ENDMASKS
<structure> ::= BGNSTR STRNAME [STRCLASS] {<element>}* ENDSTR
  <element> ::= { <boundary> | <path> | <SREF> | <AREF>
    | <text> | <node> | <box>} {<property>}*
    ENDEL
<boundary> ::= BOUNDARY [ELFLAGS] [PLEX] LAYER DATATYPE XY
  <path> ::= PATH [ELFLAGS] [PLEX] LAYER DATATYPE [PATHTYPE]
    [WIDTH] [BGNEXTN] [ENDEXTN] XY
  <SREF> ::= SREF [ELFLAGS] [PLEX] SNAME [<strans>] XY
  <AREF> ::= AREF [ELFLAGS] [PLEX] SNAME [<strans>] COLROW XY
  <text> ::= TEXT [ELFLAGS] [PLEX] LAYER <textbody>
  <node> ::= NODE [ELFLAGS] [PLEX] LAYER NODETYPE XY
  <box> ::= BOX [ELFLAGS] [PLEX] LAYER BOXTYPE XY
<textbody> ::= TEXTYPE [PRESENTATION] [PATHTYPE] [WIDTH] [<strans>]XY
  STRING
  <strans> ::= STRANS [MAG] [ANGLE]
<property> ::= PROPERTY PROPVALUE

```

Example of a Stream Format File

The following is an example of a Stream format file. An explanation follows the example.

```
% od -h example.sf
000 0006 0002 0258 001C 0102 0058 0009 0003.....X....
008 0000 0000 0000 0058 0009 0003 000A 0010 .....X.....
010 0000 0006 3902 0028 000A 3B02 0003 0005 ....9..(..i.....
018 0007 0010 0206 6578 616D 706C 652E 6368 .....example.ch
020 7000 005C 1F06 7265 6631 2E63 6870 0000 p.....ref1.chp..
028 0000 0000 0000 0000 0000 0000 0000 0000 .....
030 0000 0000 0000 0000 0000 0000 0000 0000 .....
****
048 0000 0000 0000 0000 0000 0000 0000 00B4 .....
050 2006 6361 6C6D 6166 6F6E 742E 666E 7400 .calmafont.fnt.
058 0000 0000 0000 0000 0000 0000 0000 0000 .....
060 0000 0000 0000 0000 0000 0000 0000 7465 .....te
068 7874 2E66 6E74 0000 0000 0000 0000 0000 xt.fnt.....
070 0000 0000 0000 0000 0000 0000 0000 0000 .....
078 0000 0000 0000 0000 0000 666F 6E74 2E66 .....font.f
080 6E74 0000 0000 0000 0000 0000 0000 0000 nt.....
088 0000 0000 0000 0000 0000 0000 0000 0000 .....
090 0000 0000 0000 7067 666F 6E74 2E66 6E74 .....pgfont.fnt
098 0000 0000 0000 0000 0000 0000 0000 0000 .....
0A0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0A8 0000 000C 2306 6174 7472 732E 6174 0006 ....#.attrs.at..
0B0 2202 0003 0014 0305 3E41 8937 4BC6 A7EF ".....>A.7K...
0B8 3944 B82F A09B 5A51 001C 0502 0058 0007 9D./..ZQ.....X..
0C0 000C 0011 001D 000A 0058 0007 0011 0011 .....X.....
0C8 003A 0014 000C 0606 6578 616D 706C 6532 .|.....example2
0D0 0004 0B00 000C 1206 6578 616D 706C 6531 .....example1
```

```

0D8  0006  1A01  8000  000C  1C05  425A  0000  0000  .....BZ....
0E0  0000  0008  1302  0002  0002  001C  1003  0000  .....
0E8  4E20  0000  4E20  0000  4E20  0001  4FF0  0001  N ..N ..N ..0...
0F0  3880  0000  4E20  0004  1100  0004  0700  001C  8...N .....
0F8  0502  0058  0007  000C  000B  001C  0009  0058  ...X.....X
100  0008  001C  000F  0039  003A  000C  0606  6578  .....9.|....ex
108  616D  706C  6531  0004  0C00  0006  0D02  0000  ample1.....
110  0006  1602  0000  0006  1701  0005  0006  1A01  .....
118  8006  000C  1B05  4120  0000  0000  0000  000C  .....A .....
120  1003  0000  4E20  0000  4E20  000E  1906  4920  ...N ..N ....I.
128  414D  2048  4552  450D  0004  1100  0004  0800  AM HERE.....
130  0006  2601  0001  0006  0D02  0002  0006  0E02  ..&.....
138  0003  0024  1003  0000  1388  0000  6D60  0000  ...$.m'..
140  2EE0  0000  6D60  0000  1F40  0000  84D0  0000  ...m'..._.....
148  1388  0000  6D60  0004  1100  0004  0900  0006  ...m'.....
150  0D02  0004  0006  0E02  003F  0006  2102  0001  .....?..!...
158  0008  0F03  0000  03E8  0024  1003  0000  3A98  .....$.|..
160  0000  36B0  0000  6590  0000  36B0  0000  84D0  ..6...e...6.....
168  0000  2328  0000  55F0  0000  1770  0006  2B02  ..#(..U...p...+.
170  0002  000A  2C06  4D45  5441  4C00  0006  2B02  .....,METAL...+.
178  000A  000C  2C06  5052  4F50  4552  5459  0004  .....,PROPERTY..
180  1100  0004  0700  0004  0400  .....

```

The database that produced this Stream format output has only two structures. They are called *example1* and *example2*. *Example1* contains a boundary that is template data, a path with two properties, and a middle-center justified text element containing the string, I AM HERE. *Example2* contains only one element, a 2 by 2 AREF of *example1*.

The following are explanations of the records contained in the example Stream file. As a reminder, the first two words (four bytes) of a record are the record header. The first word shows the record

length in bytes, and the second word identifies the record type and the data type.

```
0006 0002 0258
```

The first word reports that this record is 6 bytes long. The second word indicates that this is the HEADER (00 hex) record and that the data type is a two-byte signed integer (02 hex). The information in the third word is the Construct version number, which is version 600 (258 hex).

```
001C 0102 0058 0009 0003 0000 0000 0000 0058 0009
0003 000A 0010 0000
```

This record is 28 (1C hex) bytes. It is the BGNLIB (01 hex) record. The data type is a two-byte signed integer (02). The remaining 24 bytes of information contain the date and time the library was last modified and the date and time of last access.

For example, the last six words of information contain:

Type of Value	Value	Hexadecimal Representation
year	88	0058
month	September	0009
day	3	0003
hour	10 a.m.	000A
minute	16	0010
seconds	0	0000

This record indicates that this library was last accessed on September 3, 1988, at 10:16:00 a.m.

```
0006 3902 0028
```

This record is 6 bytes. It is the LIBDIRSIZE (39 hex) record. The data type is a two-byte signed integer (02). In this example, the directory size is 40 (28 hex) pages.

```
000A 3B02 0003 0005 0007
```

This record is 10 (A hex) bytes. It is a LIBSECUR (3B hex) record. The data type is a two-byte signed integer (02). This example has only one ACL entry. The entry has a group number of 3, a user number of 5, and access rights of 7. This means that the only one with any access rights to this library is user number 5 in group number 3. The access code (007) means this user has read and write access and is also the owner of the library.

```
0010 0206 6578 616D 706C 652E 6368 7000
```

This record is 16 (10 hex) bytes. It is the LIBNAME (02 hex) record. The data type is an ASCII string (06). The six words of information contain the library name, *example.chp*.

```
005C 1F06 7265 6631 2E63 6870 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000
```

This record is 92 (5C hex) bytes. It is the REFLIB (1F hex) record. The data type is an ASCII string (06). In this example, the library *ref1.chp* is the bound reference library. The library is padded with nulls to equal 44 bytes. At least 92 bytes of this record must be present if any reference libraries are bound to the working library. No other reference library is bound, so the last 44 bytes are filled with nulls. If more than two reference libraries are bound (Construct allows 15 reference libraries), the record is extended by 44 bytes for each additional library.

```

00B4 2006 6361 6C6D 6166 6F6E 742E 666E 7400 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 7465 7874 2E66 6E74 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 666F 6E74 2E66 6E74
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 7067 666F
6E74 2E66 6E74 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

```

This record is 180 (B4 hex) bytes. It is the FONTS (20 hex) record. The data type is an ASCII string (06). All 180 bytes of this record must be present if any textfont files are bound to this library. In this example, four textfont files (the maximum possible) are bound to the library. The files are *calmafont.fnt*, *text.fnt*, *font.fnt*, and *pgfont.fnt*. Each string is padded with nulls out to 44 bytes.

```
000C 2306 6174 7472 732E 6174
```

This record is 12 (C Hex) bytes. It is the ATTRTABLE (23 hex) record. The data type is an ASCII string (06). This record is only present if an attribute table is bound to the library. The name of the attribute table is *attrs.at*.

```
0006 2202 0003
```

This record is 6 bytes. It is the GENERATIONS (22 hex) record. The data type is a two-byte signed integer (02). In this example, three generations of a structure are retained in the library.

```
0014 0305 3E41 8937 4BC6 A7EF 3944 B82F A09B 5A51
```

This record is 20 (14 hex) bytes. It is the UNITS (03 hex) record. The data type is an eight-byte real (05). In this example, 3E41 8937 4BC6 A7EF is 1E-3. This implies that a database unit is .001 of a user unit. The record 3944 B82F A09B 5A51 is 1E-9. This implies that a database unit is 1E-9 meters (1E-3 microns).

```
001C 0502 0058 0007 000C 0011 001D 000A 0058 0007  
0011 0011 003A 0014
```

This record is 28 (1C hex) bytes. It is the BGNSTR (05 hex) record. The data type is a two-byte signed integer (02). The information in this record is the creation time and last modification time of the structure. The information is in the same format as the BGNLIB record. This structure was created July 12, 1988, at 5:29:10 p.m. and last modified July 17, 1988, at 5:58:20 p.m.

```
000C 0606 6578 616D 706C 6532
```

This record is 12 (C hex) bytes. It is the STRNAME (06 hex) record. The data type is an ASCII string (06). The structure name is *example2*.

```
0004 0B00
```

This record is 4 bytes. It is the AREF (0B hex) record. It contains no data (00). It marks the start of an AREF.

```
000C 1206 6578 616D 706C 6531
```

This record is 12 (C hex) bytes. It is the SNAME (12 hex) record. The data type is an ASCII string (06). This record contains the name of referenced structure *example1*.

```
0006 1A01 8000
```

This record is 6 bytes. It is the STRANS (1A hex) record. The data type is a bit array (01). In this example, only bit 0 is set, which implies that this AREF is reflected. Since bit 13 and 14 are not set, this structure's magnification and angle, respectively, are not absolute.

```
000C 1C05 425A 0000 0000 0000
```

This record is 12 (C hex) bytes. It is the ANGLE (1C hex) record. The data type is eight-byte real data (05). The data 425A 0000 0000

0000 represents 90.0, which implies that this AREF is placed at an angle of 90 degrees.

```
0008 1302 0002 0002
```

This record is 8 bytes. It is the COLROW (13 hex) record. The data type is a two-byte signed integer (02). This example contains a 2 x 2 AREF.

```
001C 1003 0000 4E20 0000 4E20 0000 4E20 0001 4FF0
0001 3880 0000 4E20
```

This record is 28 (1C hex) bytes. It is the XY (10 hex) record. The data type is a four-byte signed integer (03). The data, taken two words at a time, can be translated to decimal as: 20000, 20000, 20000, 86000, 80000, 20000. Multiply these by .001 (because a data base unit is .001 of a user unit). The results are the coordinates: (20, 20), (20, 86), and (80, 20). The first coordinate is the array reference point. The second coordinate is a point which is displaced from the array reference point in the Y-direction by the number of columns times the inter-column spacing. In this example, the second point was displaced 66 (86 - 20) units from the array reference point. Since the array has two columns, this implies that the inter-column spacing is 33 units. A similar calculation can be carried out to verify that the inter-row spacing is 30 units.

```
0004 1100
```

This record is 4 bytes. It is the ENDEL (11 hex) record. It contains no data (00). ENDEL marks the end of an element.

```
0004 0700
```

This record is 4 bytes. It is the ENDSTR (07 hex) record. It contains no data (00). ENDSTR marks the end of a structure.

```
001C 0502 0058 0007 000C 000B 001C 0009 0058 0008
001C 000F 0039 003A
```

This is another BGNSTR record. This structure was created July 12, 1988, at 11:28:09 a.m., and last modified August 28, 1988, at 3:57:58 p.m.

```
000C 0606 6578 616D 706C 6531
```

This is another STRNAME record. It contains the string *example1*.

```
0004 0C00
```

This record is 4 bytes. It is the TEXT (0C hex) record. It contains no data (00). Text marks the start of a text element.

```
0006 0D02 0000
```

This record is 6 bytes. It is the LAYER (0D hex) record. The data type is a two-byte signed integer (02). This text element is on layer 0.

```
0006 1602 0000
```

This record is 6 bytes. It is the TEXTTYPE (16 hex) record. The data type is a two-byte signed integer (02). This text element is texttype 0.

```
0006 1701 0005
```

This record is 6 bytes. It is the PRESENTATION (17 hex) record. The data type is a bit array (01). The hex number 0005 in binary has all bits set to zero except bits 13 and 15. Since bits 10 and 11 are 00, the text element is font 0. Since bits 12 and 13 are 01, the text has a middle vertical position. Since bits 14 and 15 are 01, the text has a center horizontal presentation.

```
0006 1A01 8006
```

This is another STRANS record. This text is reflected and has an absolute magnification and absolute angle.

```
000C 1B05 4120 0000 0000 0000
```

This record is 12 (C hex) bytes. It is the MAG (1B hex) record. The data type is eight-byte real (05). The data in this record represents 2.0, meaning that this text is magnified two times.

```
000C 1003 0000 4E20 0000 4E20
```

This is another XY record. The text is placed at coordinate (20, 20).

```
000E 1906 4920 414D 2048 4552 450D
```

This record is 14 (E hex) bytes. It is the STRING (19 hex) record. The data type is an ASCII string (06). The text string is I AM HERE.

```
0004 1100
```

This is another ENDEL record.

```
0004 0800
```

This record is 4 bytes. It is the BOUNDARY (08 hex) record. It contains no data (00). BOUNDARY marks the start of a boundary element.

```
0006 2601 0001
```

This record is 6 bytes. It is the ELFLAGS (26 hex) record. The data type is a bit array (01). Since bit 15 is set, this element is template data. Since bit 14 is not set, the element is not external data.

```
0006 0D02 0002
```

This is another LAYER record. The boundary is on layer 2.

```
0006 0E02 0003
```

This record is 6 bytes. It is the DATATYPE (0E hex) record. The data type is a two-byte signed integer (02). This boundary is of datatype 3.

```
0024 1003 0000 1388 0000 6D60 0000 2EE0 0000 6D60
0000 1F40 0000 84D0 0000 1388 0000 6D60
```

This is another XY record. The coordinates are (5, 28), (12, 28), (8, 34), and (5, 28).

```
0004 1100
```

This is another ENDEL record.

```
0004 0900
```

This record is 4 bytes. It is the PATH (09 hex) record. It contains no data (00). PATH marks the start of a path element.

```
0006 0D02 0004
```

This is another LAYER record. The path is on layer 4.

```
0006 0E02 003F
```

This is another DATATYPE record. The path is datatype 63 (3F hex).

```
0006 2102 0001
```

This record is 6 bytes. It is the PATHTYPE (21 hex) record. The data type is a two-byte signed integer (02). This path is pathtype 1.

```
0008 0F03 0000 03E8
```

This record is 8 bytes. It is the WIDTH (0F hex) record. The data type is a four-byte signed integer (03). The number 03E8 hex is 1000 in decimal. Multiply this by .001 (because a database unit is .001 of a user unit). The result is 1; therefore, the width of this path is 1.

```
0024 1003 0000 3A98 0000 36B0 0000 6590 0000 36B0
0000 84D0 0000 2328 0000 55F0 0000 1770
```

This is another XY record. This path's coordinates are (15, 14), (26, 14), (34, 9), and (22, 6).

0006 2B02 0002

This record is 6 bytes. It is the PROPATTR (2B hex) record. The data type is a two-byte signed integer (02). This path has a property with attribute number 2.

000A 2C06 4D45 5441 4C00

This record is 10 (A hex) bytes. It is the PROPVALUE (2C hex) record. The data type is an ASCII string (06). The property value for property attribute 2 (above) is METAL. The odd length string (five characters) is padded with a null.

0006 2B02 000A

This is another PROPATTR record. This path has another property associated with it. The property has attribute number 10 (A hex).

000C 2C06 5052 4F50 4552 5459

This is another PROPVALUE record. Property attribute 10 (above) has the value PROPERTY.

0004 1100

This is another ENDEL record.

0004 0700

This is another ENDSTR record.

0004 0400

This record is 4 bytes. This record is the ENDLIB (04 hex) record. It contains no data (00). ENDLIB marks the end of a Stream format file.