Common Language Infrastructure (CLI)
Partition V:
Debug Interchange Format

Table of contents

1 Portable CILDB files

Portable CILDB files provide a standard way to interchange debugging information between CLI producers and consumers. This partition serves to fill in gaps not covered by metadata, notably the names of local variables and source line correspondences.

1.1 Encoding of integers

All integers are stored in little-endian format, except for those in signatures, which are encoded as described in Partition II.

1.2 CILDB header

A CILDB file starts with a 72-byte header, whose layout is as follows:

Offset	Size	Field	Description
0	16	**Signature**	Magic signature for CILDB "_ildb_signature\0"
16	16	**GUID**	Version GUID
32	4	**UserEntryPoint**	MethodDef token of the entry point.
36	4	**CountOfMethods**	Number of rows in the SymMethod table.
40	4	**CountOfScopes**	Number of rows in the SymScopes table.
44	4	**CountOfVars**	Number of rows in the SymVariable table.
48	4	**CountOfUsing**	Number of rows in the SymUsing table.
52	4	**CountOfConstants**	Number of rows in the SymConstant table.
56	4	**CountOfDocuments**	Number of rows in the SymDocument table.
60	4	**CountOfSequencePoints**	Number of rows in the SymSequencePoint table.
64	4	**CountOfMiscBytes**	Number of bytes in the SymMisc heap.
68	4	**CountOfStringBytes**	Number of bytes in the SymString heap.

1.2.1 Version GUID

The version GUID is the 16-byte sequence shown below:

0x7F	0x55	0xE7	0xF1	0x3C	0x42	0x17	0x41
0x8D	0xA9	0xC7	0xA3	0xCD	0x98	0x8D	0xF1

1.3 Tables and heaps

The CILDB header is immediately followed by various tables and heaps, in the following order:

1. SymConstant
2. SymMethod
3. SymScopes
4. SymVariable
5. SymUsing
6. SymSequencePoint
7. SymDocument

8. SymMisc
9. SymString

Some of the tables contain CIL offsets. These offsets are in bytes, and the offset of the first instruction is zero. The offsets do not necessarily match the beginning of a CIL instruction. For example, offsets denoting the end of a range of bytes often refer to the last byte of an instruction. Lengths are also in bytes.

The rows in each of the tables 3–7 above that belong to the same method must be contiguous within their parent table.

1.3.1 SymConstant table

Each row of the SymConstant table describes a constant, as follows:

Offset	Size	Field	Description
0	4	**Scope**	Index of parent scope
4	4	**Name**	Index of the name in the SymString heap
8	4	**Signature**	Index of the signature in the SymMisc heap
12	4	**SignatureSize**	Length of the signature
16	4	**Value**	Index of the value in the SymMisc heap
20	4	**ValueSize**	Length of the value.

The value of the constant is encoded just like a Blob for the *Value* column of a Constant metadata table in Partition II, except that there is no length prefix.

1.3.2 SymDocument table

Each row of a SymDocument describes a source document, as shown below. The document can either be referred to indirectly (by its URL) or incorporated directly into the CILDB file as part of the SymMisc heap. The GUID values referred to in this subclause are not defined by this Standard; space is simply reserved for them.

Offset	Size	Field	Description
0	16	**Language**	GUID for the language.
16	16	**LanguageVendor**	GUID for the language vendor.
32	16	**DocumentType**	GUID for the document type.
48	16	**AlgorithmId**	GUID of the checksum algorithm; or 0 if there is no checksum.
64	4	**CheckSumSize**	Size of the checksum; or 0 if there is no checksum.
68	4	**CheckSumEntry**	Index of the checksum in the SymMisc heap; or 0 if there is no checksum.
72	4	**SourceSize**	Size of the source in the SymMisc heap; or 0 if the source document is not directly incorporated into the file.
76	4	**SourceEntry**	Index of the source in the SymMisc heap; or 0 if the source document is not directly incorporated into the file.
80	4	**UrlEntry**	Index of the document URL in the SymString heap.

1.3.3 SymMethod table

Each row of a SymMethod table has the following format:

Offset	Size	Field	Description

0	4	**MethodToken**	A MethodDef metadata token.
4	8	**Scopes**	[Start,Stop) range of SymScope table.
12	8	**Vars**	[Start,Stop) range of SymVariable table.
20	8	**Using**	[Start,Stop) range of SymUsing table.
28	8	**Constant**	[Start,Stop) range of SymConstant table.
36	8	**Documents**	[Start,Stop) range of SymDocument table.
44	8	**SequencePoints**	[Start,Stop) range of SymSequencePoint table.

Each [Start,Stop) range is represented as two 4-byte integers. The first integer is the index of the first related table row; the second integer is the index of one past the last related table row.

The rows of a SymMethod table are sorted in ascending order of the **MethodToken** field. There is at most one row for each method.

1.3.4 SymSequencePoint table

Each row of a SymSequencePoint table describes a sequence point, as follows:

Offset	Size	Field	Description
0	4	**Offset**	CIL offset of the sequence point.
4	4	**StartLine**	Starting line of the source document.
8	4	**StartColumn**	Starting column, or 0 if not specified.
12	4	**EndLine**	Ending line of the source document, or 0 if not specified.
16	4	**EndColumn**	Ending column, or 0 if not specified.
20	4	**Doc**	Index of the source document in the SymString heap.

Together, EndLine and EndColumn specify the column "one past" the last byte position associated with the sequence point. In other words, they specify the end of a half-open interval [start,end).

Rows of the SymSequencePoint belonging to the same Method must be contiguous and sorted in ascending order of Offset.

1.3.5 SymScope table

Each row of a SymScope table describes a scope, as follows:

Offset	Size	Field	Description
0	4	**Parent**	Index of parent SymScope row, or –1 if scope has no parent.
4	4	**StartOffset**	CIL offset of the first byte in the scope.
8	4	**EndOffset**	CIL offset of the last byte in the scope.
12	4	**HasChildren**	1 if scope has child scopes; 0 otherwise
16	4	**HasVars**	1 if scope has variables; 0 otherwise

The scopes belonging to a method must form a tree, with the following constraints:

- A parent scope must precede its child scopes.
- The StartOffset and EndOffset of a child scope must be within the (inclusive) range of offsets specified by its parent's scope.

1.3.6 SymVariable table

Each row of a SymVariable table describes a local variable.

Offset	Size	Field	Description
0	4	**Scope**	Index of the parent scope
4	4	**Name**	Index of the variable's name in the SymString heap.
8	4	**Attributes**	Flags describing the variable (see below).
12	4	**Signature**	Index of the signature in the SymMisc heap.
16	4	**SignatureSize**	Length of the signature.
20	4	**AddressKind**	Always 1.
24	4	**Address1**	Local variable number.
28	4	**Address2**	Always 0.
32	4	**Address3**	Always 0.
36	4	**StartOffset**	CIL offset where the variable is first visible .
40	4	**EndOffset**	CIL offset where the variable is last visible.
44	4	**Sequence**	Always 0.
48	4	**IsParam**	Always 0.
52	4	**IsHidden**	1 if variable should be hidden from debugger; 0 otherwise.

The least-significant bit of Attributes indicates whether the variable is user-generated (0) or compiler-generated (1). The other bits are reserved and should be set to zero.

Because parameters are fully described by the Metadata, they do not appear in the SymVariable table.

1.3.7 SymUsing table

Each row of the SymUsing table describes importation of a namespace, as follows:

Offset	Size	Field	Description
0	4	**Scope**	Index of the parent scope
4	4	**Namespace**	Index of the namespace in the SymString heap

1.3.8 SymMisc heap

The SymMisc heap holds various byte sequences (e.g., signatures and checksums).

1.3.9 SymString heap

The stream of bytes in the SymString heap has the same form as those for the #Strings heap (see Partition II).

1.4 Signatures

Signatures of variables and constants are encoded as an index into the SymMisc heap, and a signature size. The values of the bytes are similar to those for a FieldSig (see Partition II), and include the prefix FIELD (0x6), even though the variables are not fields. Because the length of the signature is encoded in the tables, it is not included in the SymMisc heap. For example, type int32 is encoded as "0x06 0x08".