1. Acknowledgements

To the author and first editor of this document, Todd Newman of Digital Equipment Corp., go our sincere thanks. Countless iterations of protocol changes and nit-fixing went into early versions of it and we gratefully acknowledge his care and dedication to this thankless task. Randi Rost and Jeff Friedberg of Digital Equipment Corp. helped in the production of the first draft of this document by explaining the original intent of the architecture team. Sally C. Barry of Digital Equipment Corp., was a patient and thorough reviewer of early versions of this document throughout its many revisions.

Randi Rost of Digital Equipment Corp. took over the job of editing this document in January of 1989, and performed this task until April of 1990. During this time, Randi was responsible for five separate revisions of the document.

We owe a major debt to the PEX architecture team which met frequently between June, 1987 and August, 1991: Jeffrey Friedberg, John McConnell, Pete Nishimoto, Randi J. Rost, and Jeffrey S. Saltz of Digital Equipment Corporation; Dave Gorgen, Tom Gross, and Jeff Stevenson of Hewlett-Packard Company; David Plunkett of Solbourne Computer; Jan Hardenbergh of Stardent Computer, Inc.; and Marty Hess, and Jim Van Loo of Sun Microsystems.

2. PEX Protocol Specification

2.1. Request Processing

Workstation regeneration and requests which process output commands are not guaranteed to be atomic with respect to concurrent rendering to the same destination drawable. However atomicity of execution is guaranteed for each individual output command primitive, excluding Execute Structure, GSE, GDP 3D, and GDP 2D. Also, requests and output commands will be executed in sequential order for a given connection.

2.2. Protocol Format

2.2.1. Request Format

Requests are listed in section 2.8 and in section 4 through the end of the document. Output Commands are described in section 3.

2.2.2. Reply Format

Replies are listed after the request.

2.2.3. Error Format

See section 2.5.7 for the encoding of particular errors.

2.2.4. Event Format

See section 2.6 for the encoding of events.

2.3. Syntax

As far as possible, this document uses the layout and syntactic conventions used in the X encoding document. In particular, note that all numbers are decimal, unless prefixed by #x, in which case they are hexadecimal. Also note that, for entities of a variable size, E, the notation pad(E) indicates the number of bytes required to pad the entity to a multiple of 4 bytes.

There are some unavoidable differences from the X encoding document conventions, however, because of the many variable sized entities and entities with optional data in PEX.

2.4. Naming Conventions

The names of the fields in requests, replies and lists of values match the names used in the *PEX Protocol Specification*.

X Window System is a trademark of the Massachusetts Institute of Technology

X3D-PEX and PEX are trademarks of the Massachusetts Institute of Technology

2.5. Common Types

The types listed in this section are some of the types mentioned in the PEX protocol. Generally, derivative types (e.g. ENUM_TYPE_INDEX) are replaced by the primitive or primitives of which they are composed (e.g. INT16). Only when this is impractical is the derivative type used; generally this is when the type is of variable length or a list of compound types is used.

2.5.1. LISTofFOO

In this document, a LISTofFOO is a counted list of elements of type FOO, the count field preceding it in the request or reply.

In cases where the number of items in the list is easily computed from the length of the list, the number of items may not be supplied. (See, for example, the **Polyline3D** output command.) The goal is to reduce the burden on the client. Such cases are indicated in the encoding document.

There is also no count provided for a list that has an associated item_mask.

Unless otherwise marked, a LISTofLISTofFOO is implemented as follows: somewhere in the request or reply there is a number of lists field (perhaps several if there are multiple lists of lists). Each LIST is preceded by a CARD32 count field.

So the overall format is:

4	CARD32	number of lists
. . .		(potentially other things here)
4	CARD32	number of items in first list
n	FOO	first item of first list
n	FOO	second item of first list
. . .		
n	FOO	last item of first list
4	CARD32	number of items in second list
n	FOO	first item of second list
. . .		

In only one place in the PEX protocol is a LISTofLISTofLISTofFOO required, in the **SetOfFillAreaSets** output command. In this case, the number of lists is a CARD16. Other counts within the list are CARD16.

The format of this is:

2	CARD16	number of lists of lists
. . .		(potentially other things here)
2	CARD16	number of lists in first list of list
2	CARD16	number of items in first list
n	FOO	first item of first list
n	FOO	second item of first list
. . .		
n	FOO	last item of first list
2	CARD16	number of items in second list
n	FOO	first item of second list
. . .		
2	CARD16	number of lists in second list of list
2	CARD16	number of items in first list
n	FOO	first item of first list
n	FOO	second item of first list
. . .		

n	FOO	last item of first list
2	CARD16	number of items in second list
n	FOO	first item of second list
. . .		
2	CARD16	number of lists in last list of list
2	CARD16	number of items in first list
n	FOO	first item of first list
n	FOO	second item of first list
. . .		
n	FOO	last item of first list
2	CARD16	number of items in second list
n	FOO	first item of second list
. . .		

2.5.2. BITMASK and LISTofVALUE

Because the values in a value list are of variable size, a list of values can only be interpreted serially; there is no way to find the Nth element without examining the N-1 previous elements.

Elements have all been padded to a multiple of 4 bytes. If values appear which could be encoded in less space, it is assumed that they will be placed in the least significant bits of the word. (Byte-swapping may occur between the client and the server.)

2.5.3. Floating Point Format - FLOAT

A FLOAT is a floating point quantity in one of the server supported formats. If the client wishes to communicate with the server in other than the server's native floating point format, the server must perform the necessary conversions. The format word of a request tells the server what format the client wishes to use.

Depending on the floating point format, the size of a FLOAT may be 4 or 8 bytes. Floating point values passed to the server may be in any one of the registered floating point formats:

1	IEEE_754_32	4 bytes
2	DEC_F_Floating	4 bytes
3		