HP_DataStream() Function Reference

Revision: p1.1
Revision Date: 26 June, 1997
Author(s):

Word for Windows File: datastream.doc
Word for Windows Version: 6.0
This copy printed: 11/18/98 11:23 AM

Document Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Revision Description</th>
<th>Date</th>
<th>Approval</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1.0</td>
<td>First Preliminary Version</td>
<td>31Mar1997</td>
<td></td>
</tr>
<tr>
<td>p1.1</td>
<td>Corrected minor spelling errors</td>
<td>26June1997</td>
<td></td>
</tr>
</tbody>
</table>

Note: This document is for HP Personnel only. Please do not copy and distribute without notification to the PCL XL Feature Definition Team.

*** NOTICE ***

HEWLETT-PACKARD COMPANY MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE SOFTWARE OR TECHNICAL INFORMATION. HEWLETT-PACKARD COMPANY DOES NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF
THE SOFTWARE OR TECHNICAL INFORMATION IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS, OR OTHERWISE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE OR TECHNICAL INFORMATION IS ASSUMED BY YOU. The exclusion of implied warranties is not permitted by some jurisdictions. The above exclusion may not apply to you.

IN NO EVENT WILL HEWLETT-PACKARD COMPANY BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION AND THE LIKE) ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE OR TECHNICAL INFORMATION EVEN IF HEWLETT-PACKARD HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to you. Hewlett-Packard liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort including negligence, product liability or otherwise), will be limited to US $50.

Copyright © 1996 Hewlett-Packard Company. All rights reserved.
**HP_DataStream()**

The HP_DataStream function added to JETLIB.CPP Version 4.0 writes the passed in data to the output stream in XL binary format. All tags and data types and values are written to the user's stream in the correct syntax and format.

The function allows writing any number of diverse attributes to the data stream. Each attribute is described in a structure that contains both ID and value. Pointers to as many attribute structures as are needed by an operator are passed to the HP_DataStream operator. This permits a single function to be used to stream out all supported XL operators.

JETLIB.CPP can also be compiled under 'C' and examples for using HP_DataStream for 'C' code can be found at the end of this document.

```c
HP_StdFuncPrefix HP_DataStream(
    HP_StreamHandleType pStream,
    HP_UByte Operator,
    ...
    void * NULL,
)
```

**Parameters**

**HP_StreamHandleType pStream**

Pointer to a structure that contains information about the current data output stream. The binary data will be written out to the stream designated in this structure.

**HP_UByte Operator**

This is the XL operator tag to be added to the data stream. The values can be found in the PCL-XL Feature Reference, Appendix B. Defines for each valid operator can be found in the file JETLIB.H.

... (0 to n pointers to ATTRIBUTE structures)

Pointers to ATTRIBUTE structures containing the tag, type and value to be used for each operator. If an operator needs no attributes, a NULL pointer will suffice. As many ATTRIBUTE pointers as needed may be passed in a single call to HP_DataStream(), but the last parameter passed must be a NULL pointer.

**Void * NULL**

A NULL pointer marking the end of the ATTRIBUTE pointer list. If no ATTRIBUTE pointers are passed, the NULL pointer must still be included in the HP_DataStream() function call.
ATTRIBUTE Structure

This is the class definition for the ATTRIBUTE structure for the all inclusive call HP_DataStream(). Attributes created using this structure carry data about themselves to the HP_DataStream() call, are parsed, and turned into the binary data the XL language uses.

```c
struct ATTRIBUTE
{
    HP_UByte Tag,
    HP_UByte Type;
    HP_UInt32 arrayLen;
    HP_UInt16 cksum;
    union DataValue
    {
        HP_UByte ubyte;
        HP_UInt16 uint16;
        HP_SInt16 sint16;
        HP_UInt32 uint32;
        HP_SInt32 sint32;
        HP_Real32 real32;
        HP_pUByte ubyte_array;
        HP_pUInt16 uint16_array;
        HP_pSInt16 sint16_array;
        HP_pUInt32 uint32_array;
        HP_pSInt32 sint32_array;
        HP_pReal32 real32_array;

        struct {
            HP_UByte x, y;
        } UByte_XY;
        struct {
            HP_UInt16 x, y;
        } UInt16_XY;
        struct {
            HP_SInt16 x, y;
        } SInt16_XY;
        struct {
            HP_UInt32 x, y;
        } UInt32_XY;
        struct {
            HP_SInt32 x, y;
        } SInt32_XY;
        struct {
            HP_Real32 x, y;
        } Real32_XY;

        struct {
            HP_UByte x1, y1, x2, y2;
        } UByte_BOX;
        struct {
```
DRAFT HP_DataStream() Function Reference

```c
HP_UInt16 x1, y1, x2, y2;
}

struct {
HP_SInt16 x1, y1, x2, y2;
}

SInt16_BOX;

struct {
HP_UInt32 x1, y1, x2, y2;
}

UInt32_BOX;

struct {
HP_SInt32 x1, y1, x2, y2;
}

SInt32_BOX;

struct {
HP_Real32 x1, y1, x2, y2;
}

Real32_BOX;

val;

ATTRIBUTE( HP_UByte aTag, HP_UByte aType )
{
Tag = aTag;
Type = aType;
arrayLen = 0;
cksum = (HP_UInt16)aTag * (HP_UInt16)aType;
}

HP_UByte Tag
This is the attribute ID for the attribute described by the current instantiation of the structure. The values can be found in the PCL-XL Feature Reference, Appendix E. Defines for each valid operator can be found in the file JETLIB.H.

HP_UByte Type
This is the Data Type Tag for the value of the attribute described by the current instantiation of the structure. The values can be found in the PCL-XL Feature Reference, Appendix D. Defines for each valid data type can be found in the file JETLIB.H.

HP_UInt32 arrayLen
When the attributes data type is a pointer to an array, this value must be set with the number of elements in that array. This is required so that the correct amount of data is copied to the output stream. The default value for this variable is 0 and is ignored if the data type tag does not indicate an array.

HP_UInt16 cksum
An internally generated number based on the Tag and Type variables to insure that the current attribute is valid. It is set by the C++ constructor and should not be touched by any user code.

union DataValue
The value associated with each attribute is described by the Type variable and stored in this part of the structure. All the XL supported data types appear here. Valid data types for each attribute are detailed in the PCL-XL Feature Reference, Appendix F. Defines for each valid data type
```
can be found in the file `JETLIB.H`. Because this item is a union, it may contain one, and only one, of the following values.

**HP_UByte  ubyte**  
A single 8 bit Unsigned byte value.

**HP(UInt16  uint16**  
A single 16 bit unsigned value.

**HP_SInt16  sint16**  
A Single 16 bit signed value.

**HP_UInt32  uint32**  
A single 32 bit unsigned value.

**HP_SInt32  sint32**  
A single 32 bit signed value.

**HP_Real32  real32**  
A single 32 bit real number.

**HP_pUByte  ubyte_array**  
A pointer to an array of 8 bit unsigned values. The exact number of elements should be set in the arrayLen variable.

**HP_pUInt16  uint16_array**  
A pointer to an array of 16 bit unsigned values. The exact number of elements should be set in the arrayLen variable.

**HP_pSInt16  sint16_array**  
A pointer to an array of 16 bit signed values. The exact number of elements should be set in the arrayLen variable.

**HP_pUInt32  uint32_array**  
A pointer to an array of 32 bit unsigned values. The exact number of elements should be set in the arrayLen variable.

**HP_pSInt32  sint32_array**  
A pointer to an array of 32 bit signed values. The exact number of elements should be set in the arrayLen variable.

**HP_pReal32  real32_array**  
A pointer to an array of floating point values. The exact number of elements should be set in the arrayLen variable.

**struct { HP_UByte x, y;}  ubyte_xy**  
A structure containing two 8 bit unsigned values.

**struct {HP_UInt16 x, y;}  uint16_xy**  
A structure containing two 16 bit unsigned values.

**struct {HP_SInt16 x, y;}  sint16_xy**  
A structure containing two 16 bit signed values.
struct {HP_UInt32 x, y; } uint32_xy
A structure containing two 32 bit unsigned values.

struct {HP_SInt32 x, y; } sint32_xy
A structure containing two 32 bit signed values.

struct {HP_Real32 x, y; } real32_xy
A structure containing two floating point numbers.

struct {HP_UByte x1, y1, x2, y2; } ubyte_box
A structure containing 8 bit unsigned numbers.

struct {HP_UInt16 x1, y1, x2, y2; } uint16_box
A structure containing four 16 bit unsigned numbers.

struct {HP_SInt16 x1, y1, x2, y2; } sint16_box
A structure containing four 16 bit signed numbers.

struct {HP_UInt32 x1, y1, x2, y2; } uint32_box
A structure containing four 32 bit unsigned numbers.

struct {HP_SInt32 x1, y1, x2, y2; } sint32_box
A structure containing four 32 bit signed numbers.

struct {HP_Real32 x1, y1, x2, y2; } real32_box
A structure containing four floating point numbers.

Constructor (C++ Only)

ATTRIBUTE( HP_UByte aTag, HP_UByte aType )
This constructor is used to fill the Tag and Type variables with the description of the attribute when the attribute is instantiated by the users code. The arrayLen variable is zeroed, and a checksum is generated and stored to validate the structure when a pointer to it is passed to the HP_DataStream() function.

If JetLIB.CPP is compiled with a standard 'C' compiler, the constructor is not compiled. Instead, the function HP_InitAttribute() is compiled and should be used to initialize each attribute structure after declaration. See the examples for details.

Destructor
None.
Examples in C++

The following example is used to write a BeginSession operator to the data stream.

```cpp
Void DR_StartXLSession(HP_StreamHandleType pStream)
{
    ATTRIBUTE measure(HP_Measure, HP_UByte);
    ATTRIBUTE units(HP_UnitsPerMeasure, HP_UInt16Xy);
    ATTRIBUTE report(HP_ErrorReport, HP_UByte);

    measure.val.uint16_xy.x = DR_GetCurrentHorizRes();
    measure.val.uint16_xy.y = DR_GetCurrentVertRes();
    units.val.ubyte = HP_eInch;
    report.val.ubyte = HP_eBackChannel

    HP_DataStream(pStream, HP_BeginSession,
                  &measure, &units, &report, NULL);
}
```

The following example writes text data to the XL data stream

```cpp
void DR_SendXLString(HP_StreamHandleType pStream,
                     HP_pUByte string)
{
    ATTRIBUTE str(HP_TextData, HP_pUByte);

    str.val.ubyte_array = string;
    str.arrayLen = strlen(string);
    HP_DataStream(pStream, HP_SystemText, &str, NULL);
}
```

The following example writes image data to the XL data stream

```cpp
void DR_SendXLImage
(
    HP_DataHandleType pStream,
    HP_pUByte image, HP_Uint32 imgSize, BOOL compressed,
    HP_UInt16 xSize, HPUInt16 ySize,
    HP_UInt16 startX, HPUInt16 StartY,
    HP_UInt16 xScale, HPUInt16 uScale
)
)
ATTRIBUTE map(HP_ColorMapping, HP_UByte);
ATTRIBUTE depth(HP_ColorDepth, HP_UByte);
ATTRIBUTE xw(HP_SourceWidth, HP_UInt16);
ATTRIBUTE yw(HP_SourceHeight, HP_UInt16);
ATTRIBUTE destSize(HP_DestinationSize, HP_UInt16Xy);
ATTRIBUTE sLine(HP_StartLine, HP_UInt16);
ATTRIBUTE eLine(HP_BlockHeight, HP_UInt16);
```
ATTRIBUTE comp(HP_CompressMode, HP_UByte);
ATTRIBUTE point(HP_Point, HP_UInt16Xy);

map.val.ubyte = HP_eIndexedPixel;
depth.val.ubyte = HP_e8Bit;
xw.val.uint16 = xSize;
yw.val.uint16 = ySize;
destSize.val.uint16_xy.x = xSize * xScale;
destSize.val.uint16_xy.y = ySize * yScale;
HP_DataStream(pStream, HP_BeginImage, &map, &depth, &destSize, &xw, &yw, NULL);

sLine.val.uint16 = 0;
eLine.val.uint16 = ySize;
if (compressed)
    comp.val.ubyte = HP_eNoCompression;
else
    comp.val.ubyte = HP_eRLECompression;

HP_DataStream(pStream, HP_ReadImage, &sLine, &eLine, &comp, NULL);
HP_DataUByteArray(pStream, image, imgSize);
HP_DataStream(pStream, EndImage, NULL);
}

The following example is used to write a BeginSession operator to the data stream.

Void DR_StartXLSession(HP_StreamHandleType pStream)
{
    ATTRIBUTE measure, units, report;

    HP_InitAttribute( &Measure, HP_Measure, HP_UByte);
    HP_InitAttribute( &units, ,HP_UnitsPerMeasure, HP_UInt16Xy);
    HP_InitAttribute( &report, HP_ErrorReport, HP_UByte);

    measure.val.uint16_xy.x = DR_GetCurrentHorizRes();
    measure.val.uint16_xy.y = DR_GetCurrentVertRes();
    units.val.ubyte = HP_eInch;
    report.val.ubyte = HP_eBackChannel

    HP_DataStream(pStream, HP_BeginSession, &measure, &units, &report, NULL);
}
Examples in 'C'

The following example writes text data to the XL data stream

```c
void DR_SendXLString(HP_StreamHandleType pStream,
                     HP_pUByte string)
{
    ATTRIBUTE str;
    HP_InitAttribute( &str, HP_TextData, HP_pUByte);
    str.val.ubyte_array = string;
    str.arrayLen = strlen(string);
    HP_DataStream(pStream, HP_SystemText, &str, NULL);
}
```

The following example writes image data to the XL data stream

```c
void DR_SendXLImage
    (HP_DataHandleType pStream,
     HP_pUByte image, HP_UInt32 imgSize, BOOL compressed,
     HP_UInt16 xSize, HPUInt16 ySize,
     HP_UInt16 startX, HPUInt16 StartY,
     HP_UInt16 xScale, HPUInt16 uScale)
{
    ATTRIBUTE map, depth, xw, yw, destSize;
    ATTRIBUTE sLine, eLine, comp, point;
    HP_InitAttribute( &map, HP_ColorMapping, HP_UByte);
    HP_InitAttribute( &depth, HP_ColorDepth, HP_UByte);
    HP_InitAttribute( &xw, HP_SourceWidth, HP_UInt16);
    HP_InitAttribute( &yw, HP_SourceHeight, HP_UInt16);
    HP_InitAttribute( &destSize, HP_DestinationSize, HP_UInt16Xy);
    HP_InitAttribute( &sLine, HP_StartLine, HP_UInt16);
    HP_InitAttribute( &eLine, HP_BlockHeight, HP_UInt16);
    HP_InitAttribute( &comp, HP_CompressMode, HP_UByte);
    HP_InitAttribute( &point, HP_Point, HP_UInt16Xy);
    map.val.ubyte = HP_eIndexedPixel;
    depth.val.ubyte = HP_e8Bit;
    xw.val.uint16 = xSize;
    yw.val.uint16 = ySize;
    destSize.val.uint16_xy.x = xSize * xScale;
    destSize.val.uint16_xy.y = ySize * yScale;
    HP_DataStream(pStream, HP_BeginImage,
                  &map, &depth, &destSize,
                  &xw, &yw, NULL);
    sLine.val.uint16 = 0;
}```
eLine.val.uint16 = ySize;
if (compressed)
    comp.val.ubyte = HP_eNoCompression;
else
    comp.val.ubyte = HP_eRLECompression;

HP_DataStream(pStream, HP_ReadImage,
              &sLine, &eLine, &comp, NULL);

HP_DataUByteArray(pStream, image, imgSize);

HP_DataStream(pStream, EndImage, NULL);

}