Release Notes for VSI GKS 7.3

on OpenVMS Integrity servers

September 2017

© Copyright 2017 VMS Software, Inc., Bolton Massachusetts, USA

CONTENTS

Preface v

1 Product Summary 1
1.1 Release Description 1
1.2 Operating System Support 1
1.3 Layered Product Rebranding 2
1.4 Motif Version 1.7C Support 2

2 Changes for Alpha AXP and Integrity server Systems 3
2.1 Changes to Escapes 3
2.1.1 Affected Escapes 3
2.1.2 How to Use the New Format 13
2.1.3 Documentation Changes 13

3 Installation Notes 14
3.1 HP GKS or HP GKSRT must be removed before installing VSI GKS Version 7.3 . . 14

4 Previous Enhancements 15
4.1 Metafile Header Format Change 15
4.2 NIST Certification 15
4.3 New Escape: Set Fill Simplification Method 16
4.4 New Escape: Hit Test 17
4.5 X11R5 Scalable Font Support 18
4.6 Inquire DBuffer Pixmap & Inquire Background Pixmap supported 18
4.7 New Environmental Controls 18
4.7.1 GKS$FILL_SIMPLIFY_METHOD / GKS$COMPLEX_FILL_METHOD 18
4.7.2 The GKS$HATCH_SIMULATION logical 19
4.7.3 The GKS$VISUAL_CLASS logical 19
4.7.4 The GKS$X_INPUT_MODEL logical 20
4.8 Metafile Logging 20
4.9 Sample User-Written Device Handler 20
4.10 New Escape: Inquire Version 21
4.11 HP LaserJet II Support 22
4.12 VAX Float support 22
4.13 New Entrypoints in GKS$/GKS3D$ Bindings 22
4.14 Ada Support 24
4.15 Pascal Support 24
4.16 Encapsulated PostScript 24
4.17 ISO-Latin1 Character Support 24
4.18 Motif Support 25
4.18.1 Using Motif Window Manager Resources 25
4.18.2 GKS Motif Problems 26
4.18.3 Avoiding Window Problems 26

5 Areas of Nonconformance 27
5.1 Initial Input Prompt Position for STROKE and LOCATOR 27

6 Restrictions 28
6.1 General Restrictions 28
6.1.1 Output Drawing Performance Degradation 28
6.1.2 Line Cap Style 28
6.1.3 Limitation with STRING Precision Text Strings 28
6.1.4 Fill Area Set 28
6.1.5 Fill Area and Fill Area Set 28
6.1.6 Fill Area Interior Style Pattern 28
6.1.7 GDP Restricted Text 28
6.1.8 Widget Callback Procedure Name and Motif 29
6.1.9 Motif Workstation Types 29
6.1.10 Ada Language Restrictions 29
6.1.11 Opening Multiple Motif Workstations 29

7 Fixed Problems 30
7.1 Problems fixed in V7.2 30
7.2 Problems fixed since V6.5 30
7.2.1 Extra Page or Output File 30
7.2.2 Choice Menu Item Not Selectable 30
7.2.3 Rangecheck Error when Printing or Converting PostScript file 30

8 Known Problems 31
8.1 General Known Problems 31
8.1.1 DDIF Files and DECwrite 31
8.1.2 Hardware Clipping on Motif Devices 31
8.1.3 Spikes Appear for Some Characters 31
8.1.4 Clipping of Borders 31
8.1.5 Inquiry Workstation Functions 31
8.1.6 PostScript Workstations 61 and 62 Leading Blank Pages 31

9 GKS Documentation 32
9.1 General Problems 32
9.1.1 Number of Escapes 32
9.1.2 Initial Cursor Position for the String Device 32
9.1.3 INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER . . . 32
9.1.4 Locator PET –13 32
9.1.5 Changing the Title String 32
9.2 DEC GKS User's Guide 33
9.3 DEC GKS GKS$ Binding Reference Manual 33
9.3.1 Choice Class 33
9.3.2 Locator Class 34
9.3.3 Pick Class 37
9.3.4 String Class 37
9.3.5 Stroke Class 38
9.3.6 Valuator Class 40
9.4 Device Specifics Reference Manual for DEC GKS and DEC PHIGS 40
9.4.1 GKS_PREDEF Program Example 40

9.5 DEC GKS GKS3D$ Binding Reference Manual 41
9.5.1 INQUIRE STROKE DEVICE STATE 41
9.5.2 INQUIRE STROKE DEVICE STATE 3 41

10 Copyright 42

TABLES

1 VSI GKS Release Notes v
2 Changed Escapes 5
3 Descriptor Creation Entrypoints 22
4 Parameters 23

Preface

This document is the release notes for VSI GKS Version 7.3 on OpenVMS Integrity servers. It describes completed bug fixes, and highlights changes and corrections to the GKS documentation set.

This document is supplied in both PostScript and text form. You can find the release notes in the following locations:

Table 1: VSI GKS Release Notes

Text	SYS$HELP:VSII64GKS073.RELEASE_NOTES
PostScript	SYS$HELP:VSII64GKS073_RELEASE_NOTES.PS

Associated Documents

- *Installing VSI GKS for OpenVMS Integrity servers Systems*
- *DEC GKS User's Guide*
- *DEC GKS C Binding Reference Manual*
- *DEC GKS FORTRAN Binding Reference Manual*
- *DEC GKS GKS3D$ Binding Reference Manual*
- *DEC GKS GKS$ Binding Reference Manual*
- *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*

1 Product Summary

VSI GKS is a library of graphics functions to be included in application programs.

VSI GKS Version 7.3 complies with the international standard for three-dimensional graphics (GKS–3D International Standard (ISO 8805(E) 1988)), and is an upwardly compatible extension to the ISO GKS standard (ISO 7942-1985).

VSI GKS is complementary to the PHIGS graphics system; GKS is suitable for unstructured, fixed images, while PHIGS is used for structured, editable graphics data.

In this document, VSI GKS refers to Version 7.3.

1.1 Release Description

VSI GKS Version 7.3 is a release of VSI GKS on OpenVMS Integrity servers and OpenVMS Alpha servers. It is essentially a port of DIGITAL GKS Version 6.5, released in 1997, with the following differences:

- VSI GKS is provided as a POLYCENTER Software Installation (PCSI) installation kit;
- VAX float and IEEE floating point arithmetic are supported;

At present, VSI GKS Version 7.3 for OpenVMS Integrity servers supports the following Device types.

- Motif
- PostScript
- Color PostScript
- HP-GL
- LaserJet (HP PCL)
- ReGIS
- Tektronix 4014
- Tektronix 4107, 4128, 4129, and 4207
- Color Sixel (LJ250, LA324)
- Black & White Sixel (LN03, LA34, LA50, LA100, DEClaser 2100 & 2200)
- DDIF
- CGM

1.2 Operating System Support

VSI GKS Version 7.3 for OpenVMS Integrity servers supports OpenVMS Integrity servers Version 8.4-1H1 and higher. Applications built with earlier versions of the operating system may need to be relinked.

1.3 Layered Product Rebranding

VSI has rebranded and released new versions of many HPE layered products. Note that while the VSI version of a layered product is a new version, its functionality is the same as the HPE application that it replaces.

Copyright notices in layered products have been modified from HPE to VSI. However, the text that appears in some help files, display screens, or other areas may still say HPE. This may be modified in a future release.

1.4 Motif Version 1.7C Support

This release of VSI GKS supports Motif Version 1.7C and higher.

NOTE

You may need to set the Mwm*clientAutoPlace resource to False in the Motif Window Manager resource file on the server. The name of this file is DECW$MWM.DAT. Doing so will ensure correct window placement.

If, on successive runs of a GKS application, you see messages like this:

```
VSI GKS - WARNING - 'BORDER SIZE' attribute incorrect: use
"DEFINE GKS$DECW_BORDER_SIZE 224" and restart application.
VSI GKS - WARNING - 'TITLE SIZE' attribute incorrect: use
"DEFINE GKS$DECW_TITLE_SIZE -196" and restart application.
```

with numbers that vary even if you do not redefine the logicals as suggested, then you should apply the Mwm*clientAutoPlace fix.

2 Changes for Alpha AXP and Integrity server Systems

The escape, input initialization, and input inquiry functions were changed in DEC GKS Version 5.1 to expect a structure for the data record argument. This section describes the changes necessary to use GKS. Old applications may not have to be recoded at this time. However, VSI recommends that you recode the affected functions to the new format to gain maximum portability between supported platforms.

2.1 Changes to Escapes

The addition of 64-bit addresses and data types required changes to the escape data record. In previous versions of DEC GKS, pointers were returned in the integer array, as were some data objects (such as XIDs), which were declared long in C. On OpenVMS Alpha and OpenVMS Integrity servers, an integer, an address, and a C long are all 32-bit quantities. Its previous form (in a C description) was as follows:

```
typedef struct {
    Gint     nints;           /* Number of integers */
    Gint     nfloats;         /* Number of floats */
    Gint     nstrings;        /* Number of strings */
    Gint     *int_array;      /* Pointer to integer array */
    Gfloat   *float_array;    /* Pointer to float array */
    Gint     *str_len_array;  /* Pointer to string length array */
    Gchar   **str_ptr_array;  /* Pointer to string pointer array */
} Gescaperecord;
```

The data structure now contains four new items:

- A count of generic pointers
- An array of generic pointers
- A count of longs
- An array of longs

The new data structure is as follows:

```
typedef struct {
    Gint        nints;             /* Number of integers */
    Gint        nfloats;           /* Number of floats */
    Gint        nstrings;          /* Number of strings */
    Gint        *int_array;        /* Pointer to integer array */
    Gfloat      *float_array;      /* Pointer to float array */
    Gint        *str_len_array;    /* Pointer to string length array */
    Gchar       **str_ptr_array;   /* Pointer to string pointer array */
    Gint        npointers;         /* Number of generic pointers */
    Gchar       **ptr_ptr_array;   /* Pointer to generic pointer array */
    Gint        nlongs;            /* Number of longs */
    Glong       *long_array;       /* Pointer to long array */
} Gescaperecord;
```

2.1.1 Affected Escapes

OpenVMS Alpha and OpenVMS Integrity servers still support the old escape record format, but the newer format is preferred. Escapes that previously passed pointers or objects (such as XIDs) declared long in C will require require the most changes. These escapes are as follows:

- Inquire Window Identifiers (–304)

- Inquire Pasteboard Identifier (–307)
- Inquire Menu Bar Identifier (–308)
- Inquire Shell Identifier (–309)
- Inquire GDP Extent (–404)
- Set Background Pixmap (–501)
- Inquire Double Buffer Pixmap (–502)
- Inquire Background Pixmap (–503)

Table 2 describes the new escape data records.

Table 2: Changed Escapes

Argument	Required Value
–304 Inquire Window Identifiers	
in_data	Number of integers (1)
	Number of reals (0)
	Number of strings (0)
	Address of integer array (ws_id)
	Address of real array (NULL)
	Address of string lengths array (NULL)
	Address of string pointer array (NULL)
	Number of pointers (0)
	Address of generic pointer array (NULL)
	Number of longs (0)
	Address of long array (NULL)
out_data	Number of integers (0)
	Number of reals (0)
	Number of strings (0)
	Address of integer array (NULL)
	Address of real array (NULL)
	Address of string lengths array (NULL)
	Address of string pointer array (NULL)
	Number of pointers (1)
	Address of generic pointer array (X_display_id)
	Number of longs (1)
	Address of long array (X_window_id)

Table 2 (Cont.): Changed Escapes

Argument	Required Value
–307 Inquire Pasteboard Identifier	
in_data	Number of integers (1) Number of reals (0) Number of strings (0) Address of integer array (ws_id) Address of real array (NULL) Address of string lengths array (NULL) Address of string pointer array (NULL) Number of pointers (0) Address of generic pointer array (NULL) Number of longs (0) Address of long array (NULL)
out_data	Number of integers (0) Number of reals (0) Number of strings (0) Address of integer array (NULL) Address of real array (NULL) Address of string lengths array (NULL) Address of string pointer array (NULL) Number of pointers (1) Address of generic pointer array (Pasteboard_widget_id) Number of longs (0) Address of long array (NULL)

Table 2 (Cont.): Changed Escapes

Argument	Required Value
–308 Inquire Menu Bar Identifier	
in_data	Number of integers (1) Number of reals (0) Number of strings (0) Address of integer array (ws_id) Address of real array (NULL) Address of string lengths array (NULL) Address of string pointer array (NULL) Number of pointers (0) Address of generic pointer array (NULL) Number of longs (0) Address of long array (NULL)
out_data	Number of integers (0) Number of reals (0) Number of strings (0) Address of integer array (NULL) Address of real array (NULL) Address of string lengths array (NULL) Address of string pointer array (NULL) Number of pointers (1) Address of generic pointer array (Menu_bar_widget_id) Number of longs (0) Address of long array (NULL)

Table 2 (Cont.): Changed Escapes

Argument	Required Value
–309 Inquire Shell Identifier	
in_data	Number of integers (1)
	Number of reals (0)
	Number of strings (0)
	Address of integer array (ws_id)
	Address of real array (NULL)
	Address of string lengths array (NULL)
	Address of string pointer array (NULL)
	Number of pointers (0)
	Address of generic pointer array (NULL)
	Number of longs (0)
	Address of long array (NULL)
out_data	Number of integers (0)
	Number of reals (0)
	Number of strings (0)
	Address of integer array (NULL)
	Address of real array (NULL)
	Address of string lengths array (NULL)
	Address of string pointer array (NULL)
	Number of pointers (1)
	Address of generic pointer array (Shell_widget_id)
	Number of longs (0)
	Address of long array (NULL)

Table 2 (Cont.): Changed Escapes

Argument	Required Value
–404 Inquire GDP Extent	
in_data	Number of integers (4) Number of reals (0) Number of strings (0) Address of integer array (ws_id, number_of_points, GDP_id, sizeof(GDP_data_record)) Address of real array (NULL) Address of string lengths array (NULL) Address of string pointer array (NULL) Number of pointers (3) Address of generic pointer array (address of array of X values, address of array of Y values, address of GDP data record) Number of longs (0) Address of long array (NULL)
out_data	Number of integers (1) Number of reals (4) Number of strings (0) Address of integer array (status) Address of real array (minimum X, maximum X, minimum Y, maximum Y) Address of string lengths array (NULL) Address of string pointer array (NULL) Number of pointers (0) Address of generic pointer array (NULL) Number of longs (0) Address of long array (NULL)

Table 2 (Cont.): Changed Escapes

Argument	Required Value
–501 Set Background Pixmap	
in_data	Number of integers (1)
	Number of reals (0)
	Number of strings (0)
	Address of integer array (ws_id)
	Address of real array (NULL)
	Address of string lengths array (NULL)
	Address of string pointer array (NULL)
	Number of pointers (0)
	Address of generic pointer array (NULL)
	Number of longs (1)
	Address of long array (Pixmap_id)
out_data	Number of integers (0)
	Number of reals (0)
	Number of strings (0)
	Address of integer array (NULL)
	Address of real array (NULL)
	Address of string lengths array (NULL)
	Address of string pointer array (NULL)
	Number of pointers (0)
	Address of generic pointer array (NULL)
	Number of longs (0)
	Address of long array (NULL)

Table 2 (Cont.): Changed Escapes

Argument	Required Value
–502 Inquire Double Buffer Pixmap	
in_data	Number of integers (1) Number of reals (0) Number of strings (0) Address of integer array (ws_id) Address of real array (NULL) Address of string lengths array (NULL) Address of string pointer array (NULL) Number of pointers (0) Address of generic pointer array (NULL) Number of longs (0) Address of long array (NULL)
out_data	Number of integers (0) Number of reals (0) Number of strings (0) Address of integer array (NULL) Address of real array (NULL) Address of string lengths array (NULL) Address of string pointer array (NULL) Number of pointers (0) Address of generic pointer array (NULL) Number of longs (1) Address of long array (Pixmap_id)

Table 2 (Cont.): Changed Escapes

Argument	Required Value
–503 Inquire Background Pixmap	
in_data	Number of integers (1)
	Number of reals (0)
	Number of strings (0)
	Address of integer array (ws_id)
	Address of real array (NULL)
	Address of string lengths array (NULL)
	Address of string pointer array (NULL)
	Number of pointers (0)
	Address of generic pointer array (NULL)
	Number of longs (0)
	Address of long array (NULL)
out_data	Number of integers (0)
	Number of reals (0)
	Number of strings (0)
	Address of integer array (NULL)
	Address of real array (NULL)
	Address of string lengths array (NULL)
	Address of string pointer array (NULL)
	Number of pointers (0)
	Address of generic pointer array (NULL)
	Number of longs (1)
	Address of long array (Pixmap_id)

2.1.2 How to Use the New Format

To use the new data record format, do the following:

1. Convert your arrays of integers to data records. The include files contain language-specific escape data records.
2. Make sure you are passing both an input and an output record. In previous versions of GKS, you could have a zero size output data record when no data was to be returned.
3. Load the appropriate sections of the data record for the escape to be performed. Make sure to enter 0 in all counts that are not used, and make sure unused pointers are a safe invalid value (such as NULL in C). This holds true for both the input and output data records, even if they are not used. If the escape is one of the escapes listed in Section 2.1, you must rearrange some of the data.
4. Make sure the sizes passed for the data records are correct. This is especially true if your sizes were previously hardcoded. Use a compiler or language feature (such as the C sizeof( ) operator) to get the size in bytes, because the size will vary depending on the architecture and operating system.

2.1.3 Documentation Changes

Some product documents explicitly state sizes in bytes. These are only correct for the older style of record. The stated sizes are *not* valid in the Alpha and Integrity server system environments, due to the addition of data in the escape and input records and the change in the size of some of the data objects.

3 Installation Notes

VSI GKS Version 7.3 for OpenVMS requires OpenVMS Integrity Version 8.4-1H1 or higher.

3.1 HP GKS or HP GKSRT must be removed before installing VSI GKS Version 7.3

If HP GKS or HP GKSRT are installed on your system they must be removed before installing VSI GKS. You can remove them manually using the following commands:

```
$ PRODUCT REMOVE GKS /PRODUCER=HP
or
$ PRODUCT REMOVE GKSRT /PRODUCER=HP
```

If you don't remove them manually, the installation procedure for VSI GKS will prompt you to remove them.

4 Previous Enhancements

This section describes the enhancements made in previous HPE versions of GKS for OpenVMS. These are included in VSI GKS Version 7.3.

4.1 Metafile Header Format Change

The format of the metafile header line has been updated to include a four-digit year field and to be uniform across all platforms. The new format is designated Version 3. Old-format (Version 2) metafiles can still be read by VSI GKS.

The new format is as follows:

```
0         1         2         3         4         5         6         7
12345678901234567890123456789012345678901234567890123456789012345678901234567890
GKS3   username     /DEC GKS   Version 6.4  1996/07/25 3 4 3 81214 1 1
^    ^author designation (39 bytes)         ^date      ^ ^ ^ ^ ^ ^
^item header                                 format version^ ^ ^ ^ ^ ^
.                                            item header size^ ^ ^ ^ ^
.                                                item type size^ ^ ^ ^
.                                                data length size^ ^ ^
.                                                integer field size^ ^
.                                           floating-point field size^
```

4.2 NIST Certification

GKS V6.0 on OpenVMS Alpha was certified by the U.S. National Institute of Standards and Technology. GKS was certified on and for a restricted subset of all supported workstations:

WS type	Description
231	Motif output/input
61	Black & white PostScript

Certain changes in the behavior of VSI GKS were necessary for certification; all these changes are enabled by defining the GKS$NIST_CERTIFIED logical to be any non-empty value:

- Valid workstations are restricted to the list above; this change affects Open Workstation, Inquire List of Workstation Types, and the WDT inquiry functions.
- The error returned in certain error conditions is different:
  - Normalization transformation number invalid in Initialize Locator or Initialize Stroke;
  - Initial point(s) invalid in Initialize Locator or Initialize Stroke;
  - Number of points less than zero in Initialize Stroke;
  - Initial string larger than buffer in Initialize String;
  - Initial segment name less than or equal to zero, or initial pick identifier less than zero, in Initialize Pick or Initialize Pick 3;
  - GKS not in proper state in any of the Initialize Input routines.

- The Motif workstation is mapped at Open Workstation time.
- The number of marker types is restricted to 18.
- Separate viewport data for both current and requested viewports is maintained for inquiries.
- IRGs are not allowed to be optimized.
- Deferral functionality is changed: previously, output primitives were deferred (i.e. not sent to the workstation) if an IRG was pending. If the GKS$NIST_CERTIFIED logical is in effect, output primitives are **not** deferred.

4.3 New Escape: Set Fill Simplification Method

Digital GKS Version 6.3 introduced a new escape, *Set Fill Simplification Method*, to provide an application-level method of choosing the fill simplification method. The syntax of the Set Fill Simplification Method escape is:

–140 Set Fill Simplification Method Arguments:

Argument	Required Value
in_data	number of integers (2) number of reals (0) number of strings (0) address of integer array (*ws_id, fill_simplification_method*) address of real array (NULL) address of string lengths array (NULL) address of string pointer array (NULL) number of pointers (0) address of generic pointer array (NULL) number of longs (0) address of long array (NULL)
out_data	number of integers (0) number of reals (0) number of strings (0) address of integer array (NULL) address of real array (NULL) address of string lengths array (NULL) address of string pointer array (NULL) number of pointers (0) address of generic pointer array (NULL) number of longs (0) address of long array (NULL)

The input data record contains the following elements:

Component	Description
ws_id	The workstation ID
fill_simplification_method	0 = default 1 = MapKernel

4.4 New Escape: Hit Test

Digital GKS Version 6.2 introduced a new escape, *Hit Test*, to provide programs with an alternative asynchronous method (other than Get Pick) for determining whether a segment is picked. For example, you can input a position asynchronously via Get Locator, then use Hit Test to detect a segment hit.

You specify a workstation, a transformation, a location, and an optional aperture as input. The location may be either a two-dimensional or a three-dimensional point in world coordinates. However, if you wish to specify an aperture (in device coordinates, as for Initialize Pick), you must specify a Z-value for the location. For two-dimensional applications, specify 0.0 as the Z-value.

The syntax of the Hit Test escape is:

–390 Hit Test Arguments:

Argument	Required Value
in_data	number of integers (2) number of reals (2-4) number of strings (0) address of integer array (*ws_id, transform*) address of real array (*x, y, z (optional), aperture(optional)*) address of string lengths array (NULL) address of string pointer array (NULL) number of pointers (0) address of generic pointer array (NULL) number of longs (0) address of long array (NULL)
out_data	number of integers (3) number of reals (0) number of strings (0) address of integer array (*status, segment, pick_id*) address of real array (NULL) address of string lengths array (NULL) address of string pointer array (NULL) number of pointers (0) address of generic pointer array (NULL) number of longs (0) address of long array (NULL)

The input data record contains the following elements:

Component	Description
ws_id	The workstation ID
transform	The normalization transformation number
x, y, z	The point in world coordinates
aperture	The pick aperture

The output data record contains the following elements:

Component	Description
status	The error status; 0 = no error
segment	The segment number
pick_id	The pick ID

4.5 X11R5 Scalable Font Support

X11R5 Scalable Fonts are supported. Support is enabled by setting the GKS$USE_SCALABLE_FONTS logical to the value 1. Any other value results in scalable font support *not* being enabled.

4.6 Inquire DBuffer Pixmap & Inquire Background Pixmap supported

The Inquire Double Buffer Pixmap (–502) and Inquire Background Pixmap (–503) escapes are supported.

4.7 New Environmental Controls

4.7.1 GKS$FILL_SIMPLIFY_METHOD / GKS$COMPLEX_FILL_METHOD

The GKS$FILL_SIMPLIFY_METHOD / GKS$COMPLEX_FILL_METHOD logicals are used to control fill simplification. Defining either of these logicals to be one (1) enables the MapKernel fill simplification method. MapKernel is currently supported only for two-dimensional geometry. MapKernel is intended for applications (like mapping) which involve extremely complex fill areas with holes, "bridges," and self-intersections. In general, MapKernel simplification is slower than the default method, but it will generate correct output in cases where the default method will not (see "Complex Fill Areas" in the "Problems Fixed" section).

4.7.2 The GKS$HATCH_SIMULATION logical

The GKS$HATCH_SIMULATION logical is used to control hatch simulation. GKS interior hatch styles may be *simulated* in a device-independent way, or they may be drawn by the output device (*if* the output device supports hatch styles). Setting the GKS$HATCH_SIMULATION logical to any non-null value forces GKS to do the device-independent hatch simulation regardless of whether the output device supports hatch styles.

4.7.3 The GKS$VISUAL_CLASS logical

The GKS$VISUAL_CLASS logical is used to control the class of visual selected for the Motif workstation types (230, 231). Normally, GKS selects the "default" visual provided by the display node's X server. However, sometimes it is desirable to override the X server's choice of visual and use a visual with different characteristics. For example, you may have an application which depends on rapid color changes. This is easily achieved without screen regeneration, via the GKS Set Color Representation call, on PseudoColor visuals. However, on displays with advanced graphics options, e.g. the ZLX(p)-E, ZLX-M, and ZLX(p)-L cards, the default is usually a TrueColor visual. TrueColor visuals do *not* support dynamic color redefinition, so each Set Color Representation call results in an implicit regeneration, dramatically impacting your application's performance. In this case, you can maximize performance by defining GKS$VISUAL_CLASS to have any of the values "P," "Pseudo," or "PseudoColor."

GKS$VISUAL_CLASS takes on string values as shown in the following table. Case is ignored.

Value	Visual Class Selected
TrueColor	TrueColor
True	
T	
PseudoColor	PseudoColor
Pseudo	
P	
DirectColor	DirectColor
Direct	
D	
GrayScale	GrayScale
Gray	
G	
StaticColor	StaticColor
SC	
StaticGray	StaticGray
SG	

4.7.4 The GKS$X_INPUT_MODEL logical

The GKS$X_INPUT_MODEL logical is used to control the X input model selected for the Motif workstation types (230, 231). The X input models (No Input, Passive, Locally Active, and Globally Active) are described in the Inter-Client Communication Conventions Manual, section 4.1.7. An application's input model determines how the application handles input focus.

The GKS$X_INPUT_MODEL logical takes on string values as shown in the following table. Case is ignored.

Value	X Input Model Selected
None	No Input
No_Input	
N	
Passive	Passive Input
P	
Local	Locally Active
Locally_Active	
L	
Global	Globally Active
Globally_Active	
G	

The default input model in GKS is Globally Active. However, in some circumstances it is useful to use simpler input models. For example, some low-end graphics options (e.g. ZLX-E1) do not support multiple simultaneous colormaps. So, if your GKS application creates a new, different colormap, the application may not display in the correct colors unless you change the X input model to Passive or Locally Active *and* set focus to your application's window via clicking or moving the mouse pointer into it.

4.8 Metafile Logging

To use the metafile logging feature, define the GKS$META_LOG logical to be the filename of the file to which you wish to send metafile logging output, then run your GKS application. GKS calls will be logged to the output file as if it were a normal metafile output workstation (wstype 2) with additional data reflecting changes in deferral state and input-related transactions.

4.9 Sample User-Written Device Handler

For instructions on building the sample device handler and running the sample application, please read the README.FIRST file in the GKS$EXAMPLES directory.

4.10 New Escape: Inquire Version

DEC GKS Version 6.0 introduced a new escape, *Inquire Version*, to enable programs to find out certain information about the version of GKS with which they are running. The syntax of the Inquire Version escape is:

–399 Inquire Version Arguments:

Argument	Required Value
in_data	number of integers (0) number of reals (0) number of strings (0) address of integer array (NULL) address of real array (NULL) address of string lengths array (NULL) address of string pointer array (NULL) number of pointers (0) address of generic pointer array (NULL) number of longs (0) address of long array (NULL)
out_data	number of integers (5) number of reals (1) number of strings (1) address of integer array (*error_status, IEEE/VAXfloat indicator, version_number:major, version_number:minor, version_number:ECO/MUP level*) address of real array (*always_1.0*) address of string lengths array (*length of version_string*) address of string pointer array (*version_string*) number of pointers (0) address of generic pointer array (NULL) number of longs (0) address of long array (NULL)

The output data record contains the following elements:

Component	Description
error_status	The error status; 0 = no error
IEEE/VAXfloat	1 = IEEE, 2 = VAXfloat
version:major	The number to the left of the decimal point in the version number
version:minor	The number to the right of the decimal point in the version number
version:ECO/MUP	(If present) trailing letter; 1 = A, 2 = B, etc.; otherwise zero
always_1.0	Used to verify linking consistency on OpenVMS Alpha
version_string	"VSI GKS V7.3"

4.11 HP LaserJet II Support

The Hewlett-Packard LaserJet II printer, and other devices utilizing HP PCL Printer Language, are now supported. Workstation type 261 indicates a PCL device. Type $ HELP GKS for more information.

4.12 VAX Float support

VAX Float arithmetic is supported on VSI GKS for OpenVMS Integrity servers, in addition to the default IEEE floating point format. To make use of VAX Float arithmetic, you must compile your source modules with the /FLOAT=G_FLOAT or /FLOAT=D_FLOAT option.

Linking is also different. You must link your program with one of the linker options files provided with GKS, as in the following examples:

```
IEEE Float (default)
$ LINK program,SYS$LIBRARY:GKSCBND/OPT      !C binding
$ LINK program,SYS$LIBRARY:GKSFORBND/OPT    !Fortran binding
$ LINK program,SYS$LIBRARY:GKS$BND/OPT      !GKS$ binding
$ LINK program,SYS$LIBRARY:GKS3D$BND/OPT    !GKS3D$ binding

VAX Float
$ LINK program,SYS$LIBRARY:GKSCBND_VF/OPT      !C binding
$ LINK program,SYS$LIBRARY:GKSFORBND_VF/OPT    !Fortran binding
$ LINK program,SYS$LIBRARY:GKS$BND_VF/OPT      !GKS$ binding
$ LINK program,SYS$LIBRARY:GKS3D$BND_VF/OPT    !GKS3D$ binding
```

4.13 New Entrypoints in GKS$/GKS3D$ Bindings

Several entrypoints exits to facilitate the creation of descriptors in C, Fortran, or Ada programs using the GKS$ or GKS3D$ binding.

Table 3: Descriptor Creation Entrypoints

Entrypoint Name	Description
gks$init_1d_int_desc	Initialize 1D integer array descriptor
gks$init_1d_real_desc	Initialize 1D float array descriptor
gks$init_2d_int_desc	Initialize 2D integer array descriptor
gks$init_2d_real_desc	Initialize 2D float array descriptor
gks$init_fort_2d_int_desc	Initialize Fortran 2D integer array descriptor
gks$init_fort_2d_real_desc	Initialize Fortran 2D float array descriptor
gks$init_string_desc	Initialize character string descriptor
gks3d$init_1d_int_desc	Initialize 1D integer array descriptor
gks3d$init_1d_real_desc	Initialize 1D float array descriptor
gks3d$init_2d_int_desc	Initialize 2D integer array descriptor
gks3d$init_2d_real_desc	Initialize 2D float array descriptor
gks3d$init_fort_2d_int_desc	Initialize Fortran 2D integer array descriptor
gks3d$init_fort_2d_real_desc	Initialize Fortran 2D float array descriptor
gks3d$init_string_desc	Initialize character string descriptor

The interface is as follows (in C terms); parameters are described in the following table.

```
gks$init_1d_int_desc( iarray, lbound, ubound, desc );
gks$init_1d_real_desc( rarray, lbound, ubound, desc );
gks$init_2d_int_desc( i2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks$init_2d_real_desc( r2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks$init_fort_2d_int_desc( i2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks$init_fort_2d_real_desc( r2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks$init_string_desc( string, length, desc );
gks3d$init_1d_int_desc( iarray, lbound, ubound, desc );
gks3d$init_1d_real_desc( rarray, lbound, ubound, desc );
gks3d$init_2d_int_desc( i2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks3d$init_2d_real_desc( r2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks3d$init_fort_2d_int_desc( i2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks3d$init_fort_2d_real_desc( r2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks3d$init_string_desc( string, length, desc );
```

Table 4: Parameters

Data Type	Name	In/Out	Description
int	*iarray	In	1-D Array of integers
float	*rarray	In	1-D Array of reals
int	*i2array	In	2-D Array of integers
float	*r2array	In	2-D Array of reals
int	*lbound	In	Lower bound
int	*ubound	In	Upper bound
int	*lbound1	In	Row lower bound
int	*ubound1	In	Row upper bound
int	*lbound2	In	Column lower bound
int	*ubound2	In	Column upper bound
int	*length	In	Length of character string
gks3d_1darray_desc	*desc	Out	1-D Array Descriptor
gks3d_2darray_desc	*desc2d	Out	2-D Array Descriptor

NOTE

These routines check the alignment of the descriptor (output) argument. The user is responsible for providing a suitably aligned (and sized) array or data structure for the descriptor argument. A 72-byte array will suffice, if correctly aligned. Correct alignment in Fortran for all platforms can be achieved as follows:

```
C     Fragment of Cell Array Example
      REAL*4        CORN1(3)
      REAL*4        CORN2(3)
      REAL*4        CORN3(3)
      INTEGER*4     DESC(18)
      REAL*8        ALIGNME
      EQUIVALENCE  (ALIGNME, DESC(1))
      INTEGER*4     COLORS(100,200)
      ...
      CALL  GKS3D$INIT_FORT_2D_INT_DESC( COLORS, 1, 100, 1, 200, DESC )
      CALL  GKS3D$CELL_ARRAY3( CORN1, CORN2, CORN3,
     *                         1, 1, 100, 200, DESC )
```

4.14 Ada Support

Development of Ada programs using an Ada interface to the GKS$ and GKS3D$ bindings is supported on all platforms. See the GKS3D_DEFS.ADA and GKSDEFS.ADA files provided with the VSI GKS kit.

NOTE

Use the INIT_*_DESC routines to aid in constructing the GKS3D_DESCRIPTOR or GKS_DESCRIPTOR records.

For maximum portability, use the GKS*DESCRIPTOR alternatives on all platforms.

4.15 Pascal Support

A number of environment files SYS$LIBRARY:GKS*.PEN are provided with the VSI GKS kit. The GKS*.PAS source files are provided as well.

Should it become necessary to rebuild the environment files, some additional steps are necessary:

1. Create a file (usually an empty file, unless your installation has specific requirements):

```
$ OPEN/WRITE EMPTYFILE EMPTY.FILE
$ CLOSE EMPTYFILE
```

2. Define a logical name <name>_MODULE_INIT pointing to the file just created:

```
$ DEFINE GKS3D_DEFS_MODULE_INIT EMPTY.FILE
```

3. Compile the GKS*.PAS file as usual:

```
$ PASCAL/ENVIRONMENT=SYS$LIBRARY:GKS3D_DEFS.PEN GKS3D_DEFS.PAS
```

The environment files (.PEN) provided with the GKS Development kit are IEEE compatible on IA64 and VAX_FLOAT compatible on Alpha. To create .PEN files using a different float type, complete Step 1 and Step 2, and then specify the /FLOAT qualifier in the PASCAL command (Step 3) as shown:

```
$ PASCAL/ENVIRONMENT=SYS$LIBRARY:GKS3D_DEFS.PEN/FLOAT=<float_type> GKS3D_DEFS.PAS
```

4.16 Encapsulated PostScript

VSI GKS Version 7.3 supports encapsulated PostScript. Use either workstation type 65 (black and white) or 66 (color).

4.17 ISO-Latin1 Character Support

The PostScript device handler supports ISO-Latin1 characters. To enable this feature on OpenVMS systems, set the GKS$ISO_ENCODE logical to 1. By default, this feature is disabled (0).

Font IDs –101 to –129 are supported for ISO-Latin1. Spacing and positioning errors may occur if you use the other font IDs.

No special setup is required to output ISO-Latin1 characters on OpenVMS printer queues.

4.18 Motif Support

This release supports DECwindows Motif. All the functionality provided by the DECwindows workstation types (21*x*) is provided by the Motif workstation types (23*x*).

The Motif workstation types are as follows:

- 230—Output only
- 231—Output and input
- 232—X drawable
- 233—Motif widget
- 330—Japanese Output only
- 331—Japanese Output and input
- 332—Japanese X drawable
- 333—Japanese Motif widget

4.18.1 Using Motif Window Manager Resources

The following resources for the Motif window manager are useful for controlling the focus of windows created by VSI GKS:

- Mwm*keyboardFocusPolicy (Explicit, Pointer)—Set this resource to Explicit to determine window focus by clicking on the window or border. Set it to Pointer to determine window focus by the location of the pointer.
- Mwm*autoKeyFocus (True, False)—If keyboardFocusPolicy is set to Explicit, set this resource to True to return focus to the window originally in focus. Set it to False to avoid giving focus to any window when the window currently with focus is iconified or removed.
- Mwm*deiconifyKeyFocus (True, False)—If the keyboardFocusPolicy is set to Explicit, set this resource to True to give a window focus when it is restored from an icon. Set it to False to retain the current focus when a window is deiconified. Because GKS iconifies a window when it is created and then deiconifies it, use the deiconifyKeyFocus resource to control the initial focus of the application window.
- Mwm*focusAutoRaise (True, False)—Set this resource to True to raise a window to the top of the stack when it is given focus. Set it to False to retain the current stacking order when a window is given focus.
- Mwm*startupKeyfocus (True, False)—If the keyboardFocusPolicy is set to Explicit, set this resource to True to give input focus to a window when it is mapped. Set it to False to avoid giving focus to a window when it is mapped.
- Mwm*clientAutoPlace (True, False)—You may need to set this resource to False to ensure correct window placement when you are running your application on an OpenVMS client and displaying to an OpenVMS server running Motif Version 1.2.

4.18.2 GKS Motif Problems

The following problems exist with the VSI GKS Motif device handler:

- On OpenVMS systems, using the "CLOSE" option on the title bar kills the application. There is no method to disable this behavior.
- String input device number 3 (ASCII) with echoing ON does not trigger on each character (as it does in NOECHO mode).
- Choice devices 2, 3, 4, and 5 with echoing ON respond to their respective keyboard and mouse buttons, as well as selection of the menu with the mouse. They should only respond to the keyboard.
- Choice device 9 with NOECHO appears to have no possible trigger.
- Choice device 9 with echo ON triggers with any mouse movement over the window region.
- The RESET button is an item in the menu bar. This is incorrect for Motif style guide conformance, which requires the button to be an item in a pulldown menu with the name of the pulldown menu listed in the menu bar. This will be addressed in a future release.

4.18.3 Avoiding Window Problems

If you are using the Motif workstation types , you may see the following messages when running applications:

```
VSI GKS - 'MENU BAR SIZE' attribute incorrect: use
"DEFINE GKS$DECW_MENU_BAR_SIZE 30" and restart application.

VSI GKS -'BORDER SIZE' attribute incorrect: use
"DEFINE GKS$DECW_BORDER_SIZE 16" and restart application.

VSI GKS -'TITLE SIZE' attribute incorrect: use
"DEFINE GKS$DECW_TITLE_SIZE 17" and restart application.
```

If you see these messages, VSI GKS is making assumptions about the size of the window manager borders, title bar, and the application menu bar, based on the window manager and the resolution of the screen. VSI GKS uses these values to calculate the maximum display space remaining for the application to use. If for some reason these assumptions are not correct, VSI GKS displays these error messages. Digital suggests that you follow the error message instructions and restart the application. If you do not follow the instructions, you may have the following problems:

- The default positioning will not be correct.
- The application window may not fill the screen (if requested).
- The RESET button may not properly reset the window to the correct location and size.
- Scroll bars may be displayed when you use the default window size.
- Resizing the window may result in parts of the graphic output being clipped.

It is recommended that you use the Motif device handler only with a Motif window manager.

5 Areas of Nonconformance

This section describes areas of nonconformance in VSI GKS.

5.1 Initial Input Prompt Position for STROKE and LOCATOR

VSI GKS implements PETs in a windowing environment using a pointer device (a mouse). It is against the philosophy of a windowing system to position the mouse from the application. In VSI GKS, the initial locator and stroke positions are not prompted at the initial position, as required by the GKS standard.

6 Restrictions

This section describes the VSI GKS Version 7.3 restrictions.

6.1 General Restrictions

This section describes the general restrictions to VSI GKS.

6.1.1 Output Drawing Performance Degradation

When input is active (SAMPLE or EVENT mode) and the active PETs have to be simulated by VSI GKS, output drawing performance is slow. Digital recommends that you turn off input before you call a group of GKS output primitives.

6.1.2 Line Cap Style

The line cap style attribute is applied only to lines that have the SOLID linestyle attribute.

6.1.3 Limitation with STRING Precision Text Strings

If text of precision STRING contains invalid character codes, the text is drawn with the default stroke font number (1).

6.1.4 Fill Area Set

The fill area interior style is always simulated if the fill area set includes more than one fill area. This results in reduced performance. Also, the fill area interior style PATTERN is not supported with the function FILL AREA SET if there is more than one fill area defined in the set.

6.1.5 Fill Area and Fill Area Set

FILL AREA and FILL AREA SET calls not defining planar fill areas can result in an unexpected failure of VSI GKS. VSI GKS only provides support for planar fill areas and fill area sets (as required by the standard).

6.1.6 Fill Area Interior Style Pattern

The following pattern functions are provided, but have no effect on output:

- SET PATTERN SIZE
- SET PATTERN REFERENCE POINT AND VECTORS
- SET PATTERN REFERENCE POINT

The fill area interior style PATTERN is not supported with the function FILL AREA SET if there is more than one fill area defined in the set.

6.1.7 GDP Restricted Text

The Restricted Text GDP function is not supported by VSI GKS Version 7.3.

6.1.8 Widget Callback Procedure Name and Motif

When using widget callback procedures for the Motif workstation type, avoid using the name "gfx_create_proc". This is already used by the Motif UIL files and a name collision will occur.

6.1.9 Motif Workstation Types

The workstation description table inquiry functions for the Motif workstation types require operating states WSOP, WSAC, and SGOP. To call these inquiries while not in these states, use escape –110, Associate Workstation Type Connection ID.

6.1.10 Ada Language Restrictions

Applications using Ada tasks should make sure that all VSI GKS calls are performed from the same Ada task, or that proper synchronization is done.

6.1.11 Opening Multiple Motif Workstations

Applications that open multiple Motif workstations of type 230, 231, 240, or 241 must use the same value for the workstation type each time. There is a limitation in the device handlers that breaks X event handling if the workstation type value is different. For example, opening two workstations using type %x000a00e7 for each is acceptable, but opening one with %X000000e7 and another with %x000a00e7 is not (workstation type 231 with the default number of colors and with 10 colors). This will be addressed in a future release.

7 Fixed Problems

This section describes problems fixed since the last release.

7.1 Problems fixed in V7.2

HP GKS Run-Time kit does not recognize the GKS-RT license.

7.2 Problems fixed since V6.5

7.2.1 Extra Page or Output File

For NIST conformance, the behavior of Set Color Representation was changed such that setting the representation of the first color index caused the display to become NOT EMPTY even if nothing had been drawn yet.

This change caused hardcopy devices to emit a blank page (e.g. wstype 62, PostScript) or an extra output file (e.g. wstype 66, Encapsulated PostScript) if Set Color Representation was called for color #0 before anything was drawn.

This behavior has now been placed under the control of the GKS$NIST_CERTIFIED logical. If this logical is NOT in effect, no blank page or extra file will be generated.

7.2.2 Choice Menu Item Not Selectable

In some situations, an item on a choice menu that should have been active (selectable) was not selectable.

This behavior occurred in situations like the following:

1. Application brings up a choice menu in Request Mode;
2. User clicks on one of the choices and leaves the mouse in place;
3. Application brings up another choice menu, replacing the first one;
4. The choice item under the mouse pointer is now unselectable; the user must move the mouse away and then back to make the item selectable.

7.2.3 Rangecheck Error when Printing or Converting PostScript file

A rangecheck error occurred when a Japanese PostScript file was sent to certain printers or to a PostScript-to-PDF converter like Adobe Acrobat Distiller.

The problem was that newer printers and the conversion tools expect CID fonts, rather than the OCF fonts emitted by GKS.

VSI GKS now supports CID fonts.

8 Known Problems

This section describes known technical problems in this release.

8.1 General Known Problems

This section describes the known general technical problems with VSI GKS Version 7.3.

8.1.1 DDIF Files and DECwrite

When a DDIF file is imported into the DECwrite product, the image is redrawn 3 times.

8.1.2 Hardware Clipping on Motif Devices

There is a known problem with the hardware clipping of GDPs against the workstation viewport on Motif devices. The workaround is to use software clipping. Software clipping is the default for VSI GKS Version 7.3.

8.1.3 Spikes Appear for Some Characters

It is possible for spikes to appear when drawing STROKE and CHARACTER precision text strings. The characters A, M, and W have this problem. The problem occurs on OpenVMS systems containing the SPX graphics processor. The output window must be partially obscured for the problem to occur. The problem lies in the Xcfb server. The server has problems when drawing zero-width capped lines.

8.1.4 Clipping of Borders

Depending on the current workstation transformation or whether double buffering is on, the border pixels may be clipped. If clipping occurs, a line drawn along the workstation window border may not appear on the output device.

8.1.5 Inquiry Workstation Functions

If you call inquiry workstation functions before workstation types 232 or 233 are open, the program ends unexpectedly.

8.1.6 PostScript Workstations 61 and 62 Leading Blank Pages

The PostScript workstations 61 and 62 generate leading blank pages if the workstation transformation is changed prior to output.

9 GKS Documentation

The following documents exist for GKS:

- *Installing VSI GKS for OpenVMS Integrity servers Systems*
- *DEC GKS User's Guide*
- *DEC GKS C Binding Reference Manual*
- *DEC GKS FORTRAN Binding Reference Manual*,
- *DEC GKS GKS3D$ Binding Reference Manual*
- *DEC GKS GKS$ Binding Reference Manual*
- *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*
- *Building a Device Handler System for DEC GKS and DEC PHIGS*

The following section describes changes not included in the product manuals. The information is arranged by manual title.

9.1 General Problems

This section lists the changes that affect more than one manual.

9.1.1 Number of Escapes

The bindings manuals incorrectly state that there are 61 GKS escapes. There are 65 escapes.

9.1.2 Initial Cursor Position for the String Device

The string class input data record contains a component for the initial cursor position. Although most workstations ignore this value, the Motif workstation positions the cursor at the position indicated by this component.

9.1.3 INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER

The maximum number of normalization transformations is 256, numbered 0 to 255.

9.1.4 Locator PET –13

In the GKS$, C, and FORTRAN bindings, the section describing the locator PETs incorrectly describes PET 13. The PET number should be –13.

9.1.5 Changing the Title String

All the choice, string, and valuator PETs, and locator PET 6 allow you to change the title string.

The default title is the name of the device class. To change the string, increase the data record size by twice the size of a pointer and add the following components to the end of the data record:

Component	Description	Data Type	Used or Ignored
Nexttolast	Title string address	Address	U
Last	Title string length	Integer	U

If you want to have a blank title for a particular device, use a title string of one blank (`' '`) and a title string length of 1. If you do not specify the title string and title string length, VSI GKS uses the default title. If you set the title string address to NULL and the title string length to zero, VSI GKS uses the default title.

Be sure the data record length accurately reflects whether you want to use the title components.

9.2 DEC GKS User's Guide

The example programs in the *DEC GKS User's Guide* are included on the VSI GKS kit. The program names are as follows:

```
File name                  Corresponding manual example

user_manual_1_3.c          Example 2-1
user_manual_3_1.c          Example 6-1
user_manual_4_1.c          Example 3-1
user_manual_5_1.c          Example 4-1
user_manual_6_1.c          Example 5-1
user_manual_7_1.c          Example 5-2
```

9.3 DEC GKS GKS$ Binding Reference Manual

The gks3d$sizeof/gks$sizeof utility function, used to determine the size of an input data record, is available with this release of VSI GKS. The language-dependent data structures that define the input data record structures for each of the languages supported by the GKS$ binding are also available with this release. See the GKS3D$DEFS.*/GKS$DEFS.* include files to see what data structures and size macros are available for a particular language.

The following sections list all the input data records for the GKS$ binding. The tables within these sections include the order of the component within the input data record, a description of the component, the data type, and whether the workstation uses (U) or ignores (I) the input data record component. When you pass input data records in the INITIALIZE functions, the information order *must* match the order listed in the tables.

9.3.1 Choice Class

This section lists the input data record information required for choice-class prompt and echo types (PETs).

Choice-Class PET 2

The input data record size (INITIALIZE function argument *rec_size*) is 8. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Number of choice alternatives	Integer	U
2	Address of the array of prompts turned off (GKS$K_CHOICE_PROMPT_OFF) or on (GKS$K_CHOICE_PROMPT_ON)	Address	U

Choice-Class PETs –1, 1, and 3

The input data record size (INITIALIZE function argument *rec_size*) is 12. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Number of choice alternatives	Integer	U
2	Address of the array of choice string lengths	Address	U
3	Address of the array of choice string addresses	Address	U

9.3.2 Locator Class

This section lists the input data record information required for locator-class PETs.

Locator-Class PET –1

The input data record size (INITIALIZE function argument *rec_size*) is 8. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	X dimension of the box in world coordinate points	Real	U
2	Y dimension of the box in world coordinate points	Real	U

Locator-Class PETs –11, 1, 2, 3, and 6

The input data record size (INITIALIZE function argument *rec_size*) is 0. The input data record (INITIALIZE function argument *data_rec*) is a dummy record.

Locator-Class PETs –12, –10, –9, –5, –4, and 4

These prompt and echo types require you to use one of two input data records. Which input data record you use depends on the value of the attribute control flag. If the attribute control flag is GKS$K_ACF_CURRENT, the prompt and echo displays the current set of output attributes. If the attribute control flag is GKS$K_ACF_SPECIFIED, the prompt and echo displays the output attributes specified in the input data record.

To specify current output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 4. The input data record (INITIALIZE function argument *data_rec*) contains the component listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Attribute control flag (GKS$K_ACF_CURRENT)	Integer	U

To specify new output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 32. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Attribute control flag (GKS$K_ACF_SPECIFIED)	Integer	U
2	Line type ASF (GKS$K_ASF_BUNDLED or GKS$K_ ASF_INDIVIDUAL)	Integer	I
3	Line width scale factor ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL)	Integer	I
4	Polyline color index (GKS$K_ASF_BUNDLED or GKS$K_ ASF_INDIVIDUAL)	Integer	I
5	Polyline index	Integer	I
6	Line type index	Integer	U
7	Line width scale factor	Real	U
8	Polyline color index	Integer	I

Locator-Class PETs –2 and 5

Although four input data records are theoretically possible for these PETs, VSI GKS supports only two. The input data records depend on the values of the polyline-fill-area control flag, and on the attribute control flag. If the polyline-fill-area control flag is GKS$K_ ACF_POLYLINE, VSI GKS uses a polyline to draw the prompt and echo rectangle. If the polyline-fill-area control flag is GKS$K_ACF_FILL_AREA, VSI GKS uses a fill area to draw the prompt and echo. VSI GKS currently supports only the polyline rectangle.

If the attribute control flag is GKS$K_ACF_CURRENT, the prompt and echo displays the current set of output attributes. If the attribute control flag is GKS$K_ACF_SPECIFIED, the prompt and echo displays the output attributes specified in the input data record.

To specify current output attribute values and a polyline rectangle, the input data record size (INITIALIZE function argument *rec_size*) is 8. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Polyline-fill-area control flag (GKS$K_ACF_POLYLINE)	Integer	I
2	Attribute control flag (GKS$K_ACF_CURRENT)	Integer	U

To specify new output attribute values and a polyline rectangle, the input data record size (INITIALIZE function argument *rec_size*) is 36. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Polyline-fill-area control flag (GKS$K_ACF_POLYLINE)	Integer	I
2	Attribute control flag (GKS$K_ACF_SPECIFIED)	Integer	U
3	Line type ASF (GKS$K_ASF_BUNDLED or GKS$K_ ASF_INDIVIDUAL)	Integer	I
4	Line width scale factor ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL)	Integer	I

Component	Description	Data Type	Used or Ignored
5	Polyline color index ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL)	Integer	I
6	Polyline index	Integer	I
7	Line type index	Integer	U
8	Line width scale factor	Real	U
9	Polyline color index	Integer	I

Locator-Class PETs –8, –7, –6, and –3

These PETs require you to use one of two input data records. Which input record you use depends on the value of the attribute control flag. If the attribute control flag is GKS$K_ACF_CURRENT, the prompt and echo displays the current set of output attributes. If the attribute control flag is GKS$K_ACF_SPECIFIED, the prompt and echo displays the output attributes specified in the input data record.

To specify the current output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 20. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Attribute control flag (GKS$K_ACF_CURRENT)	Integer	U
2	X component of the first world coordinate point	Real	U
3	Y component of the first world coordinate point	Real	U
4	X component of the second world coordinate point	Real	U
5	Y component of the second world coordinate point	Real	U

To specify new output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 48. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Attribute control flag (GKS$K_ACF_SPECIFIED)	Integer	U
2	Line type ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL)	Integer	I
3	Line width scale factor ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL)	Integer	I
4	Polyline color ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL)	Integer	I
5	Polyline index	Integer	I
6	Line type index	Integer	U
7	Line width scale factor	Real	U
8	Polyline color index	Integer	I
9	X component of the first world coordinate point	Real	U

Component	Description	Data Type	Used or Ignored
10	Y component of the first world coordinate point	Real	U
11	X component of the second world coordinate point	Real	U
12	Y component of the second world coordinate point	Real	U

Locator-Class PET –13

The input data record size (INITIALIZE function argument *rec_size*) is 4. The input data record (INITIALIZE function argument *data_rec*) contains the component listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Segment identifier of the segment to be used for the cursor segment	Integer	U

9.3.3 Pick Class

This section lists the input data record information required for pick-class PETs.

Pick-Class PETs 1, 2, and 3

The input data record size (INITIALIZE function argument *rec_size*) is 4. The input data record (INITIALIZE function argument *data_rec*) contains the component listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Size of the pick aperture (prompt) in device coordinates	Real	U

9.3.4 String Class

This section lists the input data record information required for string-class PETs.

String-Class PET 1

The input data record size (INITIALIZE function argument *rec_size*) is 8. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Number of characters in the input buffer	Integer	U
2	Initial cursor position within the string, $1 \leq$ position $\leq$ string_length	Integer	I

9.3.5 Stroke Class

This section lists the input data record information required for stroke-class PETs.

Stroke-Class PET 1

The input data record size (INITIALIZE function argument *rec_size*) is 20. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Number of stroke points in the input buffer	Integer	U
2	Editing position expressed as a stroke point	Integer	I
3	X component of the world coordinate change vector	Real	U
4	Y component of the world coordinate change vector	Real	U
5	Time interval, in seconds	Real	I

Stroke-Class PET 3

This PET requires you to use one of two input data records. Which input data records you use depends on the value of the attribute control flag. If the attribute control flag is GKS$K_ACF_CURRENT, the prompt and echo displays the current set of output attributes. If the attribute control flag is GKS$K_ACF_SPECIFIED, the prompt and echo displays the output attributes specified in the input data record.

To specify the current output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 24. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Number of stroke points in the input buffer	Integer	U
2	Editing position expressed as a stroke point	Integer	I
3	X component of the world coordinate change vector	Real	U
4	Y component of the world coordinate change vector	Real	U
5	Time interval, in seconds	Real	I
6	Attribute control flag (GKS$K_ACF_CURRENT)	Integer	U

To specify new output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 52. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Number of stroke points in the input buffer	Integer	U
2	Editing position expressed as a stroke point	Integer	I
3	X component of the world coordinate change vector	Real	U
4	Y component of the world coordinate change vector	Real	U

Component	Description	Data Type	Used or Ignored
5	Time interval, in seconds	Real	I
6	Attribute control flag (GKS$K_ACF_CURRENT)	Integer	U
7	Polymarker type ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL)	Integer	I
8	Polymarker scale factor ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL)	Integer	I
9	Polymarker color ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL)	Integer	I
10	Polymarker index	Integer	I
11	Polymarker type index	Integer	U
12	Polymarker scale factor	Real	U
13	Polymarker color index	Integer	I

Stroke-Class PET 4

This PET requires you to use one of two input data records. Which input record you use depends on the value of the attribute control flag. If the attribute control flag is GKS$K_ACF_CURRENT, the prompt and echo displays the current set of output attributes. If the attribute control flag is GKS$K_ACF_SPECIFIED, the prompt and echo displays the output attributes specified in the input data record.

To specify the current output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 24. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Number of stroke points in the input buffer	Integer	U
2	Editing position expressed as a stroke point	Integer	I
3	X component of the world coordinate change vector	Real	U
4	Y component of the world coordinate change vector	Real	U
5	Time interval, in seconds	Real	I
6	Attribute control flag (GKS$K_ACF_CURRENT)	Integer	U

To specify new output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 52. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Number of stroke points in the input buffer	Integer	U
2	Editing position expressed as a stroke point	Integer	I
3	X component of the world coordinate change vector	Real	U
4	Y component of the world coordinate change vector	Real	U
5	Time interval, in seconds	Real	I
6	Attribute control flag (GKS$K_ACF_CURRENT)	Integer	U

Component	Description	Data Type	Used or Ignored
7	Line type ASF (GKS$K_ASF_BUNDLED or GKS$K_ ASF_INDIVIDUAL)	Integer	I
8	Line width scale factor ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL)	Integer	I
9	Polyline color ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL)	Integer	I
10	Polyline index	Integer	I
11	Line type index	Integer	U
12	Line width scale factor	Real	U
13	Polyline color index	Integer	I

9.3.6 Valuator Class

This section lists the input data record information required for valuator-class PETs.

Valuator-Class PETs –3, –2, –1, 1, 2, and 3

The input data record size (INITIALIZE function argument *rec_size*) is 8. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

Component	Description	Data Type	Used or Ignored
1	Low value of the numeric range	Real	U
2	High value of the numeric range	Real	U

9.4 Device Specifics Reference Manual for DEC GKS and DEC PHIGS

This section lists the changes in the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS* manual that occurred after the manual was printed in 1994.

9.4.1 GKS_PREDEF Program Example

The GKS_PREDEF program example on page 1-7 is incorrect. The output should be as follows:

```
Color Information for Digital LCP01 or LCG01 printer (15) devices

     Default color model: RGB (1)
     Number of color models: 4
     Color models:
            1 - RGB
            2 - CIE
            3 - HSV
            4 - HLS
     Number of colors: 8
     Color availability: Color
     Number of predefined colors: 8
     Predefined color table:
     ================================================
     Index               Red       Green        Blue
     ================================================
       0               1.000       1.000       1.000
```

```
  1              0.000       1.000       0.000
  2              1.000       0.000       0.000
  3              0.000       0.000       1.000
  4              0.000       1.000       1.000
  5              1.000       0.000       1.000
  6              1.000       1.000       0.000
  7              0.000       0.000       0.000
==============================================
```

9.5 DEC GKS GKS3D$ Binding Reference Manual

The input data record information described in Section 9.3 is also true for the GKS3D$ binding, with the following exceptions:

- The GKS3D$ include files are GKS3D$DEFS.*.
- All the constants begin with GKS3D$ instead of GKS$ (for example, GKS3D$K_ACF_ CURRENT).
- The INITIALIZE function parameter *rec_size* is called *data_len* in the GKS3D$ binding.

9.5.1 INQUIRE STROKE DEVICE STATE

The syntax lists *stroke_x* and *stroke_y* as separate arguments. In fact, these values are included in a single argument, *wc_stroke2*.

9.5.2 INQUIRE STROKE DEVICE STATE 3

There are a few errors in the description of the INQUIRE STROKE DEVICE STATE 3 function. In the syntax section, the argument should be *ret_pts*, not *ret_points*.

The syntax also lists *stroke_x*, *stroke_y*, and *stroke_z* as separate arguments. In fact, these values are included in a single argument, *wc_stroke3*. Further, in the description of this argument, it should read "The points are ordered . . . , for each of the *ret_pts*.

10 Copyright

© Copyright 2017 VMS Software, Inc., Bolton Massachusetts, USA
Confidential computer software. Valid license from VSI required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendors standard commercial license.

The information contained herein is subject to change without notice. The only warranties for VSI products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. VSI shall not be liable for technical or editorial errors, or omissions contained herein.

HPE, HPE Integrity, and HPE Alpha are trademarks or registered trademarks of Hewlett Packard Enterprise.

Apple is a trademark of Apple Computer, Inc., registered in the U.S. and other countries.

Intel, Intel Itanium and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Motif and OSF/1 are registered trademarks of The OpenGroup.

PostScript is a registered trademark of Adobe Systems Incorporated.

TEKTRONIX and Tek are registered trademarks of Tektronix, Inc.

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

The VSI OpenVMS documentation set is available on CD.