INTERNATIONAL STANDARD

ISO/IEC 9593-3

First edition
1990-04-15

Information technology — Computer graphics — Programmer's Hierarchical Interactive Graphics System (PHIGS) language bindings —

Part 3 : Ada

Technologies de l'information — Infographie — Interfaces langage avec PHIGS — Partie 3 : Ada

Reference number
ISO/IEC 9593-3 : 1990 (E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) together form a system for worldwide standardization as a whole. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for approval before their acceptance as International Standards. They are approved in accordance with procedures requiring at least 75 % approval by the national bodies voting.

International Standard ISO/IEC 9593-3 was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*.

Introduction

ISO/IEC 9592 is specified in a language independent manner and needs to be embedded in language dependent layers (language bindings) for use with particular programming languages.

The purpose of this part of ISO/IEC 9593 is to define a standard binding of PHIGS to the Ada computer programming language.

This page intentionally left blank

Information technology — Computer graphics — Programmer's Hierarchical Interactive Graphics System (PHIGS) language bindings —

Part 3 :
Ada

1 Scope

ISO/IEC 9592 specifies a language independent nucleus of a graphics system. For integration into a programming language, PHIGS is embedded in a language dependent layer obeying the particular conventions of that language. This part of ISO/IEC 9593 specifies such a language dependent layer for the Ada computer programming language.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9593. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9593 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 8652 : 1987, *Programming languages - Ada (Endorsement of ANSI Standard 1815A-1983).*

ISO/IEC 9592-1 : 1989, *Information processing systems - Computer graphics - Programmer's Hierarchical Interactive Graphics System (PHIGS) - Part 1: Functional description.*

3 Principles

The PHIGS Binding to Ada is intended to be implementation independent except as it can be adapted by the PHIGS_CONFIGURATION package. The PHIGS Binding to Ada makes no assumptions concerning the availability of implementation dependent facilities defined by the Ada language. It does, however, limit the use of multi-tasking as described in 3.2.5. The Ada compiler shall be able to support the number of declarations contained in this PHIGS Binding to Ada.

This binding does not make any assumptions regarding the machine representation of the predefined Ada numeric types.

This binding assumes that the application programmer will supply an error file name, archive file names, and connection identifiers that are in an acceptable format for the Ada implementation.

This binding makes no assumptions regarding the format of a string specifying an error file name, archive file names, or connection identifiers for devices or metafiles.

3.1 Conformance

This binding incorporates the rules of conformance defined in the PHIGS Standard (ISO/IEC 9592) for PHIGS implementations, with these additional requirements specifically defined for Ada implementations of PHIGS.

The following criteria are established for determining conformance or non-conformance of an implementation to this binding:

- The semantics of an implementation shall be those stated in ISO/IEC 9592 as modified or extended for Ada as stated in this part of ISO/IEC 9593.
- The package corresponding to PHIGS shall be an available Ada library unit, with all names as specified by this part of ISO/IEC 9593.

3.2 Implications of the Language

3.2.1 Functional Mapping

The functions of PHIGS are all mapped to Ada procedures. The mapping utilizes a one-to-one correspondence between the PHIGS functions and Ada procedures except for the PHIGS function INQUIRE TEXT EXTENT. This function is mapped to two overloaded Ada functions each named INQ_TEXT_EXTENT one which

supports modelling text and one which supports annotation text. Certain functions required by the binding but not defined by PHIGS are mapped to Ada functions and procedures.

3.2.2 Implementation and Host Dependencies

There are a number of implementation and host dependencies associated with the Ada compiler and runtime system used. These will affect the portability of application programs and their use of PHIGS. The application programmer should follow accepted practices for ensuring portability of Ada programs to avoid introducing problems when rehosting the application on another system. Implementation dependencies include runtime storage management and processor management.

3.2.3 Error Handling

The inquiry functions utilize error indicator parameters for the error returns, and do not raise Ada exceptions. The application program shall ensure that these error indicators are checked before attempting to access other parameters, since Ada implementations are not required to raise an exception if an undefined value is accessed.

The error handling requirements of PHIGS can be summarized as follows:

1. By default, a procedure named ERROR_HANDLING will be provided that simply reports the error by calling ERROR_LOGGING. This is called from the PHIGS function that detects the error.

2. The ERROR_HANDLING procedure may be replaced by one defined by the user.

The procedure ERROR_HANDLING is defined as a library subprogram:

```
with PHIGS_TYPES;
use PHIGS_TYPES;
procedure ERROR_HANDLING
  (ERROR_INDICATOR  : in ERROR_NUMBER;
   PHIGS_FUNCTION   : in STRING;
   ERROR_FILE       : in FILE_ID := DEFAULT_ERROR_FILE);

-- The procedure ERROR_HANDLING is defined as a library subprogram,
-- and is not declared within package PHIGS.
```

This binding defines two different bodies for this subprogram; each shall be supplied by the implementation. The default body is the one required by PHIGS semantics. It simply calls ERROR_LOGGING and returns. The second body calls ERROR_LOGGING and then raises the exception PHIGS_ERROR. The PHIGS function shall be written so as not to handle PHIGS_ERROR (this is a requirement of the implementation). Thus, by Ada rules, the exception will be propagated back to the application program that called the PHIGS function in which the error was detected.

The means by which the user replaces the default body by either the exception-raising version or another one of his or her choosing is dependent upon the Ada library manager. Some implementations support multiple versions of a body with a single specification or otherwise allow hierarchical libraries with the sharing of common units. In other implementations, it may be necessary to duplicate the PHIGS library for each version of ERROR_HANDLING.

3.2.4 Data mapping

The simple and compound data types of PHIGS are bound to a variety of Ada scalar and compound types. Constraints on permitted values are reflected where possible in the type definitions. The general correspondence between the PHIGS data types and Ada binding data types is summarized below:

- PHIGS integer types (I) are mapped to Ada integer types.
- PHIGS real types (R) are mapped to Ada floating-point types.
- PHIGS string types (S) are mapped to the predefined Ada type STRING, or to a type providing for variable length strings.
- PHIGS point types (P2, P3) are mapped to Ada record types.
- PHIGS vector types (V2, V3) are mapped to Ada record types.
- PHIGS enumeration types (E) are mapped to Ada enumeration types.
- PHIGS name types (NM) are mapped to an Ada integer type. The PHIGS name set composite type SET(NM) is mapped to an Ada private type. A set of subprograms for operating on objects of this private type is explicitly defined by this binding.
- PHIGS filter types (FR) are mapped to an Ada record type.
- PHIGS pick path item type (PP) is mapped to an Ada record type. PHIGS pick paths are mapped to an Ada array type.
- PHIGS element reference type (ER) is mapped to an Ada record type.
- PHIGS half-space types (HS2, HS3) are mapped to Ada record types.
- PHIGS font/precision pair type (FP) is mapped to an Ada record type.
- PHIGS structure element type (SE) is mapped to an Ada record type.
- PHIGS posted structure type (PS) is mapped to an Ada record type.
- PHIGS bounding range types (B) are mapped to Ada record types.
- PHIGS colour specification type (CLR) is mapped to an Ada record type.
- PHIGS chromaticity coefficient type (CC) is mapped to an Ada record type.
- PHIGS connection identifier type (C) is mapped to the Ada STRING type.
- PHIGS file types (F) are mapped to Ada STRING types.
- PHIGS workstation type (W) is mapped to an Ada integer type.
- PHIGS modelling clipping volume type (MCV) is not used by the binding. Implementations can map this type to an implementation specific private type.
- PHIGS generalized drawing primitive identifier types (G2, G3) are mapped to Ada integer types.
- PHIGS generalized structure element identifier type (GS) is mapped to an Ada integer type.
- PHIGS archive file identifier type (AI) is mapped to an Ada integer type.
- PHIGS pick identifier type (PI) is mapped to an Ada integer type.
- PHIGS escape identifier type (EI) is mapped to an Ada integer type.
- PHIGS function name type (FN) is mapped to an Ada type providing for variable length strings.

- PHIG workstation identifier type (WI) is mapped to an Ada integer type.
- PHIGS tuples are mapped to Ada record types.
- PHIGS matrices are mapped to Ada array types.
- PHIGS arrays are mapped to either an unconstrained Ada array type, or to a record type providing for variable length arrays.
- PHIGS lists are mapped to an Ada private type declared in the generic PHIGS_LIST_UTILITIES package.
- PHIGS set types are mapped to Ada private types with generic accessing functions provided as necessary.
- PHIGS data records are mapped to Ada private types. In some cases, a set of subprograms for operating on the data records is explicitly defined by this binding. This is because the content and structure of the data record is implementation dependent. An implementation of PHIGS may provide other subprograms for manipulating implementation dependent data records.

Additional types used by the binding are declared as needed in a manner compatible with the PHIGS types.

3.2.5 Multi-tasking

The Ada language definition provides explicit support for concurrency. The Ada tasking model includes facilities for declaring and allocating tasks, and operations allowing intertask communication and synchronization.

The PHIGS standard, and hence this binding standard, neither requires nor prohibits an implementation from protecting against problems which could arise from asynchronous access to the PHIGS data structures from concurrent tasks. Implementors of PHIGS should provide information in the user's documentation regarding whether protection against such problems is implemented.

Annex D contains guidelines for implementors who want to support multi-tasking application programs. This annex does not form an integral part of the binding standard, but provides additional information.

3.2.6 Packaging

The PHIGS standard defines all of its graphic functionality as a cohesive whole. An implementation of PHIGS shall implement the entire functionality of PHIGS. To support this concept, this binding defines a single Ada package which corresponds to all of the PHIGS functionality. This package is named

```
package PHIGS is ... end PHIGS;
```

Associated with this package is a data type package which provides the type declarations as defined in 4.2 and the exception defined in 4.3. This package is

```
package PHIGS_TYPES is ... end PHIGS_TYPES;
```

A minimal program referencing PHIGS is shown below.

```
with PHIGS,
     PHIGS_TYPES;

procedure APPLICATION is
begin
```

```
    null;
end APPLICATION;
```

Several additional Ada packages are defined in this binding. These packages are

- generic package PHIGS_COORDINATE_SYSTEM
- generic package PHIGS_LIST_UTILITIES
- package PHIGS_NAME_SET_FACILITY
- package PHIGS_CONFIGURATION

These packages support the declaration types in the PHIGS_TYPES package described above. PHIGS_COORDINATE_SYSTEM is a generic package which defines an assortment of types supporting each of the PHIGS coordinate systems. PHIGS_LIST_UTILITIES is also a generic package which provides type declarations and operations for list types which correspond to the PHIGS list types. The PHIGS_NAME_SET_FACILITY package defines private types for name sets and provides functions for building and manipulating name sets. The PHIGS_CONFIGURATION package sets implementation dependent limit values for various types defined in this binding. The PHIGS_CONFIGURATION package also provides for possible application modification of these limits.

3.2.7 Application Program Environment

An application program utilizing an Ada implementation of PHIGS will need to be aware of the environment in which both PHIGS and the application program(s) reside.

One aspect of the environment is the Ada program library. The Ada language requires that the application program have access to the program library in which the PHIGS software resides. The Ada standard ISO 8652 does not specify whether there is a single library or multiple libraries, or how access to the libraries is granted or managed. The user's documentation for the PHIGS implementation should specify where the PHIGS library exists in the system, and how access to the library is acquired.

Input/Output interfaces are also implementation-dependent, and are required to be described in the user's documentation. Besides the obvious graphics device interface information, interfaces to the file system shall be included in the documentation. Specifically, this includes the interface to the PHIGS error file, archive files, and metafile storage.

Subclause 3.2.2 discusses implications about the application program environment which may also apply to application programs.

3.2.8 Registration[1]

PHIGS reserves certain value ranges for registration as graphical items. The registered graphical items will be bound to Ada (and other programming languages). The registered item bindings will be consistent with the binding presented in this document.

---

1) For the purpose of this International Standard and according to the rules for the designation and operation of registration authorities in the ISO/IEC JTC 1 procedures, the ISO and IEC Councils have designated the National Institute of Standards and Technology (National Computer Systems Laboratory) A-266 Technology Building, Gaithersburg, MD 20899, USA, to act as registration authority.

4 Tables

4.1 Abbreviations used in procedure names

ASF	aspect source flag
CHAR	character
ESC	escape
GDP	generalized drawing primitive
GSE	generalized structure element
HLHSR	hidden line/hidden surface removal
INQ	inquire
PHIGS	Programmer's Hierarchical Interactive Graphics System
U	unregistered
WS	workstation

4.1.1 List of procedures using the abbreviations

ASF	SET_INDIVIDUAL_ASF
CHAR	SET_ANNOTATION_TEXT_CHAR_HEIGHT SET_ANNOTATION_TEXT_CHAR_UP_VECTOR SET_CHAR_EXPANSION_FACTOR SET_CHAR_HEIGHT SET_CHAR_SPACING SET_CHAR_UP_VECTOR
ESC	PHIGS_ESCAPE.GENERALIZED_ESC PHIGS_ESC_<name of the escape procedure>.ESC PHIGS_UESC_<name of the escape procedure>.ESC
GDP	INQ_GDP (2D) INQ_GDP (3D) INQ_LIST_OF_AVAILABLE_GDP (2D) INQ_LIST_OF_AVAILABLE_GDP (3D) PHIGS_GDP.GENERALIZED_GDP PHIGS_GDP_<name of the GDP procedure>.GDP PHIGS_UGDP_<name of the GDP procedure>.GDP
GSE	INQ_GSE_FACILITIES INQ_LIST_OF_AVAILABLE_GSE PHIGS_GSE.GENERALIZED_GSE PHIGS_GSE_<name of the GSE procedure>.GSE PHIGS_UGSE_<name of the GSE procedure>.GSE
HLHSR	INQ_HLHSR_FACILITIES INQ_HLHSR_MODE SET_HLHSR_IDENTIFIER SET_HLHSR_MODE
INQ	INQ_ALL_CONFLICTING_STRUCTURES

INQ_ANNOTATION_FACILITIES
INQ_ARCHIVE_FILES
INQ_ARCHIVE_STATE_VALUE
INQ_CHOICE_DEVICE_STATE (2D)
INQ_CHOICE_DEVICE_STATE (3D)
INQ_COLOUR_FACILITIES
INQ_COLOUR_MODEL
INQ_COLOUR_MODEL_FACILITIES
INQ_COLOUR_REPRESENTATION
INQ_CONFLICT_RESOLUTION
INQ_CONFLICTING_STRUCTURES_IN_NETWORK
INQ_CURRENT_ELEMENT_CONTENT
INQ_CURRENT_ELEMENT_TYPE_AND_SIZE
INQ_DEFAULT_CHOICE_DEVICE_DATA (2D)
INQ_DEFAULT_CHOICE_DEVICE_DATA (3D)
INQ_DEFAULT_DISPLAY_UPDATE_STATE
INQ_DEFAULT_LOCATOR_DEVICE_DATA (2D)
INQ_DEFAULT_LOCATOR_DEVICE_DATA (3D)
INQ_DEFAULT_PICK_DEVICE_DATA (2D)
INQ_DEFAULT_PICK_DEVICE_DATA (3D)
INQ_DEFAULT_STRING_DEVICE_DATA (2D)
INQ_DEFAULT_STRING_DEVICE_DATA (3D)
INQ_DEFAULT_STROKE_DEVICE_DATA (2D)
INQ_DEFAULT_STROKE_DEVICE_DATA (3D)
INQ_DEFAULT_VALUATOR_DEVICE_DATA (2D)
INQ_DEFAULT_VALUATOR_DEVICE_DATA (3D)
INQ_DISPLAY_SPACE_SIZE (2D)
INQ_DISPLAY_SPACE_SIZE (3D)
INQ_DISPLAY_UPDATE_STATE
INQ_DYNAMICS_OF_STRUCTURES
INQ_DYNAMICS_OF_WS_ATTRIBUTES
INQ_EDGE_FACILITIES
INQ_EDGE_REPRESENTATION
INQ_EDIT_MODE
INQ_ELEMENT_CONTENT
INQ_ELEMENT_POINTER
INQ_ELEMENT_TYPE_AND_SIZE
INQ_ERROR_HANDLING_MODE
INQ_GDP (2D)
INQ_GDP (3D)
INQ_GSE_FACILITIES
INQ_HIGHLIGHTING_FILTER
INQ_HLHSR_FACILITIES
INQ_HLHSR_MODE
INQ_INPUT_QUEUE_OVERFLOW
INQ_INTERIOR_FACILITIES
INQ_INTERIOR_REPRESENTATION
INQ_INVISIBILITY_FILTER
INQ_LIST_OF_AVAILABLE_GDP (2D)
INQ_LIST_OF_AVAILABLE_GDP (3D)
INQ_LIST_OF_AVAILABLE_GSE
INQ_LIST_OF_AVAILABLE_WS_TYPES
INQ_LIST_OF_COLOUR_INDICES
INQ_LIST_OF_EDGE_INDICES
INQ_LIST_OF_INTERIOR_INDICES
INQ_LIST_OF_PATTERN_INDICES
INQ_LIST_OF_POLYLINE_INDICES
INQ_LIST_OF_POLYMARKER_INDICES
INQ_LIST_OF_TEXT_INDICES
INQ_LIST_OF_VIEW_INDICES
INQ_LOCATOR_DEVICE_STATE (2D)
INQ_LOCATOR_DEVICE_STATE (3D)
INQ_MODELLING_CLIPPING_FACILITIES
INQ_MORE_SIMULTANEOUS_EVENTS
INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES
INQ_NUMBER_OF_DISPLAY_PRIORITIES_SUPPORTED
INQ_OPEN_STRUCTURE
INQ_PATHS_TO_ANCESTORS
INQ_PATHS_TO_DESCENDANTS
INQ_PATTERN_FACILITIES
INQ_PATTERN_REPRESENTATION

	INQ_PHIGS_FACILITIES
	INQ_PICK_DEVICE_STATE (2D)
	INQ_PICK_DEVICE_STATE (3D)
	INQ_POLYLINE_FACILITIES
	INQ_POLYLINE_REPRESENTATION
	INQ_POLYMARKER_FACILITIES
	INQ_POLYMARKER_REPRESENTATION
	INQ_POSTED_STRUCTURES
	INQ_PREDEFINED_COLOUR_REPRESENTATION
	INQ_PREDEFINED_EDGE_REPRESENTATION
	INQ_PREDEFINED_INTERIOR_REPRESENTATION
	INQ_PREDEFINED_PATTERN_REPRESENTATION
	INQ_PREDEFINED_POLYLINE_REPRESENTATION
	INQ_PREDEFINED_POLYMARKER_REPRESENTATION
	INQ_PREDEFINED_TEXT_REPRESENTATION
	INQ_PREDEFINED_VIEW_REPRESENTATION
	INQ_SET_OF_OPEN_WS
	INQ_SET_OF_WS_TO_WHICH_POSTED
	INQ_STRING_DEVICE_STATE (2D)
	INQ_STRING_DEVICE_STATE (3D)
	INQ_STROKE_DEVICE_STATE (2D)
	INQ_STROKE_DEVICE_STATE (3D)
	INQ_STRUCTURE_IDENTIFIERS
	INQ_STRUCTURE_STATE_VALUE
	INQ_STRUCTURE_STATUS
	INQ_SYSTEM_STATE_VALUE
	INQ_TEXT_EXTENT (text)
	INQ_TEXT_EXTENT (annotation text)
	INQ_TEXT_FACILITIES
	INQ_TEXT_REPRESENTATION
	INQ_VALUATOR_DEVICE_STATE (2D)
	INQ_VALUATOR_DEVICE_STATE (3D)
	INQ_VIEW_FACILITIES
	INQ_VIEW_REPRESENTATION
	INQ_WS_CATEGORY
	INQ_WS_CLASSIFICATION
	INQ_WS_CONNECTION_AND_TYPE
	INQ_WS_STATE_TABLE_LENGTHS
	INQ_WS_STATE_VALUE
	INQ_WS_TRANSFORMATION (2D)
	INQ_WS_TRANSFORMATION (3D)
PHIGS	CLOSE_PHIGS
	EMERGENCY_CLOSE_PHIGS
	INQ_PHIGS_FACILITIES
	OPEN_PHIGS
U	PHIGS_UESC_<name of the escape procedure>.ESC
	PHIGS_UGDP_<name of the GDP procedure>.GDP
	PHIGS_UGSE_<name of the GSE procedure>.GSE
WS	CLOSE_WS
	INQ_DYNAMICS_OF_WS_ATTRIBUTES
	INQ_LIST_OF_AVAILABLE_WS_TYPES
	INQ_SET_OF_OPEN_WS
	INQ_SET_OF_WS_TO_WHICH_POSTED
	INQ_WS_CATEGORY
	INQ_WS_CLASSIFICATION
	INQ_WS_CONNECTION_AND_TYPE
	INQ_WS_STATE_TABLE_LENGTHS
	INQ_WS_STATE_VALUE
	INQ_WS_TRANSFORMATION (2D)
	INQ_WS_TRANSFORMATION (3D)
	OPEN_WS
	SET_WS_VIEWPORT (2D)
	SET_WS_VIEWPORT (3D)
	SET_WS_WINDOW (2D)
	SET_WS_WINDOW (3D)
	UPDATE_WS

4.1.2 Alphabetical by bound name

This subclause alphabetically lists the Ada procedures corresponding to each PHIGS function. Instances in which the binding of a name corresponds to more than one PHIGS function indicates the use of Ada overloading.

ADD_NAMES_TO_SET	add names to set
ANNOTATION_TEXT_RELATIVE	annotation text relative
ANNOTATION_TEXT_RELATIVE	annotation text relative 3
APPLICATION_DATA	application data
ARCHIVE_ALL_STRUCTURES	archive all structures
ARCHIVE_STRUCTURE_NETWORKS	archive structure networks
ARCHIVE_STRUCTURES	archive structures
AWAIT_EVENT	await event
BUILD_TRANSFORMATION_MATRIX	build transformation matrix
BUILD_TRANSFORMATION_MATRIX	build transformation matrix 3
CELL_ARRAY	cell array
CELL_ARRAY	cell array 3
CHANGE_STRUCTURE_IDENTIFIER	change structure identifier
CHANGE_STRUCTURE_IDENTIFIER_AND_REFERENCES	change structure identifier and references
CHANGE_STRUCTURE_REFERENCES	change structure references
CLOSE_ARCHIVE_FILE	close archive file
CLOSE_PHIGS	close PHIGS
CLOSE_STRUCTURE	close structure
CLOSE_WS	close workstation
COMPOSE_MATRIX	compose matrix
COMPOSE_MATRIX	compose matrix 3
COMPOSE_TRANSFORMATION_MATRIX	compose transformation matrix
COMPOSE_TRANSFORMATION_MATRIX	compose transformation matrix 3
COPY_ALL_ELEMENTS_FROM_STRUCTURE	copy all elements from structure
DELETE_ALL_STRUCTURES	delete all structures
DELETE_ALL_STRUCTURES_FROM_ARCHIVE	delete all structures from archive
DELETE_ELEMENT	delete element
DELETE_ELEMENT_RANGE	delete element range
DELETE_ELEMENTS_BETWEEN_LABELS	delete elements between labels
DELETE_STRUCTURE	delete structure
DELETE_STRUCTURE_NETWORK	delete structure network
DELETE_STRUCTURE_NETWORKS_FROM_ARCHIVE	delete structure networks from archive
DELETE_STRUCTURES_FROM_ARCHIVE	delete structures from archive
ELEMENT_SEARCH	element search
EMERGENCY_CLOSE_PHIGS	emergency close PHIGS
EMPTY_STRUCTURE	empty structure
ERROR_HANDLING	error handling
ERROR_LOGGING	error logging
ESCAPE	escape
EVALUATE_VIEW_MAPPING_MATRIX	evaluate view mapping matrix
EVALUATE_VIEW_MAPPING_MATRIX	evaluate view mapping matrix 3
EVALUATE_VIEW_ORIENTATION_MATRIX	evaluate view orientation matrix
EVALUATE_VIEW_ORIENTATION_MATRIX	evaluate view orientation matrix 3
EXECUTE_STRUCTURE	execute structure
FILL_AREA	fill area
FILL_AREA	fill area 3
FILL_AREA_SET	fill area set
FILL_AREA_SET	fill area set 3
FLUSH_DEVICE_EVENTS	flush device events
GENERALIZED_GDP	generalized drawing primitive
GENERALIZED_GDP	generalized drawing primitive 3
GENERALIZED_GSE	generalized structure element
GET_CHOICE	get choice
GET_ITEM_TYPE_FROM_METAFILE	get item type from metafile
GET_LOCATOR	get locator
GET_LOCATOR	get locator 3
GET_PICK	get pick
GET_STRING	get string
GET_STROKE	get stroke

GET_STROKE	get stroke 3
GET_VALUATOR	get valuator
INCREMENTAL_SPATIAL_SEARCH	incremental spatial search
INCREMENTAL_SPATIAL_SEARCH	incremental spatial search 3
INITIALIZE_CHOICE	initialize choice
INITIALIZE_CHOICE	initialize choice 3
INITIALIZE_LOCATOR	initialize locator
INITIALIZE_LOCATOR	initialize locator 3
INITIALIZE_PICK	initialize pick
INITIALIZE_PICK	initialize pick 3
INITIALIZE_STRING	initialize string
INITIALIZE_STRING	initialize string 3
INITIALIZE_STROKE	initialize stroke
INITIALIZE_STROKE	initialize stroke 3
INITIALIZE_VALUATOR	initialize valuator
INITIALIZE_VALUATOR	initialize valuator 3
INQ_ALL_CONFLICTING_STRUCTURES	inquire all conflicting structures
INQ_ANNOTATION_FACILITIES	inquire annotation facilities
INQ_ARCHIVE_FILES	inquire archive files
INQ_ARCHIVE_STATE_VALUE	inquire archive state value
INQ_CHOICE_DEVICE_STATE	inquire choice device state
INQ_CHOICE_DEVICE_STATE	inquire choice device state 3
INQ_COLOUR_FACILITIES	inquire colour facilities
INQ_COLOUR_MODEL	inquire colour model
INQ_COLOUR_MODEL_FACILITIES	inquire colour model facilities
INQ_COLOUR_REPRESENTATION	inquire colour representation
INQ_CONFLICT_RESOLUTION	inquire conflict resolution
INQ_CONFLICTING_STRUCTURES_IN_NETWORK	inquire conflicting structures in network
INQ_CURRENT_ELEMENT_CONTENT	inquire current element content
INQ_CURRENT_ELEMENT_TYPE_AND_SIZE	inquire current element type and size
INQ_DEFAULT_CHOICE_DEVICE_DATA	inquire default choice device data
INQ_DEFAULT_CHOICE_DEVICE_DATA	inquire default choice device data 3
INQ_DEFAULT_DISPLAY_UPDATE_STATE	inquire default display update state
INQ_DEFAULT_LOCATOR_DEVICE_DATA	inquire default locator device data
INQ_DEFAULT_LOCATOR_DEVICE_DATA	inquire default locator device data 3
INQ_DEFAULT_PICK_DEVICE_DATA	inquire default pick device data
INQ_DEFAULT_PICK_DEVICE_DATA	inquire default pick device data 3
INQ_DEFAULT_STRING_DEVICE_DATA	inquire default string device data
INQ_DEFAULT_STRING_DEVICE_DATA	inquire default string device data 3
INQ_DEFAULT_STROKE_DEVICE_DATA	inquire default stroke device data
INQ_DEFAULT_STROKE_DEVICE_DATA	inquire default stroke device data 3
INQ_DEFAULT_VALUATOR_DEVICE_DATA	inquire default valuator device data
INQ_DEFAULT_VALUATOR_DEVICE_DATA	inquire default valuator device data 3
INQ_DISPLAY_SPACE_SIZE	inquire display space size
INQ_DISPLAY_SPACE_SIZE	inquire display space size 3
INQ_DISPLAY_UPDATE_STATE	inquire display update state
INQ_DYNAMICS_OF_STRUCTURES	inquire dynamics of structures
INQ_DYNAMICS_OF_WS_ATTRIBUTES	inquire dynamics of workstation attributes
INQ_EDGE_FACILITIES	inquire edge facilities
INQ_EDGE_REPRESENTATION	inquire edge representation
INQ_EDIT_MODE	inquire edit mode
INQ_ELEMENT_CONTENT	inquire element content
INQ_ELEMENT_POINTER	inquire element pointer
INQ_ELEMENT_TYPE_AND_SIZE	inquire element type and size
INQ_ERROR_HANDLING_MODE	inquire error handling mode
INQ_GDP	inquire generalized drawing primitives
INQ_GDP	inquire generalized drawing primitives 3
INQ_GSE_FACILITIES	inquire generalized structure element facilities
INQ_HIGHLIGHTING_FILTER	inquire highlighting filter
INQ_HLHSR_FACILITIES	inquire hlhsr facilities
INQ_HLHSR_MODE	inquire hlhsr mode
INQ_INPUT_QUEUE_OVERFLOW	inquire input queue overflow
INQ_INTERIOR_FACILITIES	inquire interior facilities
INQ_INTERIOR_REPRESENTATION	inquire interior representation
INQ_INVISIBILITY_FILTER	inquire invisibility filter
INQ_LIST_OF_AVAILABLE_GDP	inquire list of available generalized drawing primitives
INQ_LIST_OF_AVAILABLE_GDP	inquire list of available generalized drawing primitives 3
INQ_LIST_OF_AVAILABLE_GSE	inquire list of available generalized structure elements
INQ_LIST_OF_AVAILABLE_WS_TYPES	inquire list of available workstation types
INQ_LIST_OF_COLOUR_INDICES	inquire list of colour indices
INQ_LIST_OF_EDGE_INDICES	inquire list of edge indices

INQ_LIST_OF_INTERIOR_INDICES	inquire list of interior indices
INQ_LIST_OF_PATTERN_INDICES	inquire list of pattern indices
INQ_LIST_OF_POLYLINE_INDICES	inquire list of polyline indices
INQ_LIST_OF_POLYMARKER_INDICES	inquire list of polymarker indices
INQ_LIST_OF_TEXT_INDICES	inquire list of text indices
INQ_LIST_OF_VIEW_INDICES	inquire list of view indices
INQ_LOCATOR_DEVICE_STATE	inquire locator device state
INQ_LOCATOR_DEVICE_STATE	inquire locator device state 3
INQ_MODELLING_CLIPPING_FACILITIES	inquire modelling clipping facilities
INQ_MORE_SIMULTANEOUS_EVENTS	inquire more simultaneous events
INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES	inquire number of available logical input devices
INQ_NUMBER_OF_DISPLAY_PRIORITIES_SUPPORTED	inquire number of structure priorities supported
INQ_OPEN_STRUCTURE	inquire open structure
INQ_PATHS_TO_ANCESTORS	inquire paths to ancestors
INQ_PATHS_TO_DESCENDANTS	inquire paths to descendants
INQ_PATTERN_FACILITIES	inquire pattern facilities
INQ_PATTERN_REPRESENTATION	inquire pattern representation
INQ_PHIGS_FACILITIES	inquire phigs facilities
INQ_PICK_DEVICE_STATE	inquire pick device state
INQ_PICK_DEVICE_STATE	inquire pick device state 3
INQ_POLYLINE_FACILITIES	inquire polyline facilities
INQ_POLYLINE_REPRESENTATION	inquire polyline representation
INQ_POLYMARKER_FACILITIES	inquire polymarker facilities
INQ_POLYMARKER_REPRESENTATION	inquire polymarker representation
INQ_POSTED_STRUCTURES	inquire posted structures
INQ_PREDEFINED_COLOUR_REPRESENTATION	inquire predefined colour representation
INQ_PREDEFINED_EDGE_REPRESENTATION	inquire predefined edge representation
INQ_PREDEFINED_INTERIOR_REPRESENTATION	inquire predefined interior representation
INQ_PREDEFINED_PATTERN_REPRESENTATION	inquire predefined pattern representation
INQ_PREDEFINED_POLYLINE_REPRESENTATION	inquire predefined polyline representation
INQ_PREDEFINED_POLYMARKER_REPRESENTATION	inquire predefined polymarker representation
INQ_PREDEFINED_TEXT_REPRESENTATION	inquire predefined text representation
INQ_PREDEFINED_VIEW_REPRESENTATION	inquire predefined view representation
INQ_SET_OF_OPEN_WS	inquire set of open workstations
INQ_SET_OF_WS_TO_WHICH_POSTED	inquire set of workstations to which posted
INQ_STRING_DEVICE_STATE	inquire string device state
INQ_STRING_DEVICE_STATE	inquire string device state 3
INQ_STROKE_DEVICE_STATE	inquire stroke device state
INQ_STROKE_DEVICE_STATE	inquire stroke device state 3
INQ_STRUCTURE_IDENTIFIERS	inquire structure identifiers
INQ_STRUCTURE_STATE_VALUE	inquire structure state value
INQ_STRUCTURE_STATUS	inquire structure status
INQ_SYSTEM_STATE_VALUE	inquire system state value
INQ_TEXT_EXTENT (text)	inquire text extent
INQ_TEXT_EXTENT (annotation text)	inquire text extent
INQ_TEXT_FACILITIES	inquire text facilities
INQ_TEXT_REPRESENTATION	inquire text representation
INQ_VALUATOR_DEVICE_STATE	inquire valuator device state
INQ_VALUATOR_DEVICE_STATE	inquire valuator device state 3
INQ_VIEW_FACILITIES	inquire view facilities
INQ_VIEW_REPRESENTATION	inquire view representation
INQ_WS_CATEGORY	inquire workstation category
INQ_WS_CLASSIFICATION	inquire workstation classification
INQ_WS_CONNECTION_AND_TYPE	inquire workstation connection and type
INQ_WS_STATE_TABLE_LENGTHS	inquire workstation state table lengths
INQ_WS_STATE_VALUE	inquire workstation state value
INQ_WS_TRANSFORMATION	inquire workstation transformation
INQ_WS_TRANSFORMATION	inquire workstation transformation 3
INTERPRET_ITEM	interpret item
LABEL	label
MESSAGE	message
OFFSET_ELEMENT_POINTER	offset element pointer
OPEN_ARCHIVE_FILE	open archive fill
OPEN_PHIGS	open PHIGS
OPEN_STRUCTURE	open structure
OPEN_WS	open workstation
POLYLINE	polyline
POLYLINE	polyline 3
POLYMARKER	polymarker
POLYMARKER	polymarker 3
POST_STRUCTURE	post structure

READ_ITEM_FROM_METAFILE	read item from metafile
REDRAW_ALL_STRUCTURES	redraw all structures
REMOVE_NAMES_FROM_SET	remove names from set
REQUEST_CHOICE	request choice
REQUEST_LOCATOR	request locator
REQUEST_LOCATOR	request locator 3
REQUEST_PICK	request pick
REQUEST_STRING	request string
REQUEST_STROKE	request stroke
REQUEST_STROKE	request stroke 3
REQUEST_VALUATOR	request valuator
RESTORE_MODELLING_CLIPPING_VOLUME	restore modelling clipping volume
RETRIEVE_ALL_STRUCTURES	retrieve all structures
RETRIEVE_PATHS_TO_ANCESTORS	retrieve paths to ancestors
RETRIEVE_PATHS_TO_DESCENDANTS	retrieve paths to descendants
RETRIEVE_STRUCTURE_IDENTIFIERS	retrieve structure identifiers
RETRIEVE_STRUCTURE_NETWORKS	retrieve structure networks
RETRIEVE_STRUCTURES	retrieve structures
ROTATE	rotate
ROTATE_X	rotate x
ROTATE_Y	rotate y
ROTATE_Z	rotate z
SAMPLE_CHOICE	sample choice
SAMPLE_LOCATOR	sample locator
SAMPLE_LOCATOR	sample locator 3
SAMPLE_PICK	sample pick
SAMPLE_STRING	sample string
SAMPLE_STROKE	sample stroke
SAMPLE_STROKE	sample stroke 3
SAMPLE_VALUATOR	sample valuator
SCALE	scale
SCALE	scale 3
SET_ANNOTATION_STYLE	set annotation style
SET_ANNOTATION_TEXT_ALIGNMENT	set annotation text alignment
SET_ANNOTATION_TEXT_CHAR_HEIGHT	set annotation text character height
SET_ANNOTATION_TEXT_CHAR_UP_VECTOR	set annotation text character up vector
SET_ANNOTATION_TEXT_PATH	set annotation text path
SET_CHAR_EXPANSION_FACTOR	set character expansion factor
SET_CHAR_HEIGHT	set character height
SET_CHAR_SPACING	set character spacing
SET_CHAR_UP_VECTOR	set character up vector
SET_CHOICE_MODE	set choice mode
SET_COLOUR_MODEL	set colour model
SET_COLOUR_REPRESENTATION	set colour representation
SET_CONFLICT_RESOLUTION	set conflict resolution
SET_DISPLAY_UPDATE_STATE	set display update state
SET_EDGE_COLOUR_INDEX	set edge colour index
SET_EDGE_FLAG	set edge flag
SET_EDGE_INDEX	set edge index
SET_EDGE_REPRESENTATION	set edge representation
SET_EDGETYPE	set edgetype
SET_EDGEWIDTH_SCALE_FACTOR	set edgewidth scale factor
SET_EDIT_MODE	set edit mode
SET_ELEMENT_POINTER	set element pointer
SET_ELEMENT_POINTER_AT_LABEL	set element pointer at label
SET_ERROR_HANDLING_MODE	set error handling mode
SET_GLOBAL_TRANSFORMATION	set global transformation
SET_GLOBAL_TRANSFORMATION	set global transformation 3
SET_HIGHLIGHTING_FILTER	set highlighting filter
SET_HLHSR_IDENTIFIER	set hlhsr identifier
SET_HLHSR_MODE	set hlhsr mode
SET_INDIVIDUAL_ASF	set individual asf
SET_INTERIOR_COLOUR_INDEX	set interior colour index
SET_INTERIOR_INDEX	set interior index
SET_INTERIOR_REPRESENTATION	set interior representation
SET_INTERIOR_STYLE	set interior style
SET_INTERIOR_STYLE_INDEX	set interior style index
SET_INVISIBILITY_FILTER	set invisibility filter
SET_LINETYPE	set linetype
SET_LINEWIDTH_SCALE_FACTOR	set linewidth scale factor
SET_LOCAL_TRANSFORMATION	set local transformation

SET_LOCAL_TRANSFORMATION	set local transformation 3
SET_LOCATOR_MODE	set locator mode
SET_MARKER_SIZE_SCALE_FACTOR	set marker size scale factor
SET_MARKER_TYPE	set marker type
SET_MODELLING_CLIPPING_INDICATOR	set modelling clipping indicator
SET_MODELLING_CLIPPING_VOLUME	set modelling clipping volume
SET_MODELLING_CLIPPING_VOLUME	set modelling clipping volume 3
SET_PATTERN_REFERENCE_POINT	set pattern reference point
SET_PATTERN_REFERENCE_POINT_AND_VECTORS	set pattern reference point and vectors
SET_PATTERN_REPRESENTATION	set pattern representation
SET_PATTERN_SIZE	set pattern size
SET_PICK_FILTER	set pick filter
SET_PICK_IDENTIFIER	set pick identifier
SET_PICK_MODE	set pick mode
SET_POLYLINE_COLOUR_INDEX	set polyline colour index
SET_POLYLINE_INDEX	set polyline index
SET_POLYLINE_REPRESENTATION	set polyline representation
SET_POLYMARKER_COLOUR_INDEX	set polymarker colour index
SET_POLYMARKER_INDEX	set polymarker index
SET_POLYMARKER_REPRESENTATION	set polymarker representation
SET_STRING_MODE	set string mode
SET_STROKE_MODE	set stroke mode
SET_TEXT_ALIGNMENT	set text alignment
SET_TEXT_COLOUR_INDEX	set text colour index
SET_TEXT_FONT	set text font
SET_TEXT_INDEX	set text index
SET_TEXT_PATH	set text path
SET_TEXT_PRECISION	set text precision
SET_TEXT_REPRESENTATION	set text representation
SET_VALUATOR_MODE	set valuator mode
SET_VIEW_INDEX	set view index
SET_VIEW_REPRESENTATION	set view representation
SET_VIEW_REPRESENTATION	set view representation 3
SET_VIEW_TRANSFORMATION_INPUT_PRIORITY	set view transformation input priority
SET_WS_VIEWPORT	set workstation viewport
SET_WS_VIEWPORT	set workstation viewport 3
SET_WS_WINDOW	set workstation window
SET_WS_WINDOW	set workstation window 3
TEXT	text
TEXT	text 3
TRANSFORM_POINT	transform point
TRANSFORM_POINT	transform point 3
TRANSLATE	translate
TRANSLATE	translate 3
UNPOST_ALL_STRUCTURES	unpost all structures
UNPOST_STRUCTURE	unpost structure
UPDATE_WS	update workstation
WRITE_ITEM_TO_METAFILE	write item to metafile

4.1.3 Alphabetical PHIGS functions

The functions are in the same order when listed alphabetically according to the PHIGS function names as they are when listed alphabetically according to the bound names. Therefore, the table provided in 4.1.2 alphabetically lists the PHIGS functions.

4.2 Data type definitions

4.2.1 Abbreviations used in the data type definitions

ARCL	archive closed
AROP	archive open
ASAP	as soon as possible
ASF	aspect source flag
ASTI	at some time
BNIG	before next interaction globally
BNIL	before next interaction locally
CBS	can be simulated
CHAR	character
CSS	central structure store
DC	device coordinate
GDP	generalized drawing primitive
GSE	generalized structure element
HLHSR	hidden line/hidden surface removal
ID	identifier
IDS	identifiers
IMM	immediate
IND	index
IRG	implicit regeneration
MAX	maximum
MC	modelling coordinates
MI	metafile input
MIN	minimum
MISC	miscellaneous
MO	metafile output
NIVE	no immediate visual effect
NPC	normalized projection coordinates
PHCL	PHIGS closed
PHIGS	Programmer's Hierarchical Interactive Graphics System
PHOP	PHIGS open
PT	point
REF	reference
SF	scale factor
STCL	structure closed
STOP	structure open
UQUM	use quick update method
UWOR	update without regeneration
VC	viewing coordinates
WC	world coordinates
WS	workstation
WSCL	workstation closed
WSOP	workstation open

4.2.2 Alphabetical list of type definitions

This subclause contains an alphabetical listing of all of the data type definitions used to define the Ada binding to PHIGS. Each element declared includes a comment about the type and/or use of the type. Some of the declarations employ constant values in the definition of the type. All of these constant declarations are included in the PHIGS_TYPES package.

ACCESS_COLOUR_MATRIX

```
type ACCESS_COLOUR_MATRIX is access COLOUR_MATRIX;

-- Provides for returning variable sized matrices containing colour
-- indices corresponding to a cell array or pattern array.
```

ACCESS_STRING

```
type ACCESS_STRING is access PHIGS_STRING;

-- Defines a variable length string needed to return strings in
-- structure element records.
```

ANGLE

```
type ANGLE is digits PHIGS_PRECISION;

-- Values used in the modelling transformation utility functions
-- for specifying angles of rotation in radians.  Positive
-- indicates a counterclockwise direction.
```

ANNOTATION_STYLE

```
type ANNOTATION_STYLE is new PHIGS_INTEGER;

-- Defines the annotation styles for annotation primitives.
```

ANNOTATION_STYLES

```
package ANNOTATION_STYLES is
        new PHIGS_LIST_UTILITIES (ANNOTATION_STYLE,
                                  MAX_ANNOTATION_STYLES_SUPPORTED);

-- Provides for lists of annotation styles.
```

APPLICATION_DATA_RANGE

```
subtype APPLICATION_DATA_RANGE is
        PHIGS_NATURAL range 0..MAX_APPLICATION_DATA;

-- Defines range of lengths for application data objects.
```

APPLICATION_DATA_RECORD

```
type APPLICATION_DATA_RECORD (LENGTH : APPLICATION_DATA_RANGE := 0) is
        record
          DATA : PHIGS_STRING (1..LENGTH);
        end record;

-- Defines type for defining application data objects.
```

ARCHIVE_ID

```
type ARCHIVE_ID is new PHIGS_NATURAL;

-- Provides for an application specified pointer to archive
-- files separate from the file identifier.
```

ARCHIVE_IDS

```
package ARCHIVE_IDS is new PHIGS_LIST_UTILITIES (ARCHIVE_ID,
                                                 MAX_ARCHIVE_IDS_SUPPORTED);

-- Provides for lists of archive identifiers.
```

ARCHIVE_STATE

```
type ARCHIVE_STATE is (ARCL,
                       AROP);

-- The type used to return the archive state.
```

ASF

```
type ASF is (BUNDLED,
             INDIVIDUAL);

-- This type defines an aspect source flag whose
-- value indicates whether an aspect of a primitive
-- should be set from a bundle table or from an
-- individual attribute.
```

ASPECT

```
type ASPECT is (TYPE_OF_LINE,
                LINEWIDTH_SF,
                LINE_COLOUR,
```

```
                    TYPE_OF_MARKER,
                    SIZE,
                    MARKER_COLOUR,

                    FONT,
                    PRECISION,
                    EXPANSION,
                    SPACING,
                    TEXT_COLOUR,

                    STYLE_OF_INTERIOR,
                    STYLE_IND,
                    INTERIOR_COLOUR,

                    FLAG,
                    TYPE_OF_EDGE,
                    EDGEWIDTH_SF,
                    EDGE_COLOUR);

    -- This type lists the aspects for which an aspect source
    -- flag exists.
```

ATTRIBUTES_USED

```
    package ATTRIBUTES_USED is
            new PHIGS_LIST_UTILITIES
                (ATTRIBUTES_USED_TYPE,
                 ATTRIBUTES_USED_TYPE'POS(ATTRIBUTES_USED_TYPE'LAST)+1);

    -- Provides for a list of the attributes used which is
    -- returned by the INQ_GDP and LOCATOR_ATTRIBUTES_USED
    -- functions.
```

ATTRIBUTES_USED_TYPE

```
    type ATTRIBUTES_USED_TYPE is (POLYLINE_ATTRIBUTES,
                                  POLYMARKER_ATTRIBUTES,
                                  TEXT_ATTRIBUTES,
                                  INTERIOR_ATTRIBUTES,
                                  EDGE_ATTRIBUTES);

    -- The types of attributes which may be used in generating
    -- output for a GDP and in generating prompt and echo
    -- information for certain prompt and echo types of certain
    -- classes of input devices.
```

CHAR_EXPANSION

```
    type CHAR_EXPANSION is new SCALE_FACTOR range
            SCALE_FACTOR'SAFE_SMALL..SCALE_FACTOR'LAST;

    -- Defines a character expansion factor.
```

CHAR_SET

```
type CHAR_SET is new PHIGS_NATURAL;

-- Provides identifications for the available character
-- sets.  ISO 646 character set maps to value zero.
```

CHAR_SETS

```
package CHAR_SETS is new PHIGS_LIST_UTILITIES (CHAR_SET,
                                               MAX_CHAR_SETS_SUPPORTED);

-- Provides for lists of character set identifications.
```

CHAR_SPACING

```
type CHAR_SPACING is new SCALE_FACTOR;

-- Defines a character spacing factor.  The factors are
-- unitless.  A positive value indicates the amount of extra
-- space between character boxes in a text string, and a
-- negative value indicates the amount of overlap between
-- character boxes in a text string.
```

CHOICE_DEVICE_NUMBER

```
type CHOICE_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for choice device identifiers.
```

CHOICE_NUMBER

```
type CHOICE_NUMBER is new PHIGS_POSITIVE;

-- Defines the choice numbers available on an implementation.
```

CHOICE_PROMPT

```
type CHOICE_PROMPT is new OFF_ON;

-- Indicates whether a specified choice prompt is to be displayed
-- or not.
```

CHOICE_PROMPTS

```
package CHOICE_PROMPTS is
        new PHIGS_LIST_UTILITIES
                (CHOICE_PROMPT,
                 MAX_CHOICE_PROMPTS_SUPPORTED);

-- Provides for lists of choice prompts.
```

CHOICE_PROMPT_ECHO_TYPE

```
type CHOICE_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the choice prompt and echo type.
```

CHOICE_PROMPT_ECHO_TYPES

```
package CHOICE_PROMPT_ECHO_TYPES is
        new PHIGS_LIST_UTILITIES
                (CHOICE_PROMPT_ECHO_TYPE,
                 MAX_CHOICE_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of choice prompt and echo types.
```

CHOICE_PROMPT_STRING

```
type CHOICE_PROMPT_STRING (LENGTH : STRING_SMALL_NATURAL := 0) is
        record
          CONTENTS : PHIGS_STRING (1..LENGTH);
        end record;

-- Provides for a variable length prompt.  Objects of this
-- type should be declared unconstrained to allow for dynamic
-- modification of the length.
```

CHOICE_PROMPT_STRING_ARRAY

```
type CHOICE_PROMPT_STRING_ARRAY is array (PHIGS_POSITIVE range <>) of
        CHOICE_PROMPT_STRING;

-- Provides for an array of prompt strings.
```

CHOICE_PROMPT_STRING_LIST

```
type CHOICE_PROMPT_STRING_LIST (LENGTH : CHOICE_SMALL_NATURAL := 0) is
        record
          LIST : CHOICE_PROMPT_STRING_ARRAY (1..LENGTH);
        end record;

-- Provides for lists of prompt strings.
```

CHOICE_REQUEST_STATUS

```
type CHOICE_REQUEST_STATUS is (OK,
                               NOCHOICE,
                               NONE);

-- Defines the status of a choice input operation for the
-- request function.
```

CHOICE_SMALL_NATURAL

```
subtype CHOICE_SMALL_NATURAL is
        PHIGS_NATURAL range 0..MAX_CHOICE_SMALL_NATURAL;

-- This is a subtype declaration which allows for
-- unconstrained record objects for CHOICE_PROMPT_STRING_LIST
-- type without causing the exception STORAGE_ERROR to be raised.
```

CHOICE_STATUS

```
subtype CHOICE_STATUS is
        CHOICE_REQUEST_STATUS range OK..NOCHOICE;

-- Indicates if a choice was made by the operator for the
-- sample, get, and inquiry functions.
```

CHROMATICITY_COEFFICIENT

```
type CHROMATICITY_COEFFICIENT is
        record
          X : COLOUR_COEFFICIENT;
          Y : COLOUR_COEFFICIENT;
        end record;

-- Defines a CIE colour coefficient pair.
```

CHROMATICITY_COEFFICIENT_SET

```
type CHROMATICITY_COEFFICIENT_SET is
        record
          R  : CHROMATICITY_COEFFICIENT;
          G  : CHROMATICITY_COEFFICIENT;
          B  : CHROMATICITY_COEFFICIENT;
        end record;

-- Defines a set of CIE primary colour chromaticity coefficients.
```

CLIPPING_INDICATOR

```
type CLIPPING_INDICATOR is (CLIP,
                            NOCLIP);

-- Indicates whether or not clipping is to be performed.
```

COLOUR_AVAILABLE

```
type COLOUR_AVAILABLE is (COLOUR,
                          MONOCHROME);

-- Indicates whether colour output is available on a
-- workstation.
```

COLOUR_COEFFICIENT

```
type COLOUR_COEFFICIENT is digits PHIGS_PRECISION;

-- Defines a general type for colour components.
```

COLOUR_COEFFICIENT_ARRAY

```
type COLOUR_COEFFICIENT_ARRAY is
        array (COLOUR_COMPONENTS) of COLOUR_COEFFICIENT;

-- Defines a general type for colour components.
```

COLOUR_COEFFICIENTS

```
package COLOUR_COEFFICIENTS is
        new PHIGS_LIST_UTILITIES (COLOUR_COEFFICIENT,
                                  MAX_COLOUR_COEFFICIENTS);

-- Provides for a lists of colour coefficients for the generalized
-- colour model.
```

COLOUR_COMPONENTS

```
type COLOUR_COMPONENTS is range 1..MAX_COLOUR_COEFFICIENTS;

-- Defines a type for the size of colour component arrays.
```

COLOUR_INDEX

```
type COLOUR_INDEX is new PHIGS_NATURAL;

-- Indices into colour tables are of this type.
```

COLOUR_INDICES

```
package COLOUR_INDICES is
        new PHIGS_LIST_UTILITIES (COLOUR_INDEX,
                                  MAX_COLOUR_INDICES_SUPPORTED);

-- Provides for a set of colour indices which are available
-- on a particular workstation.
```

COLOUR_MATRIX

```
type COLOUR_MATRIX is array (COLOUR_MATRIX_SMALL_NATURAL range <>,
                             COLOUR_MATRIX_SMALL_NATURAL range <>)
                      of COLOUR_INDEX;

-- Provides for matrices containing colour indices corresponding
-- to a cell array or pattern array.
```

COLOUR_MATRIX_SMALL_NATURAL

```
subtype COLOUR_MATRIX_SMALL_NATURAL is
        PHIGS_NATURAL range 0..MAX_COLOUR_MATRIX_SMALL_NATURAL;

-- This is a subtype declaration which allows for unconstrained
-- record objects for COLOUR_MATRIX type without causing the
-- exception STORAGE_ERROR to be raised.
```

COLOUR_MODEL

```
type COLOUR_MODEL is new PHIGS_INTEGER;

-- Indicates the colour models available in PHIGS.
```

COLOUR_MODELS

```
package COLOUR_MODELS is
        new PHIGS_LIST_UTILITIES (COLOUR_MODEL,
                                  MAX_COLOUR_MODELS_SUPPORTED);

-- Provides for lists of colour models.
```

COLOUR_REPRESENTATION

```
type COLOUR_REPRESENTATION (MODEL : COLOUR_MODEL := RGB) is
      record
        case MODEL is

          when RGB = >
            RED_RGB                 : INTENSITY;
            GREEN_RGB               : INTENSITY;
            BLUE_RGB                : INTENSITY;

          when CIELUV = >
            L_STAR_CIE              : COLOUR_COEFFICIENT;
            U_STAR_CIE              : COLOUR_COEFFICIENT;
            V_STAR_CIE              : COLOUR_COEFFICIENT;

          when HSV = >
            HUE_HSV                 : INTENSITY;
            SATURATION_HSV          : INTENSITY;
            VALUE_HSV               : INTENSITY;

          when HLS = >
            HUE_HLS                 : INTENSITY;
            LIGHTNESS_HLS           : INTENSITY;
            SATURATION_HLS          : INTENSITY;

          when others = >
            GENERIC_COMPONENTS  : COLOUR_COEFFICIENT_ARRAY;

        end case;
      end record;

-- Defines the representation of a colour as a combination of components
-- whose meaning depends on the colour model.  Additional variants may be
-- included when additional registered or unregistered colour models
-- are supported by an implementation.
```

COMPOSITION_TYPE

```
type COMPOSITION_TYPE is (PRECONCATENATE,
                          POSTCONCATENATE,
                          REPLACE);

-- Defines the type of matrix composition allowed between
-- modelling transformations.
```

CONFLICT_RESOLUTION

```
type CONFLICT_RESOLUTION is (MAINTAIN,
                             ABANDON,
                             UPDATE);

-- Defines a type for specifying the conflict resolution mode.
```

CONNECTION_ID

```
type CONNECTION_ID is new PHIGS_STRING;

-- Provides for specifying connection identifiers.
```

CONTROL_FLAG

```
type CONTROL_FLAG is (CONDITIONALLY,
                      ALWAYS);

-- The control flag is used to indicate the conditions
-- under which the display surface should be cleared.
```

DC

```
package DC is new PHIGS_COORDINATE_SYSTEM (DC_TYPE);

-- Defines the Device Coordinate System.
```

DC_TYPE

```
type DC_TYPE is digits PHIGS_PRECISION;

-- The type of a coordinate component in the Device Coordinate
-- System.
```

DC_UNITS

```
type DC_UNITS is (METRES,
                  OTHER);

-- Device coordinate units for a particular workstation
-- may be in metres, or some other unit (such as inches).
```

DEFERRAL_MODE

```
type DEFERRAL_MODE is (ASAP,
                       BNIG,
                       BNIL,
                       ASTI,
                       WAIT);

-- Defines the five PHIGS deferral modes.
```

DEVICE_NUMBER

```
subtype DEVICE_NUMBER is PHIGS_POSITIVE;

-- Defines the base type for input device numbers.
```

DISPLAY_CLASS

```
type DISPLAY_CLASS is (VECTOR_DISPLAY,
                       RASTER_DISPLAY,
                       OTHER_DISPLAY);

-- The classification of a workstation of category OUTPUT
-- or OUTIN.
```

DISPLAY_PRIORITY

```
type DIPLAY_PRIORITY is
      digits PHIGS_PRECISION range 0.0..1.0;

-- Defines type for specifying structure display priority.
```

DISPLAY_SURFACE_EMPTY

```
type DISPLAY_SURFACE_EMPTY is (EMPTY,
                               NOTEMPTY);

-- Indicates whether there is graphical output on the display
-- surface.
```

DYNAMIC_MODIFICATION

```
type DYNAMIC_MODIFICATION is (IRG,
                              IMM,
                              CBS);

-- Indicates whether an update to the state list is performed
-- immediately (IMM), requires implicit regeneration (IRG), or
-- can be simulated (CBS).
```

ECHO_SWITCH

```
type ECHO_SWITCH is (ECHO,
                     NOECHO);

-- Indicates whether or not echoing of the measure is
-- performed.
```

EDGE_DATA

```
type EDGE_DATA is record
                    FLAG_ASF              : ASF;
                    TYPE_OF_EDGE_ASF      : ASF;
                    EDGEWIDTH_SF_ASF      : ASF;
                    EDGE_COLOUR_ASF       : ASF;
                    FLAG                  : EDGE_FLAG;
                    TYPE_OF_EDGE          : EDGETYPE;
                    EDGEWIDTH_SF          : EDGEWIDTH;
                    EDGE_COLOUR           : COLOUR_INDEX;
                  end record;

-- A record containing information needed to specify the
-- appearance of a fill area set edge.
```

EDGE_FLAG

```
type EDGE_FLAG is new OFF_ON;

-- Defines a type for indicating whether or not to depict
-- edges on fill area set and related primitives.
```

EDGE_INDEX

```
type EDGE_INDEX is new PHIGS_POSITIVE;

-- Defines the edge bundle table indices.
```

EDGE_INDICES

```
package EDGE_INDICES is
        new PHIGS_LIST_UTILITIES (EDGE_INDEX,
                                  MAX_EDGE_INDICES_SUPPORTED);

-- Provides for a list of edge indices.
```

EDGETYPE

```
type EDGETYPE is new PHIGS_INTEGER;

-- Defines a type for describing the form of edges.
```

EDGETYPES

```
package EDGETYPES is new PHIGS_LIST_UTILITIES (EDGETYPE,
                                               MAX_EDGETYPES_SUPPORTED);

-- Provides for lists of edgetypes.
```

EDGEWIDTH

```
type EDGEWIDTH is
        new SCALE_FACTOR range 0.0..SCALE_FACTOR'LAST;

-- Defines a type for describing the width scale factor of
-- edges.
```

EDIT_MODE

```
type EDIT_MODE is (INSERT,
                   REPLACE);

-- Defines a type for specifying the mode in which editing
-- takes place.
```

ELEMENT_POSITION

```
type ELEMENT_POSITION is new PHIGS_INTEGER;

-- Base type for element pointer values and offsets.
```

ELEMENT_TYPE

```
type ELEMENT_TYPE is
        (ALL_ELEMENT_TYPES,
         NIL,

         POLYLINE_3,
         POLYLINE,

         POLYMARKER_3,
         POLYMARKER,

         TEXT_3,
         TEXT,

         ANNOTATION_TEXT_RELATIVE_3,
         ANNOTATION_TEXT_RELATIVE,

         FILL_AREA_3,
         FILL_AREA,
         FILL_AREA_SET_3,
         FILL_AREA_SET,

         CELL_ARRAY_3,
         CELL_ARRAY,

         GDP_3,
         GDP,

         SET_POLYLINE_INDEX,
         SET_POLYMARKER_INDEX,
         SET_TEXT_INDEX,
         SET_INTERIOR_INDEX,
         SET_EDGE_INDEX,

         SET_LINETYPE,
         SET_LINEWIDTH_SCALE_FACTOR,
         SET_POLYLINE_COLOUR_INDEX,

         SET_MARKER_TYPE,
         SET_MARKER_SIZE_SCALE_FACTOR,
         SET_POLYMARKER_COLOUR_INDEX,

         SET_TEXT_FONT,
         SET_TEXT_PRECISION,
         SET_CHAR_EXPANSION_FACTOR,
         SET_CHAR_SPACING,
         SET_TEXT_COLOUR_INDEX,
         SET_CHAR_HEIGHT,
         SET_CHAR_UP_VECTOR,
         SET_TEXT_PATH,
         SET_TEXT_ALIGNMENT,

         SET_ANNOTATION_TEXT_CHAR_HEIGHT,
         SET_ANNOTATION_TEXT_CHAR_UP_VECTOR,
         SET_ANNOTATION_TEXT_PATH,
         SET_ANNOTATION_TEXT_ALIGNMENT,
         SET_ANNOTATION_STYLE,
```

```
        SET_INTERIOR_STYLE,
        SET_INTERIOR_STYLE_INDEX,
        SET_INTERIOR_COLOUR_INDEX,

        SET_EDGE_FLAG,
        SET_EDGETYPE,
        SET_EDGEWIDTH_SCALE_FACTOR,
        SET_EDGE_COLOUR_INDEX,

        SET_PATTERN_SIZE,
        SET_PATTERN_REFERENCE_POINT_AND_VECTORS,
        SET_PATTERN_REFERENCE_POINT,

        ADD_NAMES_TO_SET,
        REMOVE_NAMES_FROM_SET,

        SET_INDIVIDUAL_ASF,
        SET_HLHSR_IDENTIFIER,

        SET_LOCAL_TRANSFORMATION_3,
        SET_LOCAL_TRANSFORMATION,
        SET_GLOBAL_TRANSFORMATION_3,
        SET_GLOBAL_TRANSFORMATION,
        SET_MODELLING_CLIPPING_VOLUME_3,
        SET_MODELLING_CLIPPING_VOLUME,
        SET_MODELLING_CLIPPING_INDICATOR,
        RESTORE_MODELLING_CLIPPING_VOLUME,
        SET_VIEW_INDEX,

        EXECUTE_STRUCTURE,

        LABEL,
        APPLICATION_DATA,
        GSE,
        SET_PICK_IDENTIFIER);

    -- Provides for specification of element types.

-------------------------------------------------------------------------------

ELEMENT_TYPES

    package ELEMENT_TYPES is
        new PHIGS_LIST_UTILITIES (ELEMENT_TYPE,
                                  ELEMENT_TYPE'POS(ELEMENT_TYPE'LAST)+1);

    -- Provides for lists of element types.

-------------------------------------------------------------------------------

ERROR_HANDLING_MODE

    type ERROR_HANDLING_MODE is new OFF_ON;

    -- Type for inquiring the error handling mode.
```

ERROR_NUMBER

```
type ERROR_NUMBER is new PHIGS_INTEGER;

-- Defines the type for error indicator values.
```

ESCAPE_ID

```
type ESCAPE_ID is new PHIGS_INTEGER;

-- Defines a type for identifying different escapes.
```

ESCAPE_IDS

```
package ESCAPE_IDS is new PHIGS_LIST_UTILITIES (ESCAPE_ID,
                                                MAX_ESCAPE_IDS_SUPPORTED);

-- Provides for lists of Escape ID's.
```

ESCAPE_DATA_RECORD

```
type ESCAPE_DATA_RECORD (ESCAPE : ESCAPE_ID := 0) is
        record
          case ESCAPE is

            -- For each ESCAPE which needs a record, an
            -- implementation dependent entry is defined.
            -- For others, a null record is defined.

            when others = >
               null;
          end case;
        end record;

-- Provides for the definition of ESCAPE data records.
```

EVENT_DEVICE_NUMBER

```
type EVENT_DEVICE_NUMBER (CLASS : INPUT_CLASS := NONE) is
        record
          case CLASS is

            when NONE = > null;

            when LOCATOR_INPUT
               = > LOCATOR_EVENT_DEVICE : LOCATOR_DEVICE_NUMBER;

            when STROKE_INPUT
               = > STROKE_EVENT_DEVICE : STROKE_DEVICE_NUMBER;
```

```
                when VALUATOR_INPUT
                  => VALUATOR_EVENT_DEVICE : VALUATOR_DEVICE_NUMBER;

                when CHOICE_INPUT
                  => CHOICE_EVENT_DEVICE : CHOICE_DEVICE_NUMBER;

                when PICK_INPUT
                  => PICK_EVENT_DEVICE : PICK_DEVICE_NUMBER;

                when STRING_INPUT
                  => STRING_EVENT_DEVICE : STRING_DEVICE_NUMBER;

              end case;
            end record;

        -- Provides for returning any class of device number from the
        -- event queue.
```

EVENT_QUEUE_DEVICE_NUMBER

```
        type EVENT_QUEUE_DEVICE_NUMBER
                (CLASS : INPUT_QUEUE_CLASS := LOCATOR_INPUT) is
            record
              case CLASS is

                when LOCATOR_INPUT
                  => LOCATOR_EVENT_DEVICE : LOCATOR_DEVICE_NUMBER;

                when STROKE_INPUT
                  => STROKE_EVENT_DEVICE : STROKE_DEVICE_NUMBER;

                when VALUATOR_INPUT
                  => VALUATOR_EVENT_DEVICE : VALUATOR_DEVICE_NUMBER;

                when CHOICE_INPUT
                  => CHOICE_EVENT_DEVICE : CHOICE_DEVICE_NUMBER;

                when PICK_INPUT
                  => PICK_EVENT_DEVICE : PICK_DEVICE_NUMBER;

                when STRING_INPUT
                  => STRING_EVENT_DEVICE : STRING_DEVICE_NUMBER;

              end case;
            end record;

        -- Allows EVENT_DEVICE_NUMBERs which cannot have value NONE.  This is
        -- used when returning only real input classes as causes of event
        -- queue overflow and for flushing the queue.
```

FILE_ID

```
        subtype FILE_ID is PHIGS_STRING;

        -- Provides for file identifiers.
```

FILE_SMALL_NATURAL

```
subtype FILE_SMALL_NATURAL is
        PHIGS_NATURAL range 0..MAX_FILE_SMALL_NATURAL;

-- This is a subtype declaration which allows for unconstrained
-- record objects for VARIABLE_FILE_ID type without
-- causing the exception STORAGE_ERROR to be raised.
```

GDP_3_ID

```
type GDP_3_ID is new PHIGS_INTEGER;

-- Defines a type for selecting a Generalized Drawing
-- Primitive.
```

GDP_3_IDS

```
package GDP_3_IDS is new PHIGS_LIST_UTILITIES (GDP_3_ID,
                                               MAX_GDP_3_IDS_SUPPORTED);

-- Provides for lists of 3D Generalized Drawing Primitive ID's.
```

GDP_3_RECORD

```
type GDP_3_RECORD (GDP : GDP_3_ID := 0) is
        record
          case GDP is

            -- For each supported GDP_3 which needs a record, an
            -- implementation dependent entry is defined.
            -- For others, a generic record is defined.

            when others =>
                GDP_INFO : APPLICATION_DATA_RECORD;
          end case;
        end record;

-- Provides for the definition of GDP_3 data records.
```

GDP_ID

```
type GDP_ID is new PHIGS_INTEGER;

-- Defines a type for selecting a Generalized Drawing
-- Primitive.
```

GDP_IDS

```
package GDP_IDS is new PHIGS_LIST_UTILITIES (GDP_ID,
                                              MAX_GDP_IDS_SUPPORTED);

-- Provides for lists of Generalized Drawing Primitive ID's.
```

GDP_RECORD

```
type GDP_RECORD (GDP : GDP_ID := 0) is
        record
          case GDP is

            -- For each supported GDP which needs a record, an
            -- implementation dependent entry is defined.
            -- For others, a generic record is defined.

            when others = >
               GDP_INFO : APPLICATION_DATA_RECORD;
          end case;
        end record;

-- Provides for the definition of GDP data records.
```

GENERAL_FLOAT_PARAMETER

```
type GENERAL_FLOAT_PARAMETER is digits PHIGS_PRECISION;

-- Defines a type for specifying implementation-specific real
-- data in GDP, GSE, and ESCAPE data records.
```

GSE_ID

```
type GSE_ID is new PHIGS_INTEGER;

-- Defines a type for selecting a Generalized Structure
-- Element.
```

GSE_IDS

```
package GSE_IDS is new PHIGS_LIST_UTILITIES (GSE_ID,
                                             MAX_GSE_IDS_SUPPORTED);

-- Provides for lists of Generalized Structure Element ID's.
```

GSE_RECORD

```
type GSE_RECORD (GSE : GSE_ID := 0) is
        record
          case GSE is

            -- For each supported GSE which needs a record, an
            -- implementation dependent entry is defined.
            -- For others, a generic record is defined.

            when others = >
               GSE_INFO : APPLICATION_DATA_RECORD;
          end case;
        end record;
```

-- Provides for the definition of GSE data records.

HATCH_STYLE

```
subtype HATCH_STYLE is STYLE_INDEX;
```

-- Defines the interior hatch styles type.

HATCH_STYLES

```
package HATCH_STYLES is
            new PHIGS_LIST_UTILITIES (HATCH_STYLE,
                                      MAX_HATCH_STYLES_SUPPORTED);
```

-- Provides for lists of hatch styles.

HLHSR_ID

```
type HLHSR_ID is new PHIGS_INTEGER;
```

-- A number used as an attribute for workstation independent
-- HLHSR factors.

HLHSR_IDS

```
package HLHSR_IDS is new PHIGS_LIST_UTILITIES (HLHSR_ID,
                                               MAX_HLHSR_IDS_SUPPORTED);
```

-- Provides for lists of HLHSR identifiers.

HLHSR_MODE

```
type HLHSR_MODE is new PHIGS_INTEGER;

-- A number used for controlling workstation dependent
-- HLHSR factors.
```

HLHSR_MODES

```
package HLHSR_MODES is
        new PHIGS_LIST_UTILITIES (HLHSR_MODE,
                                  MAX_HLHSR_MODES_SUPPORTED);

-- Provides for lists of HLHSR modes.
```

HORIZONTAL_ALIGNMENT

```
type HORIZONTAL_ALIGNMENT is (NORMAL,
                              LEFT,
                              CENTRE,
                              RIGHT);

-- The alignment of the text extent rectangle with
-- respect to horizontal positioning of the text.
```

INPUT_CLASS

```
type INPUT_CLASS is (NONE,
                     LOCATOR_INPUT,
                     STROKE_INPUT,
                     VALUATOR_INPUT,
                     CHOICE_INPUT,
                     PICK_INPUT,
                     STRING_INPUT);

-- Defines the input device classifications for
-- workstations of category INPUT or OUTIN.
```

INPUT_QUEUE_CLASS

```
subtype INPUT_QUEUE_CLASS is
        INPUT_CLASS range LOCATOR_INPUT .. STRING_INPUT;

-- Defines the input device classifications for workstations
-- of category INPUT and OUTIN that do not return a NONE
-- classification.
```

INPUT_STATUS

```
type INPUT_STATUS is (OK,
                      NONE);

-- Defines the status of a locator, stroke, valuator, or
-- string operation.
```

INPUT_STRING

```
type INPUT_STRING is (LENGTH : INPUT_STRING_SMALL_NATURAL := 0) is
        record
          CONTENTS : PHIGS_STRING (1..LENGTH);
        end record;

-- Provides a variable length string for use in STRING input functions.
-- Objects of this type should be declared unconstrained to allow for
-- dynamic modification of the length.
```

INPUT_STRING_SMALL_NATURAL

```
subtype INPUT_STRING_SMALL_NATURAL is
          PHIGS_NATURAL range 0..MAX_INPUT_STRING_SMALL_NATURAL;

-- This is a subtype declaration which allows for unconstrained
-- record objects for INPUT_STRING type without
-- causing the exception STORAGE_ERROR to be raised.
```

INTENSITY

```
subtype INTENSITY is COLOUR_COEFFICIENT range 0.0..1.0;

-- Defines the restricted range of colour components required
-- by some colour models.
```

INTERIOR_DATA

```
type INTERIOR_DATA is
        record
          STYLE_OF_INTERIOR_ASF : ASF;
          STYLE_IND_ASF         : ASF;
          INTERIOR_COLOUR_ASF   : ASF;
          INTERIOR_IND          : INTERIOR_INDEX;
          STYLE_OF_INTERIOR     : INTERIOR_STYLE;
          STYLE_IND             : STYLE_INDEX;
          INTERIOR_COLOUR       : COLOUR_INDEX;
        end record;

-- A record containing information needed to specify the
-- appearance of a filled area interior.
```

INTERIOR_INDEX

```
type INTERIOR_INDEX is new PHIGS_POSITIVE;

-- Defines interior bundle table indices.
```

INTERIOR_INDICES

```
package INTERIOR_INDICES is
        new PHIGS_LIST_UTILITIES (INTERIOR_INDEX,
                                  MAX_INTERIOR_INDICES_SUPPORTED);

-- Provides for lists of interior bundle table indices.
```

INTERIOR_STYLE

```
type INTERIOR_STYLE is (HOLLOW,
                        SOLID,
                        PATTERN,
                        HATCH,
                        EMPTY);

-- Defines the interior styles for filled areas.
```

INTERIOR_STYLES

```
package INTERIOR_STYLES is
        new PHIGS_LIST_UTILITIES
              (INTERIOR_STYLE,
               INTERIOR_STYLE'POS(INTERIOR_STYLE'LAST)+1);

-- Provides for lists of interior styles.
```

INVALID_VALUES_INDICATOR

```
type INVALID_VALUES_INDICATOR is (ABSENT,
                                  PRESENT);

-- Indicates whether invalid values are contained in
-- a colour array or matrix.
```

LABEL_ID

```
type LABEL_ID is new PHIGS_INTEGER;

-- Defines a type for specifying structure element labels.
```

LINE_DATA

```
type LINE_DATA is
      record
        TYPE_OF_LINE_ASF  : ASF;
        WIDTH_ASF         : ASF;
        LINE_COLOUR_ASF   : ASF;
        POLYLINE_IND      : POLYLINE_INDEX;
        TYPE_OF_LINE      : LINETYPE;
        WIDTH             : LINEWIDTH;
        LINE_COLOUR       : COLOUR_INDEX;
      end record;

-- A record containing information needed to specify the
-- appearance of a line for input data records.
```

LINETYPE

```
type LINETYPE is new PHIGS_INTEGER;

-- Defines the types of line styles provided by PHIGS.
```

LINETYPES

```
package LINETYPES is new PHIGS_LIST_UTILITIES (LINETYPE,
                                               MAX_LINETYPES_SUPPORTED);

-- Provides for lists of linetypes.
```

LINEWIDTH

```
type LINEWIDTH is
        new SCALE_FACTOR range 0.0..SCALE_FACTOR'LAST;

-- The width of a line is indicated by a scale factor.
```

LOCATOR_DEVICE_NUMBER

```
type LOCATOR_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for locator device identifiers.
```

LOCATOR_PROMPT_ECHO_TYPE

```
type LOCATOR_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the locator prompt and echo types supported by
-- the implementation.
```

LOCATOR_PROMPT_ECHO_TYPES

```
package LOCATOR_PROMPT_ECHO_TYPES is
        new PHIGS_LIST_UTILITIES
                (LOCATOR_PROMPT_ECHO_TYPE,
                 MAX_LOCATOR_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of locator prompt and echo types.
```

MARKER_DATA

```
type MARKER_DATA is
        record
          TYPE_OF_MARKER_ASF  : ASF;
          SIZE_ASF            : ASF;
          MARKER_COLOUR_ASF   : ASF;
          POLYMARKER_IND      : POLYMARKER_INDEX;
          TYPE_OF_MARKER      : MARKER_TYPE;
          SIZE                : MARKER_SIZE;
          MARKER_COLOUR       : COLOUR_INDEX;
        end record;

-- A record containing information needed to specify the
-- appearance of a marker for input data records.
```

MARKER_SIZE

```
type MARKER_SIZE is new SCALE_FACTOR;

-- The size of a marker is indicated by a scale factor.
```

MARKER_TYPE

```
type MARKER_TYPE is new PHIGS_INTEGER;

-- Defines the type for markers provided by PHIGS.
```

MARKER_TYPES

```
package MARKER_TYPES is
        new PHIGS_LIST_UTILITIES (MARKER_TYPE,
                                  MAX_MARKER_TYPES_SUPPORTED);

-- Provides for lists of marker types.
```

MC

```
package MC is new PHIGS_COORDINATE_SYSTEM (MC_TYPE);

-- Defines the Modelling Coordinate System.
```

MC_TYPE

```
type MC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of coordinate in the Modelling
-- Coordinate System.
```

MEMORY_UNITS

```
type MEMORY_UNITS is range 0..MAX_MEMORY_UNITS;

-- Defines the type for units of memory that may be
-- allocated by PHIGS.
```

METAFILE_ITEM_LENGTH

```
type METAFILE_ITEM_LENGTH is
        new PHIGS_NATURAL range 0..MAX_METAFILE_ITEM_LENGTH;

-- The length of an item contained in a metafile.
```

METAFILE_ITEM_TYPE

```
type METAFILE_ITEM_TYPE is
        new PHIGS_NATURAL
                range MIN_METAFILE_ITEM_TYPE..MAX_METAFILE_ITEM_TYPE;

-- The type of an item contained in a metafile.
```

MODELLING_CLIP_OPERATION_TYPE

```
type MODELLING_CLIP_OPERATION_TYPE is new PHIGS_INTEGER;

-- This type provides for specifying modelling clip operation
-- types.
```

MODELLING_CLIP_OPERATION_TYPES

```
package MODELLING_CLIP_OPERATION_TYPES is
        new PHIGS_LIST_UTILITIES
                (MODELLING_CLIP_OPERATION_TYPE,
                 MAX_MODELLING_CLIP_OPERATION_TYPES_SUPPORTED);

-- This type provides for lists of modelling clip operation types.
```

MODIFICATION_MODE

```
type MODIFICATION_MODE is (NIVE,
                           UWOR,
                           UQUM);

-- Defines type for specifying modification modes.
```

MORE_EVENTS

```
type MORE_EVENTS is (NOMORE,
                     MORE);

-- Indicates whether more simultaneous events are contained
-- in the input queue.
```

NAME

```
subtype NAME is
        PHIGS_NAME_SET_FACILITY.NAME;

-- Provides for names for each of the name set classes.
-- See 5.14.4.
```

NAMES

```
package NAMES renames
        PHIGS_NAME_SET_FACILITY.NAMES;

-- Provides for lists of name set classes.  See 5.14.4.
```

NAME_SET

```
subtype NAME_SET is PHIGS_NAME_SET_FACILITY.NAME_SET;

-- Used to define name sets.  The internal structure of
-- a name set is implementation-dependent.  "Build" functions
-- contained in the package PHIGS_NAME_SET_FACILITIES shall be
-- used to create and manipulate name sets (See 5.14.4).
```

NAME_SET_ACCEPTANCE

```
subtype NAME_SET_ACCEPTANCE is
        PHIGS_NAME_SET_FACILITY.NAME_SET_ACCEPTANCE;

-- Used to indicate the results of applying a name set
-- against a filter pair.  See 5.14.4.
```

NAME_SET_FILTER

```
subtype NAME_SET_FILTER is
        PHIGS_NAME_SET_FACILITY.NAME_SET_FILTER;

-- Used to define name set pairs used as filters.
-- See 5.14.4.
```

NAME_SET_FILTERS

```
package NAME_SET_FILTERS is
        new PHIGS_LIST_UTILITIES
                (NAME_SET_FILTER,
                 MAX_NAME_SET_FILTER_LIST_LENGTH_SUPPORTED);

-- Used to provide lists of name set filters.
-- See 5.14.4.
```

NAME_SET_MEMBERSHIP

```
subtype NAME_SET_MEMBERSHIP is
        PHIGS_NAME_SET_FACILITY.NAME_SET_MEMBERSHIP;

-- Provides for indicating whether a class is contained in a
-- name set.  See 5.14.4.
```

NPC

```
package NPC is new PHIGS_COORDINATE_SYSTEM (NPC_TYPE);

-- Defines the Normalized Projection Coordinate System.
```

NPC_TYPE

```
type NPC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of a coordinate in the Normalized
-- Projection Coordinate System.
```

OFF_ON

```
type OFF_ON is (OFF,
                ON);

-- Generic type for off/on switches.
```

OPEN_STRUCTURE_STATUS

```
type OPEN_STRUCTURE_STATUS is (NONE,
                               OPEN);

-- Returns the status of the open structure.
```

OPERATING_MODE

```
type OPERATING_MODE is (REQUEST_MODE,
                        SAMPLE_MODE,
                        EVENT_MODE);

-- Defines the operating modes of an input device.
```

PATH_DEPTH

```
subtype PATH_DEPTH is
        PHIGS_NATURAL range 0..MAX_PATH_DEPTH_SUPPORTED;

-- Defines a type for specifying the depth of a
-- traversal path.
```

PATH_ORDER

```
type PATH_ORDER is (TOPFIRST,
                    BOTTOMFIRST);

-- Provides for specifying the order of the pick and
-- reference paths.
```

PATTERN_INDEX

```
subtype PATTERN_INDEX is
        STYLE_INDEX range 1..STYLE_INDEX'LAST;

-- Defines the range of pattern table indices.
```

PATTERN_INDICES

```
package PATTERN_INDICES is
        new PHIGS_LIST_UTILITIES (PATTERN_INDEX,
                                  MAX_PATTERN_INDICES_SUPPORTED);
-- Provides for lists of pattern table indices.
```

PHIGS_INTEGER

```
type PHIGS_INTEGER is new INTEGER;

-- Base type for PHIGS integer types.
```

PHIGS_NATURAL

```
subtype PHIGS_NATURAL is
        PHIGS_INTEGER range 0..PHIGS_INTEGER'last;

-- Base type for PHIGS natural types.
```

PHIGS_POSITIVE

```
subtype PHIGS_POSITIVE is
        PHIGS_INTEGER range 1..PHIGS_INTEGER'last;

-- Base type for PHIGS positive types.
```

PHIGS_STRING

```
type PHIGS_STRING is
      array (PHIGS_INTEGER range <>) of CHARACTER;

-- Base type for PHIGS string types.
```

PICK_DEVICE_NUMBER

```
type PICK_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for pick device identifiers.
```

PICK_ID

```
type PICK_ID is new PHIGS_INTEGER;

-- Defincs thc range of pick identifiers available on an
-- implementation.
```

PICK_IDS

```
package PICK_IDS is new PHIGS_LIST_UTILITIES (PICK_ID,
                                              MAX_PICK_IDS_SUPPORTED);

-- Provides for a list of pick identifiers.
```

PICK_PATH

```
type PICK_PATH (DEPTH : PATH_DEPTH := 0) is
        record
          PATH : PICK_PATH_ARRAY (0..DEPTH);
        end record;

-- Provides for the specification of a pick path.
```

PICK_PATH_ARRAY

```
type PICK_PATH_ARRAY is array (PATH_DEPTH range < >) of
                                    PICK_PATH_ENTRY;

-- Provides for the specification of an unconstrained array of pick
-- path entries.
```

PICK_PATH_ENTRY

```
type PICK_PATH_ENTRY is
        record
          STRUCTURE_IDENTIFIER  : STRUCTURE_ID;
          PICK_IDENTIFIER       : PICK_ID;
          ELEMENT_NUMBER        : ELEMENT_POSITION;
        end record;

-- Defines the entries of a pick path.
```

PICK_PROMPT_ECHO_TYPE

```
type PICK_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the pick prompt and echo type.
```

PICK_PROMPT_ECHO_TYPES

```
package PICK_PROMPT_ECHO_TYPES is
          new PHIGS_LIST_UTILITIES (PICK_PROMPT_ECHO_TYPE,
                                    MAX_PICK_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of pick prompt and echo types.
```

PICK_REQUEST_STATUS

```
type PICK_REQUEST_STATUS is (OK,
                             NOPICK,
                             NONE);

-- Defines the status of a pick input operation for the
-- request function.
```

PICK_STATUS

```
subtype PICK_STATUS is PICK_REQUEST_STATUS range OK..NOPICK;

-- Defines the status of pick input operation for the sample
-- get, and inquiry functions.
```

POLYLINE_FILL_AREA_CONTROL_FLAG

```
type POLYLINE_FILL_AREA_CONTROL_FLAG is (POLYLINE,
                                         FILL_AREA,
                                         FILL_AREA_SET);

-- Provides for indicating the type of primitive used
-- to produce locator prompt and echo type 5.
```

POLYLINE_INDEX

```
type POLYLINE_INDEX is new PHIGS_POSITIVE;

-- Defines the range of polyline indices.
```

POLYLINE_INDICES

```
package POLYLINE_INDICES is
        new PHIGS_LIST_UTILITIES (POLYLINE_INDEX,
                                  MAX_POLYLINE_INDICES_SUPPORTED);

-- Provides for lists of polyline indices.
```

POLYMARKER_INDEX

```
type POLYMARKER_INDEX is new PHIGS_POSITIVE;

-- Defines the range of polymarker bundle table indices.
```

POLYMARKER_INDICES

```
package POLYMARKER_INDICES is
        new PHIGS_LIST_UTILITIES (POLYMARKER_INDEX,
                                  MAX_POLYMARKER_INDICES_SUPPORTED);

-- Provides for lists of polymarker indices.
```

POSITIVE_VIEW_INDEX

```
subtype POSITIVE_VIEW_INDEX is
        VIEW_INDEX range 1..VIEW_INDEX'LAST;

-- Defines a data type for specifying settable view representations.
```

POSTED_STRUCTURE

```
type POSTED_STRUCTURE is
     record
       STRUCTURE_IDENTIFIER  : STRUCTURE_ID;
       PRIORITY              : DISPLAY_PRIORITY;
     end record;

-- Defines a data type for describing posted structure networks.
```

POSTED_STRUCTURES

```
package POSTED_STRUCTURES is
        new PHIGS_LIST_UTILITIES (POSTED_STRUCTURE,
                                  MAX_POSTED_STRUCTURES_SUPPORTED);

-- Provides for the specification of lists of posted structure
-- networks.
```

PROJECTION_TYPE

```
type PROJECTION_TYPE is (PARALLEL,
                         PERSPECTIVE);

-- Defines the type of projections supported by PHIGS.
```

RASTER_UNITS

```
type RASTER_UNITS is new PHIGS_POSITIVE;

-- Defines the range of raster units.
```

RASTER_UNIT_SIZE_3

```
type RASTER_UNIT_SIZE_3 is
        record
          X : RASTER_UNITS;
          Y : RASTER_UNITS;
          Z : RASTER_UNITS;
        end record;

-- Defines the size of an object in raster units on a
-- raster device.
```

RASTER_UNIT_SIZE_2

```
type RASTER_UNIT_SIZE_2 is
        record
          X : RASTER_UNITS;
          Y : RASTER_UNITS;
        end record;

-- Defines the size of an object in raster units on a
-- raster device.
```

REFERENCE_HANDLING_FLAG

```
type REFERENCE_HANDLING_FLAG is (DELETE,
                                 KEEP);

-- Provides for specification of the handling of deleted
-- structure networks.
```

REFERENCE_PATH

```
type REFERENCE_PATH (DEPTH : PATH_DEPTH := 0) is
        record
          PATH : REFERENCE_PATH_ARRAY (0..DEPTH);
        end record;

-- Provides for the specification of a variable size
-- reference path.
```

REFERENCE_PATH_ARRAY

```
type REFERENCE_PATH_ARRAY is array (PATH_DEPTH range <>) of
                                 REFERENCE_PATH_ENTRY;

-- Provides for the specification of a basic reference path.
```

REFERENCE_PATH_ENTRY

```
type REFERENCE_PATH_ENTRY is
        record
          STRUCTURE_IDENTIFIER  : STRUCTURE_ID;
          ELEMENT_NUMBER        : ELEMENT_POSITION;
        end record;

-- Defines the entries of a reference path.
```

REFERENCE_PATHS

```
package REFERENCE_PATHS is
            new PHIGS_LIST_UTILITIES (REFERENCE_PATH,
                                      MAX_REFERENCE_PATHS_SUPPORTED);

-- Provides for the specification of lists of reference
-- paths.
```

RELATIVE_PRIORITY

```
type RELATIVE_PRIORITY is (HIGHER,
                           LOWER);

-- Indicates the relative input priority between two views.
```

RETURNED_ELEMENT_POSITION

```
subtype RETURNED_ELEMENT_POSITION is
            ELEMENT_POSITION range 0..ELEMENT_POSITION'LAST;

-- Defines a type for returning element pointer values.
```

RETURN_VALUE_TYPE

```
type RETURN_VALUE_TYPE is (SET,
                           REALIZED);

-- Indicates whether the returned values should be as they
-- were set by the program or as they were actually realized
-- on the device.
```

SCALE_FACTOR

```
type SCALE_FACTOR is digits PHIGS_PRECISION;

-- The type used for unitless scaling factors.
```

SEARCH_DIRECTION

```
type SEARCH_DIRECTION is (BACKWARD,
                          FORWARD);

-- This type provides for indicating search direction.
```

SEARCH_STATUS_INDICATOR

```
type SEARCH_STATUS_INDICATOR is (FAILURE,
                                 SUCCESS);

-- This type provides for indicating the result of a search.
```

SMALL_NATURAL

```
subtype SMALL_NATURAL is
        PHIGS_NATURAL range 0..MAX_SMALL_NATURAL;

-- This is a subtype declaration which allows for
-- unconstrained record objects for various record types
-- without causing the exception STORAGE_ERROR to be raised.
```

STRING_DEVICE_NUMBER

```
type STRING_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for string device identifiers.
```

STRING_PROMPT_ECHO_TYPE

```
type STRING_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the string prompt and echo types.
```

STRING_PROMPT_ECHO_TYPES

```
package STRING_PROMPT_ECHO_TYPES is
        new PHIGS_LIST_UTILITIES
                (STRING_PROMPT_ECHO_TYPE,
                 MAX_STRING_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of string prompt and echo types.
```

STRING_SMALL_NATURAL

```
subtype STRING_SMALL_NATURAL is
        PHIGS_NATURAL range 0..MAX_STRING_SMALL_NATURAL;

-- This is a subtype declaration which allows for unconstrained
-- record objects for various string record types without
-- causing the exception STORAGE_ERROR to be raised.
```

STROKE_DEVICE_NUMBER

```
type STROKE_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for stroke device identifiers.
```

STROKE_PROMPT_ECHO_TYPE

```
type STROKE_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the stroke prompt and echo types.
```

STROKE_PROMPT_ECHO_TYPES

```
package STROKE_PROMPT_ECHO_TYPES is
        new PHIGS_LIST_UTILITIES
                (STROKE_PROMPT_ECHO_TYPE
                 MAX_STROKE_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of stroke prompt and echo types
```

STRUCTURE_ELEMENT_RECORD

```
type STRUCTURE_ELEMENT_RECORD
        (ELEMENT_TYPE : STRUCTURE_ELEMENT_TYPE := NIL) is
    record
      case ELEMENT_TYPE is

        -- The empty element

        when NIL = >
          null;

        -- Primitive elements

        when POLYLINE_3 = >
          POLYLINE_3_POINTS : MC.ACCESS_POINT_LIST_3;
```

```
when POLYLINE = >
   POLYLINE_POINTS : MC.ACCESS_POINT_LIST_2;

when POLYMARKER_3 = >
   POLYMARKER_3_POINTS : MC.ACCESS_POINT_LIST_3;

when POLYMARKER = >
   POLYMARKER_POINTS : MC.ACCESS_POINT_LIST_2;

when TEXT_3 = >
   TEXT_3_POINT : MC.POINT_3;
   TEXT_DIRECTION_VECTORS : MC.VECTOR_PAIR_3;
   TEXT_3_CHAR_STRING : ACCESS_STRING;

when TEXT = >
   TEXT_POINT : MC.POINT_2;
   TEXT_CHAR_STRING : ACCESS_STRING;

when ANNOTATION_TEXT_RELATIVE_3 = >
   ANNOTATION_TEXT_RELATIVE_3_REF_POINT : MC.POINT_3;
   ANNOTATION_TEXT_RELATIVE_3_OFFSET : NPC.POINT_3;
   ANNOTATION_TEXT_3_CHAR_STRING : ACCESS_STRING;

when ANNOTATION_TEXT_RELATIVE = >
   ANNOTATION_TEXT_RELATIVE_REF_POINT : MC.POINT_2;
   ANNOTATION_TEXT_RELATIVE_OFFSET : NPC.POINT_2;
   ANNOTATION_TEXT_RELATIVE_CHAR_STRING : ACCESS_STRING;

when FILL_AREA_3 = >
   FILL_AREA_3_POINTS : MC.ACCESS_POINT_LIST_3;

when FILL_AREA = >
   FILL_AREA_POINTS : MC.ACCESS_POINT_LIST_2;

when FILL_AREA_SET_3 = >
   FILL_AREA_SET_3_POINT_LISTS : MC.ACCESS_LIST_OF_POINT_LIST_3;

when FILL_AREA_SET = >
   FILL_AREA_SET_POINT_LISTS : MC.ACCESS_LIST_OF_POINT_LIST_2;

when CELL_ARRAY_3 = >
   CORNER_P_3 : MC.POINT_3;
   CORNER_Q_3 : MC.POINT_3;
   CORNER_R_3 : MC.POINT_3;
   CELL_ARRAY_3_CELLS : ACCESS_COLOUR_MATRIX;

when CELL_ARRAY = >
   CORNER_P : MC.POINT_2;
   CORNER_Q : MC.POINT_2;
   CELL_ARRAY_CELLS : ACCESS_COLOUR_MATRIX;

when GDP_3 = >
   GDP_3_POINTS : MC.ACCESS_POINT_LIST_3;
   GDP_3_DATA : GDP_3_RECORD;

when GDP = >
   GDP_POINTS : MC.ACCESS_POINT_LIST_2;
   GDP_DATA : GDP_RECORD;
```

```
-- Bundle index elements

when SET_POLYLINE_INDEX = >
   POLYLINE_IND : POLYLINE_INDEX;

when SET_POLYMARKER_INDEX = >
   POLYMARKER_IND : POLYMARKER_INDEX;

when SET_TEXT_INDEX = >
   TEXT_IND : TEXT_INDEX;

when SET_INTERIOR_INDEX = >
   INTERIOR_IND : INTERIOR_INDEX;

when SET_EDGE_INDEX = >
   EDGE_IND : EDGE_INDEX;

-- Individual aspect elements

when SET_LINETYPE = >
   TYPE_OF_LINE : LINETYPE;

when SET_LINEWIDTH_SCALE_FACTOR = >
   LINEWIDTH_SF : LINEWIDTH;

when SET_POLYLINE_COLOUR_INDEX = >
   LINE_COLOUR : COLOUR_INDEX;

when SET_MARKER_TYPE = >
   TYPE_OF_MARKER : MARKER_TYPE;

when SET_MARKER_SIZE_SCALE_FACTOR = >
   SIZE : MARKER_SIZE;

when SET_POLYMARKER_COLOUR_INDEX = >
   MARKER_COLOUR : COLOUR_INDEX;

when SET_TEXT_FONT = >
   FONT : TEXT_FONT;

when SET_TEXT_PRECISION = >
   PRECISION : TEXT_PRECISION;

when SET_CHAR_EXPANSION_FACTOR = >
   EXPANSION : CHAR_EXPANSION;

when SET_CHAR_SPACING = >
   SPACING : CHAR_SPACING;

when SET_TEXT_COLOUR_INDEX = >
   TEXT_COLOUR : COLOUR_INDEX;

when SET_CHAR_HEIGHT = >
   HEIGHT : MC.MAGNITUDE;

when SET_CHAR_UP_VECTOR = >
   CHAR_UP_VECTOR : MC.VECTOR_2;
```

```
when SET_TEXT_PATH = >
   PATH : TEXT_PATH;

when SET_TEXT_ALIGNMENT = >
   ALIGNMENT : TEXT_ALIGNMENT;

when SET_ANNOTATION_TEXT_CHAR_HEIGHT = >
   ANNOTATION_HEIGHT : NPC.MAGNITUDE;

when SET_ANNOTATION_TEXT_CHAR_UP_VECTOR = >
   ANNOTATION_CHAR_UP_VECTOR : NPC.VECTOR_2;

when SET_ANNOTATION_TEXT_PATH = >
   ANNOTATION_PATH : TEXT_PATH;

when SET_ANNOTATION_TEXT_ALIGNMENT = >
   ANNOTATION_ALIGNMENT : TEXT_ALIGNMENT;

when SET_ANNOTATION_STYLE = >
   STYLE_OF_ANNOTATION : ANNOTATION_STYLE;

when SET_INTERIOR_STYLE = >
   STYLE_OF_INTERIOR : INTERIOR_STYLE;

when SET_INTERIOR_STYLE_INDEX = >
   STYLE_IND : STYLE_INDEX;

when SET_INTERIOR_COLOUR_INDEX = >
   INTERIOR_COLOUR : COLOUR_INDEX;

when SET_EDGE_FLAG = >
   FLAG : EDGE_FLAG;

when SET_EDGETYPE = >
   TYPE_OF_EDGE : EDGETYPE;

when SET_EDGEWIDTH_SCALE_FACTOR = >
   EDGEWIDTH_SF : EDGEWIDTH;

when SET_EDGE_COLOUR_INDEX = >
   EDGE_COLOUR : COLOUR_INDEX;

-- Pattern attribute elements

when SET_PATTERN_SIZE = >
   PATTERN_SIZE : MC.SIZE_2;

when SET_PATTERN_REFERENCE_POINT_AND_VECTORS = >
   PATTERN_REFERENCE_POINT_3 : MC.POINT_3;
   PATTERN_REFERENCE_VECTORS : MC.VECTOR_PAIR_3;

when SET_PATTERN_REFERENCE_POINT = >
   PATTERN_REFERENCE_POINT : MC.POINT_2;

-- Name set elements

when ADD_NAMES_TO_SET = >
   NAMES_TO_ADD : NAME_SET;
```

```
when REMOVE_NAMES_FROM_SET = >
   NAMES_TO_REMOVE : NAME_SET;

-- ASF elements

when SET_INDIVIDUAL_ASF = >
   ATTRIBUTE_ID : ASPECT;
   SOURCE_FLAG : ASF;

-- HLHSR elements

when SET_HLHSR_IDENTIFIER = >
   HLHSR_IDENTIFIER : HLHSR_ID;

-- Transformation elements

when SET_LOCAL_TRANSFORMATION_3 = >
   LOCAL_MATRIX_3 : TRANSFORMATION_MATRIX_3;
   HOW_APPLIED_3 : COMPOSITION_TYPE;

when SET_LOCAL_TRANSFORMATION = >
   LOCAL_MATRIX : TRANSFORMATION_MATRIX_2;
   HOW_APPLIED : COMPOSITION_TYPE;

when SET_GLOBAL_TRANSFORMATION_3 = >
   GLOBAL_MATRIX_3 : TRANSFORMATION_MATRIX_3;

when SET_GLOBAL_TRANSFORMATION = >
   GLOBAL_MATRIX : TRANSFORMATION_MATRIX_2;

when SET_MODELLING_CLIPPING_VOLUME_3 = >
   MODELLING_CLIPPING_OPERATOR_3 : MODELLING_CLIP_OPERATION_TYPE;
   MODELLING_CLIPPING_LIMITS_3 : MC.ACCESS_HALF_SPACE_LIST_3;

when SET_MODELLING_CLIPPING_VOLUME = >
   MODELLING_CLIPPING_OPERATOR : MODELLING_CLIP_OPERATION_TYPE;
   MODELLING_CLIPPING_LIMITS : MC.ACCESS_HALF_SPACE_LIST_2;

when SET_MODELLING_CLIPPING_INDICATOR = >
   MODELLING_CLIPPING_INDICATOR : CLIPPING_INDICATOR;

when RESTORE_MODELLING_CLIPPING_VOLUME = >
   null;

when SET_VIEW_INDEX = >
   VIEW_IND : VIEW_INDEX;

-- Invocation elements

when EXECUTE_STRUCTURE = >
   STRUCTURE_IDENTIFIER : STRUCTURE_ID;

-- Structure content identification elements

when LABEL = >
   LABEL_IDENTIFIER : LABEL_ID;

when APPLICATION_DATA = >
   DATA : APPLICATION_DATA_RECORD;
```

```
                when GSE = >
                   GSE_DATA : GSE_RECORD;

                when SET_PICK_IDENTIFIER = >
                   PICK_IDENTIFIER : PICK_ID;

             end case;
          end record;

      -- This type defines the format of structure contents and
      -- is used to return structure elements during inquiry.
```

STRUCTURE_ELEMENT_TYPE

```
      subtype STRUCTURE_ELEMENT_TYPE is
                ELEMENT_TYPE range NIL..ELEMENT_TYPE'LAST;

      -- Provides for identifying the types of individual
      -- structure elements.
```

STRUCTURE_ID

```
      type STRUCTURE_ID is new PHIGS_NATURAL;

      -- Defines a type for structure identifiers.
```

STRUCTURE_IDS

```
      package STRUCTURE_IDS is
                new PHIGS_LIST_UTILITIES (STRUCTURE_ID,
                                          MAX_STRUCTURE_IDS_SUPPORTED);

      -- Provides for lists of structure identifiers.
```

STRUCTURE NETWORK SOURCE

```
      type STRUCTURE_NETWORK_SOURCE is (CSS,
                                        ARCHIVE);

      -- Provides for indicating source for check of conflicting
      -- structure identifiers.
```

STRUCTURE_STATE

```
      type STRUCTURE_STATE is (STCL,
                               STOP);

      -- The type used to return the structure state value.
```

STRUCTURE_STATUS

```
type STRUCTURE_STATUS is (NON_EXISTENT,
                          EMPTY,
                          NOTEMPTY);

-- The type used to return the structure status.
```

STYLE_INDEX

```
type STYLE_INDEX is new PHIGS_INTEGER;

-- Defines an interior style index.
```

SUBPROGRAM_NAME

```
subtype SUBPROGRAM_NAME is PHIGS_STRING;

-- Defines the name of a PHIGS function detecting an error.
```

SYSTEM_STATE

```
type SYSTEM_STATE is (PHCL,
                      PHOP);

-- The type used to return the system state value.
```

TEXT_ALIGNMENT

```
type TEXT_ALIGNMENT is
        record
          HORIZONTAL : HORIZONTAL_ALIGNMENT;
          VERTICAL   : VERTICAL_ALIGNMENT;
        end record;

-- The type of the attribute controlling the positioning of
-- the text extent parallelogram in relation to the text
-- position, having horizontal and vertical components as
-- defined above.
```

TEXT_FONT

```
type TEXT_FONT is new PHIGS_INTEGER;

-- Defines the types of fonts provided by the implementation.
```

TEXT_FONT_PRECISION

```
type TEXT_FONT_PRECISION is
        record
          FONT          : TEXT_FONT;
          PRECISION     : TEXT_PRECISION;
        end record;

-- This type defines a record describing the text font and
-- precision supported.
```

TEXT_FONT_PRECISIONS

```
package TEXT_FONT_PRECISIONS is
            new PHIGS_LIST_UTILITIES
                  (TEXT_FONT_PRECISION,
                   MAX_TEXT_FONT_PRECISION_PAIRS_SUPPORTED);

-- Provides for lists of text font and precision pairs.
```

TEXT_INDEX

```
type TEXT_INDEX is new PHIGS_POSITIVE;

-- Defines the range of text bundle table indices.
```

TEXT_INDICES

```
package TEXT_INDICES is
            new PHIGS_LIST_UTILITIES (TEXT_INDEX,
                                      MAX_TEXT_INDICES_SUPPORTED);

-- Provides for lists of text indices.
```

TEXT_PATH

```
type TEXT_PATH is (RIGHT,
                   LEFT,
                   UP,
                   DOWN);

-- The direction taken by a text string.
```

TEXT_PRECISION

```
type TEXT_PRECISION is (STRING_PRECISION,
                        CHAR_PRECISION,
                        STROKE_PRECISION);

-- The precision with which text appears.
```

TRANSFORMATION_MATRIX_2

```
type TRANSFORMATION_MATRIX_2 is
      array (1..3, 1..3) of MC_TYPE;

-- For 2D modelling and view transformation matrices.
```

TRANSFORMATION_MATRIX_3

```
type TRANSFORMATION_MATRIX_3 is
      array (1..4, 1..4) of MC_TYPE;

-- For 3D modelling and view transformation matrices.
```

UPDATE_REGENERATION_FLAG

```
type UPDATE_REGENERATION_FLAG is (PERFORM,
                                  POSTPONE);

-- Flag indicating regeneration action on display.
```

UPDATE_STATE

```
type UPDATE_STATE is (NOTPENDING,
                      PENDING);

-- Indicates whether or not a workstation transformation
-- change has been requested and not yet provided.
```

VALUATOR_DEVICE_NUMBER

```
type VALUATOR_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for valuator device identifiers.
```

VALUATOR_PROMPT_ECHO_TYPE

```
type VALUATOR_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the possible range of valuator prompt and
-- echo types.
```

VALUATOR_PROMPT_ECHO_TYPES

```
package VALUATOR_PROMPT_ECHO_TYPES is
        new PHIGS_LIST_UTILITIES
                (VALUATOR_PROMPT_ECHO_TYPE,
                 MAX_VALUATOR_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of valuator prompt and echo types.
```

VALUATOR_VALUE

```
type VALUATOR_VALUE is digits PHIGS_PRECISION;

-- Defines the range of accuracy of valuator input values
-- on an implementation.
```

VARIABLE_CONNECTION_ID

```
type VARIABLE_CONNECTION_ID(LENGTH : STRING_SMALL_NATURAL := 0) is
        record
          CONNECT : CONNECTION_ID (1..LENGTH);
        end record;

-- Defines a variable length connection identifier for
-- INQ_WS_CONNECTION_AND_TYPE.
```

VARIABLE_FILE_ID

```
type VARIABLE_FILE_ID (LENGTH : FILE_SMALL_NATURAL := 0) is
        record
          NAME : FILE_ID (1..LENGTH);
        end record;

-- Provides for variable length file identifiers.
```

VARIABLE_FILE_IDS

```
package VARIABLE_FILE_IDS is
        new PHIGS_LIST_UTILITIES(VARIABLE_FILE_ID,
                                 MAX_FILE_IDS_SUPPORTED);

-- This type is used to define lists of file identifiers.
```

VERTICAL_ALIGNMENT

```
type VERTICAL_ALIGNMENT is (NORMAL,
                            TOP,
                            CAP,
                            HALF,
                            BASE,
                            BOTTOM);

-- The alignment of the text extent parallelogram with
-- respect to the vertical positioning of the text.
```

VIEW_INDEX

```
type VIEW_INDEX is new PHIGS_NATURAL;

-- Defines a type for specifying view indices.
```

VIEW_INDICES

```
package VIEW_INDICES is
        new PHIGS_LIST_UTILITIES (VIEW_INDEX,
                                  MAX_VIEW_INDICES_SUPPORTED);

-- Provides for lists of view indices.
```

VISUAL_REPRESENTATION_STATE

```
type VISUAL_REPRESENTATION_STATE is (CORRECT,
                                     DEFERRED,
                                     SIMULATED);

-- Specifies state of image on display surface.
```

VRC

```
package VRC is new PHIGS_COORDINATE_SYSTEM (VRC_TYPE);

-- Defines the view reference coordinate system.
```

VRC_TYPE

```
type VRC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of view reference coordinate components.
```

WC

```
package WC is new PHIGS_COORDINATE_SYSTEM (WC_TYPE);

-- Defines the world coordinate system.
```

WC_TYPE

```
type WC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of coordinate in the World Coordinate
-- System.
```

WS_CATEGORY

```
type WS_CATEGORY is (OUTPUT,
                     INPUT,
                     OUTIN,
                     MO,
                     MI);

-- Type for PHIGS workstation categories.
```

WS_DEPENDENCY

```
type WS_DEPENDENCY is (WS_INDEPENDENT,
                       WS_DEPENDENT);

-- Type for PHIGS workstation categories.
```

WS_DEPENDENCIES

```
package WS_DEPENDENCIES is
        new PHIGS_LIST_UTILITIES (WS_DEPENDENCY,
                                  MAX_GSE_IDS_SUPPORTED);

-- Provides for lists of workstation dependencies.
```

WS_ID

```
type WS_ID is new PHIGS_POSITIVE;

-- Defines the range of workstation identifiers.
```

WS_IDS

```
package WS_IDS is new PHIGS_LIST_UTILITIES (WS_ID,
                                            MAX_OPEN_WS_SUPPORTED);

-- Provides for lists of workstation identifiers.
```

WS_STATE

```
type WS_STATE is (WSCL,
                  WSOP);

-- The type used to return the workstation state value.
```

WS_TYPE

```
type WS_TYPE is new PHIGS_POSITIVE;

-- Range of values corresponding to valid workstation types.
-- Constants specifying names for the various types of
-- workstations should be provided by an implementation.
```

WS_TYPES

```
package WS_TYPES is new PHIGS_LIST_UTILITIES (WS_TYPE,
                                              MAX_WS_TYPES_SUPPORTED);

-- Provides for lists of workstation types.
```

4.2.3 Alphabetical List of Private Type Definitions

This section contains an alphabetical listing of all of the private type definitions used to define the PHIGS binding to Ada. All of these elements are Ada PRIVATE type declarations. These declarations except for NAME_SET are included in the PHIGS package to facilitate the manipulations of the private types. The NAME_SET private type is declared as part of the PHIGS_NAME_SET_FACILITY package.

CHOICE_DATA_RECORD

```
type CHOICE_DATA_RECORD
        (PROMPT_ECHO_TYPE : CHOICE_PROMPT_ECHO_TYPE := 1) is private;

-- Defines a record for initialising choice input.  The
-- structure of the record is implementation-defined.  Since
-- it is a private type, the components of the record may be
-- retrieved only through the use of the subprograms for
-- manipulating the input data records (5.14.1).
```

LOCATOR_DATA_RECORD

```
type LOCATOR_DATA_RECORD
        (PROMPT_ECHO_TYPE : LOCATOR_PROMPT_ECHO_TYPE := 1) is private;

-- Defines a locator input data record.
```

METAFILE_DATA_RECORD

```
type METAFILE_DATA_RECORD
        (TYPE_OF_ITEM : METAFILE_ITEM_TYPE := 0
         LENGTH       : NATURAL := 0) is private;

-- A data record for metafiles.
```

NAME_SET

```
type NAME_SET is private;

-- Defines a name set data element.
```

PICK_DATA_RECORD

```
type PICK_DATA_RECORD
        (PROMPT_ECHO_TYPE : PICK_PROMPT_ECHO_TYPE := 1) is private;

-- Defines a record for initialising pick input.  The
-- structure of the record is implementation-defined.  Since
-- it is a private type, the components of the record may be
-- retrieved only through the use of the subprograms for
-- manipulating the input data records ( 5.14.1).
```

```
--------------------------------------------------------------------------------------------------------------------

STRING_DATA_RECORD

        type STRING_DATA_RECORD
                (PROMPT_ECHO_TYPE : STRING_PROMPT_ECHO_TYPE := 1) is private;

        -- Defines a string input data record.

--------------------------------------------------------------------------------------------------------------------

STROKE_DATA_RECORD

        type STROKE_DATA_RECORD
                (PROMPT_ECHO_TYPE : STROKE_PROMPT_ECHO_TYPE := 1) is private;

        -- Defines a stroke input data record.

--------------------------------------------------------------------------------------------------------------------

VALUATOR_DATA_RECORD

        type VALUATOR_DATA_RECORD
                (PROMPT_ECHO_TYPE : VALUATOR_PROMPT_ECHO_TYPE := 1) is private;

        -- Defines a valuator input data record.
```

4.2.4 List of Constant Declarations

This section contains the declarations of implementation dependent constants for defining PHIGS/Ada types. Some of the constants are used for defining default parameter values for PHIGS procedures defined in 5.0. This section also contains the constants that provide the PHIGS standard values defined for some PHIGS/Ada types. The following constants define the PHIGS standard linetypes:

```
SOLID_LINE                  : constant LINETYPE := 1;
DASHED_LINE                 : constant LINETYPE := 2;
DOTTED_LINE                 : constant LINETYPE := 3;
DASHED_DOTTED_LINE          : constant LINETYPE := 4;
```

The following constants define the PHIGS standard marker types:

```
DOT_MARKER                  : constant MARKER_TYPE := 1;
PLUS_MARKER                 : constant MARKER_TYPE := 2;
STAR_MARKER                 : constant MARKER_TYPE := 3;
ZERO_MARKER                 : constant MARKER_TYPE := 4;
X_MARKER                    : constant MARKER_TYPE := 5;
```

The following constants define the PHIGS standard edgetypes:

```
SOLID_EDGE                  : constant EDGETYPE := 1;
DASHED_EDGE                 : constant EDGETYPE := 2;
DOTTED_EDGE                 : constant EDGETYPE := 3;
DASHED_DOTTED_EDGE          : constant EDGETYPE := 4;
```

The following constants define the PHIGS standard annotation styles:

```
UNCONNECTED_ANNOTATION      : constant ANNOTATION_STYLE := 1;
LEAD_LINE_ANNOTATION        : constant ANNOTATION_STYLE := 2;
```

The following constants define the colour models specified by PHIGS:

```
RGB                         : constant COLOUR_MODEL := 1;
CIELUV                      : constant COLOUR_MODEL := 2;
HSV                         : constant COLOUR_MODEL := 3;
HLS                         : constant COLOUR_MODEL := 4;
```

The following constants define the prompt and echo types specified by PHIGS:

```
DEFAULT_LOCATOR             : constant LOCATOR_PROMPT_ECHO_TYPE := 1;
CROSS_HAIR_LOCATOR          : constant LOCATOR_PROMPT_ECHO_TYPE := 2;
TRACKING_CROSS_LOCATOR      : constant LOCATOR_PROMPT_ECHO_TYPE := 3;
RUBBER_BAND_LINE_LOCATOR    : constant LOCATOR_PROMPT_ECHO_TYPE := 4;
RECTANGLE_LOCATOR           : constant LOCATOR_PROMPT_ECHO_TYPE := 5;
DIGITAL_LOCATOR             : constant LOCATOR_PROMPT_ECHO_TYPE := 6;

DEFAULT_STROKE              : constant STROKE_PROMPT_ECHO_TYPE := 1;
DIGITAL_STROKE              : constant STROKE_PROMPT_ECHO_TYPE := 2;
MARKER_STROKE               : constant STROKE_PROMPT_ECHO_TYPE := 3;
LINE_STROKE                 : constant STROKE_PROMPT_ECHO_TYPE := 4;

DEFAULT_VALUATOR            : constant VALUATOR_PROMPT_ECHO_TYPE := 1;
GRAPHICAL_VALUATOR          : constant VALUATOR_PROMPT_ECHO_TYPE := 2;
DIGITAL_VALUATOR            : constant VALUATOR_PROMPT_ECHO_TYPE := 3;

DEFAULT_CHOICE              : constant CHOICE_PROMPT_ECHO_TYPE := 1;
PROMPT_ECHO_CHOICE          : constant CHOICE_PROMPT_ECHO_TYPE := 2;
STRING_PROMPT_CHOICE        : constant CHOICE_PROMPT_ECHO_TYPE := 3;
STRING_INPUT_CHOICE         : constant CHOICE_PROMPT_ECHO_TYPE := 4;
STRUCTURE_CHOICE            : constant CHOICE_PROMPT_ECHO_TYPE := 5;

DEFAULT_PICK                : constant PICK_PROMPT_ECHO_TYPE := 1;
GROUP_HIGHLIGHT_PICK        : constant PICK_PROMPT_ECHO_TYPE := 2;
STRUCTURE_HIGHLIGHT_PICK    : constant PICK_PROMPT_ECHO_TYPE := 3;
```

The following constants are used for defining the modelling clipping operators specified by PHIGS:

```
REPLACE_VOLUME              : constant MODELLING_CLIP_OPERATION_TYPE := 1;
INTERSECT_VOLUME            : constant MODELLING_CLIP_OPERATION_TYPE := 2;
```

4.2.5 PHIGS Configuration Values

To facilitate the configuration of PHIGS implementations, all implementation-dependent values are specified in package PHIGS_CONFIGURATION. When this package is **with**ed, these values become accessible. The names by which the values are accessed shall be contained within the visible part of package PHIGS_CONFIGURATION. There are three kinds of values which may be represented in this package:

- PHIGS base type characteristics
- The name of the default error file
- Range limits for various PHIGS types.

The precision of the PHIGS floating point base types is a named number defined in the visible portion of the PHIGS_CONFIGURATION package specification:

PHIGS_PRECISION

The range limits of PHIGS integer base types are obtained from names which specify the largest negative and largest positive integers to be represented. These names are defined in the visible portion of the PHIGS_CONFIGURATION package specification:

MIN_INTEGER_VALUE
MAX_INTEGER_VALUE

The name DEFAULT_ERROR_FILE specifies the default name of the PHIGS error file. This name is a string constant defined in the visible portion of the PHIGS_CONFIGURATION package specification.

The remaining names may be any Ada construct which can represent a NATURAL value. Such constructs include constant declarations, variables, and parameterless functions which return integers. The body of the PHIGS_CONFIGURATION package may be used to compute these values and/or provide initial values. An implementation may permit the application programmer to replace the body of the PHIGS_CONFIGURATION package. Instructions for doing this shall be provided in the implementation documentation, if allowed by an implementation.

The following name is used for defining default parameter values for PHIGS procedures defined in 5.0:

DEFAULT_MEMORY_UNITS

The following name is used for defining the default list size for the MAX_LIST_SIZE parameter of the generic PHIGS_LIST_UTILITIES package:

DEFAULT_LIST_SIZE

The following names are used to define the maximum values for various "..._SMALL_NATURAL" data types defined in the binding:

MAX_CHOICE_SMALL_NATURAL
MAX_COLOUR_MATRIX_SMALL_NATURAL
MAX_FILE_SMALL_NATURAL
MAX_INPUT_STRING_SMALL_NATURAL
MAX_SMALL_NATURAL
MAX_STRING_SMALL_NATURAL

The following name is used to define the maximum length of an application data record:

MAX_APPLICATION_DATA

The following names are used to specify the maximum number of entities of the indicated type which are supported by the implementation:

MAX_ANNOTATION_STYLES_SUPPORTED
MAX_ARCHIVE_IDS_SUPPORTED
MAX_CHAR_SETS_SUPPORTED
MAX_CHOICE_PROMPTS_SUPPORTED
MAX_CHOICE_PROMPT_ECHO_TYPES_SUPPORTED
MAX_COLOUR_INDICES_SUPPORTED
MAX_COLOUR_MODELS_SUPPORTED
MAX_EDGE_INDICES_SUPPORTED
MAX_EDGETYPES_SUPPORTED
MAX_ESCAPE_IDS_SUPPORTED
MAX_FILE_IDS_SUPPORTED

MAX_GDP_3_IDS_SUPPORTED
MAX_GDP_IDS_SUPPORTED
MAX_GSE_IDS_SUPPORTED
MAX_HATCH_STYLES_SUPPORTED
MAX_HLHSR_IDS_SUPPORTED
MAX_HLHSR_MODES_SUPPORTED
MAX_INTERIOR_INDICES_SUPPORTED
MAX_LINETYPES_SUPPORTED
MAX_LOCATOR_PROMPT_ECHO_TYPES_SUPPORTED
MAX_MARKER_TYPES_SUPPORTED
MAX_MODELLING_CLIP_OPERATION_TYPES_SUPPORTED
MAX_MODELLING_CLIP_DISTINCT_PLANES_SUPPORTED
MAX_NAME_SUPPORTED
MAX_NAME_SET_FILTER_LIST_LENGTH_SUPPORTED
MAX_OPEN_WS_SUPPORTED
MAX_PATH_DEPTH_SUPPORTED
MAX_PATTERN_INDICES_SUPPORTED
MAX_PICK_IDS_SUPPORTED
MAX_PICK_PROMPT_ECHO_TYPES_SUPPORTED
MAX_POLYLINE_INDICES_SUPPORTED
MAX_POLYMARKER_INDICES_SUPPORTED
MAX_POSTED_STRUCTURES_SUPPORTED
MAX_REFERENCE_PATHS_SUPPORTED
MAX_STRING_PROMPT_ECHO_TYPES_SUPPORTED
MAX_STROKE_PROMPT_ECHO_TYPES_SUPPORTED
MAX_STRUCTURE_IDS_SUPPORTED
MAX_TEXT_FONT_PRECISION_PAIRS_SUPPORTED
MAX_TEXT_INDICES_SUPPORTED
MAX_VALUATOR_PROMPT_ECHO_TYPES_SUPPORTED
MAX_VIEW_INDICES_SUPPORTED
MAX_WS_IDS_SUPPORTED
MAX_WS_TYPES_SUPPORTED

The following is used to define the maximum memory units which may be allocated:

MAX_MEMORY_UNITS

The following name is used to specify the maximum number of components which can be represented in the generalized colour representation:

MAX_COLOUR_COEFFICIENTS

The following name is used to specify the length of the maximum metafile item which can be handled:

MAX_METAFILE_ITEM_LENGTH

The following names are used to define the range of values acceptable as metafile item types:

MIN_METAFILE_ITEM_TYPE
MAX_METAFILE_ITEM_TYPE

4.3 Error Codes

This binding requires the use of the PHIGS procedure ERROR_HANDLING to process any errors that occur in PHIGS procedures, except the inquiry procedures. A complete description of the error handling requirements of this binding is available in 3.2.3 of this binding.

The PHIGS inquiry functions do not generate errors. Instead, they return an error indicator parameter containing the number of the "error" which was detected. This is consistent with the PHIGS philosophy that no errors occur during inquiries. The error numbers correspond to the error numbers from Appendix C of the PHIGS specification, plus additional errors defined in this binding. Note that certain known error conditions may be detected outside the control of PHIGS due to the nature of the Ada language, and may result in an exception being raised on an inquiry.

4.3.1 Precluded Error Codes

The following PHIGS errors are listed separately because due to some feature of the Ada language or its use by this binding, they could never be detected by the PHIGS implementation. The errors might be detected by the Ada compiler, or at run-time outside the scope of PHIGS.

100 Ignoring function, the bundle index value is less than one.
112 Ignoring function, the pattern index value is less than one.
113 Ignoring function, the colour index value is less than zero.
114 Ignoring function, the view index value is less than zero.
115 Ignoring function, the view index value is less than one.
116 Ignoring function, one of the dimensions of pattern colour array is less than one.
117 Ignoring function, one of the dimensions of the colour index array is less than zero.
118 Ignoring function, one of the components of the colour specification is out of range. The valid range is dependent upon the current colour model.

5 Functions in the Ada Binding of PHIGS

5.1 Control functions

OPEN PHIGS

```
procedure OPEN_PHIGS
  (ERROR_FILE           : in FILE_ID := PHIGS_STRING(DEFAULT_ERROR_FILE);
   AMOUNT_OF_MEMORY     : in PHIGS_NATURAL := DEFAULT_MEMORY_UNITS);
```

CLOSE PHIGS

```
procedure CLOSE_PHIGS;
```

OPEN WORKSTATION

```
procedure OPEN_WS
  (WS              : in WS_ID;
   CONNECTION      : in CONNECTION_ID;
   TYPE_OF_WS      : in WS_TYPE);
```

CLOSE WORKSTATION

```
procedure CLOSE_WS
  (WS : in WS_ID);
```

REDRAW ALL STRUCTURES

```
procedure REDRAW_ALL_STRUCTURES
  (WS     : in WS_ID;
   FLAG   : in CONTROL_FLAG);
```

UPDATE WORKSTATION

```
procedure UPDATE_WS
  (WS                  : in WS_ID;
   REGENERATION        : in UPDATE_REGENERATION_FLAG);
```

SET DISPLAY UPDATE STATE

```
procedure SET_DISPLAY_UPDATE_STATE
  (WS                  : in WS_ID;
   DEFERRAL            : in DEFERRAL_MODE;
   MODIFICATION        : in MODIFICATION_MODE);
```

MESSAGE

```
procedure MESSAGE
  (WS                  : in WS_ID;
   CONTENTS            : in PHIGS_STRING);
```

5.2 Output primitive functions

POLYLINE 3

```
procedure POLYLINE
  (POINTS : in MC.POINT_LIST_3);
```

POLYLINE

```
procedure POLYLINE
  (POINTS : in MC.POINT_LIST_2);
```

POLYMARKER 3

```
procedure POLYMARKER
  (POINTS : in MC.POINT_LIST_3);
```

POLYMARKER

```
procedure POLYMARKER
  (POINTS : in MC.POINT_LIST_2);
```

TEXT 3

```
procedure TEXT
  (POSITION          : in MC.POINT_3;
   DIRECTION_VECTORS : in MC.VECTOR_PAIR_3;
   CHAR_STRING       : in PHIGS_STRING);
```

TEXT

```
procedure TEXT
  (POSITION    : in MC.POINT_2;
   CHAR_STRING : in PHIGS_STRING);
```

ANNOTATION TEXT RELATIVE 3

```
procedure ANNOTATION_TEXT_RELATIVE
  (REFERENCE_POINT   : in MC.POINT_3;
   ANNOTATION_OFFSET : in NPC.POINT_3;
   CHAR_STRING       : in PHIGS_STRING);
```

ANNOTATION TEXT RELATIVE

```
procedure ANNOTATION_TEXT_RELATIVE
  (REFERENCE_POINT   : in MC.POINT_2;
   ANNOTATION_OFFSET : in NPC.POINT_2;
   CHAR_STRING       : in PHIGS_STRING);
```

FILL AREA 3

```
procedure FILL_AREA
  (POINTS : in MC.POINT_LIST_3);
```

FILL AREA

```
procedure FILL_AREA
  (POINTS : in MC.POINT_LIST_2);
```

FILL AREA SET 3

```
procedure FILL_AREA_SET
  (POINT_LISTS : in MC.LIST_OF_POINT_LIST_3);
```

FILL AREA SET

```
procedure FILL_AREA_SET
  (POINT_LISTS : in MC.LIST_OF_POINT_LIST_2);
```

CELL ARRAY 3

```
procedure CELL_ARRAY
  (CORNER_P  : in MC.POINT_3;
   CORNER_Q  : in MC.POINT_3;
   CORNER_R  : in MC.POINT_3;
   CELLS     : in COLOUR_MATRIX);
```

CELL ARRAY

```
procedure CELL_ARRAY
  (CORNER_P  : in MC.POINT_2;
   CORNER_Q  : in MC.POINT_2;
   CELLS     : in COLOUR_MATRIX);
```

GENERALIZED DRAWING PRIMITIVE 3
GENERALIZED DRAWING PRIMITIVE

The Generalized Drawing Primitive (GDP) is bound in a one-to-many fashion, with a separate procedure implemented for each GDP, each with its own parameter interface. GDP names and parameters are registered in the ISO International Register of Graphical Items which is maintained by the Registration Authority. All supported registered GDP's are in a library package named PHIGS_GDP.

The minimal support for GDP's required by this standard is that defined in the following specification for the PHIGS_GDP package. The minimal support provides an interface to registered and unregistered GDP's not supported by the implementation. Unsupported GDP's may be inserted in structures using either form of the procedure GENERALIZED_GDP whose external specifications are shown.

```
with PHIGS_TYPES;
use  PHIGS_TYPES;
package PHIGS_GDP is
   type GDP_INTEGER_ARRAY is array (SMALL_NATURAL range <>) of
                                    INTEGER;
   type GDP_FLOAT_ARRAY   is array (SMALL_NATURAL range <>) of
                                    GENERAL_FLOAT_PARAMETER;
   type GDP_STRING_ARRAY  is array (SMALL_NATURAL range <>) of
                                    PHIGS_STRING(1..80);

   type GDP_DATA_RECORD (NUM_OF_INTEGERS : SMALL_NATURAL := 0;
                         NUM_OF_REALS    : SMALL_NATURAL := 0;
                         NUM_OF_STRINGS  : SMALL_NATURAL := 0) is

      record
         INTEGER_ARRAY : GDP_INTEGER_ARRAY (1..NUM_OF_INTEGERS);
         REAL_ARRAY    : GDP_FLOAT_ARRAY   (1..NUM_OF_REALS);
         GDP_STRINGS   : GDP_STRING_ARRAY  (1..NUM_OF_STRINGS);
      end record;

procedure GENERALIZED_GDP
   (GDP         : in GDP_3_ID;
    POINTS      : in MC.POINT_LIST_3;
    GDP_DATA    : in GDP_DATA_RECORD);

procedure GENERALIZED_GDP
   (GDP         : in GDP_ID;
    POINTS      : in MC.POINT_LIST_2;
    GDP_DATA    : in GDP_DATA_RECORD);

end PHIGS_GDP;
```

Each registered GDP procedure supported by the implementation will be included in package PHIGS_GDP as a separate procedure using the following naming convention which results in references to the GDP by the expression PHIGS_GDP.<name of the GDP procedure>. Specific names will be assigned when the GDP is registered.

```
procedure <name of the GDP procedure> (<parameters as required>);
```

Each unregistered GDP procedure supported by the implementation will be a library package using the following naming convention:

```
package PHIGS_UGDP_<name of the GDP procedure> is
   procedure GDP;
      -- Ada code for UGDP procedure

   -- The only procedure name used in the package
   -- will be GDP.

end PHIGS_UGDP_<name of the GDP procedure>;
```

5.3 Attribute specification functions

SET POLYLINE INDEX

```
procedure SET_POLYLINE_INDEX
  (POLYLINE_IND : in POLYLINE_INDEX);
```

SET POLYMARKER INDEX

```
procedure SET_POLYMARKER_INDEX
  (POLYMARKER_IND : in POLYMARKER_INDEX);
```

SET TEXT INDEX

```
procedure SET_TEXT_INDEX
  (TEXT_IND : in TEXT_INDEX);
```

SET INTERIOR INDEX

```
procedure SET_INTERIOR_INDEX
  (INTERIOR_IND : in INTERIOR_INDEX);
```

SET EDGE INDEX

```
procedure SET_EDGE_INDEX
  (EDGE_IND : in EDGE_INDEX);
```

SET LINETYPE

```
procedure SET_LINETYPE
  (TYPE_OF_LINE : in LINETYPE);
```

SET LINEWIDTH SCALE FACTOR

```
procedure SET_LINEWIDTH_SCALE_FACTOR
  (LINEWIDTH_SF : in LINEWIDTH);
```

SET POLYLINE COLOUR INDEX

```
procedure SET_POLYLINE_COLOUR_INDEX
  (LINE_COLOUR : in COLOUR_INDEX);
```

SET MARKER TYPE

```
procedure SET_MARKER_TYPE
  (TYPE_OF_MARKER : in MARKER_TYPE);
```

SET MARKER SIZE SCALE FACTOR

```
procedure SET_MARKER_SIZE_SCALE_FACTOR
  (SIZE : in MARKER_SIZE);
```

SET POLYMARKER COLOUR INDEX

```
procedure SET_POLYMARKER_COLOUR_INDEX
  (MARKER_COLOUR : in COLOUR_INDEX);
```

SET TEXT FONT

```
procedure SET_TEXT_FONT
  (FONT : in TEXT_FONT);
```

SET TEXT PRECISION

```
procedure SET_TEXT_PRECISION
  (PRECISION : in TEXT_PRECISION);
```

SET CHARACTER EXPANSION FACTOR

```
procedure SET_CHAR_EXPANSION_FACTOR
  (EXPANSION : in CHAR_EXPANSION);
```

SET CHARACTER SPACING

```
procedure SET_CHAR_SPACING
  (SPACING : in CHAR_SPACING);
```

SET TEXT COLOUR INDEX

```
procedure SET_TEXT_COLOUR_INDEX
  (TEXT_COLOUR : in COLOUR_INDEX);
```

SET CHARACTER HEIGHT

```
procedure SET_CHAR_HEIGHT
  (HEIGHT : in MC.MAGNITUDE);
```

SET CHARACTER UP VECTOR

```
procedure SET_CHAR_UP_VECTOR
  (CHAR_UP_VECTOR : in MC.VECTOR_2);
```

SET TEXT PATH

```
procedure SET_TEXT_PATH
  (PATH : in TEXT_PATH);
```

SET TEXT ALIGNMENT

```
procedure SET_TEXT_ALIGNMENT
  (ALIGNMENT : in TEXT_ALIGNMENT);
```

SET ANNOTATION TEXT CHARACTER HEIGHT

```
procedure SET_ANNOTATION_TEXT_CHAR_HEIGHT
  (ANNOTATION_HEIGHT : in NPC.MAGNITUDE);
```

SET ANNOTATION TEXT CHARACTER UP VECTOR

```
procedure SET_ANNOTATION_TEXT_CHAR_UP_VECTOR
  (ANNOTATION_CHAR_UP_VECTOR : in NPC.VECTOR_2);
```

SET ANNOTATION TEXT PATH

```
procedure SET_ANNOTATION_TEXT_PATH
  (ANNOTATION_PATH : in TEXT_PATH);
```

SET ANNOTATION TEXT ALIGNMENT

```
procedure SET_ANNOTATION_TEXT_ALIGNMENT
  (ANNOTATION_ALIGNMENT : in TEXT_ALIGNMENT);
```

SET ANNOTATION STYLE

```
procedure SET_ANNOTATION_STYLE
  (STYLE_OF_ANNOTATION : in ANNOTATION_STYLE);
```

SET INTERIOR STYLE

```
procedure SET_INTERIOR_STYLE
  (STYLE_OF_INTERIOR : in INTERIOR_STYLE);
```

SET INTERIOR STYLE INDEX

```
procedure SET_INTERIOR_STYLE_INDEX
  (STYLE_IND : in STYLE_INDEX);
```

SET INTERIOR COLOUR INDEX

```
procedure SET_INTERIOR_COLOUR_INDEX
  (INTERIOR_COLOUR : in COLOUR_INDEX);
```

SET EDGE FLAG

```
procedure SET_EDGE_FLAG
  (FLAG : in EDGE_FLAG);
```

SET EDGETYPE

```
procedure SET_EDGETYPE
  (TYPE_OF_EDGE : in EDGETYPE);
```

SET EDGEWIDTH SCALE FACTOR

```
procedure SET_EDGEWIDTH_SCALE_FACTOR
  (EDGEWIDTH_SF : in EDGEWIDTH);
```

SET EDGE COLOUR INDEX

```
procedure SET_EDGE_COLOUR_INDEX
  (EDGE_COLOUR : in COLOUR_INDEX);
```

SET PATTERN SIZE

```
procedure SET_PATTERN_SIZE
  (SIZE : in MC.SIZE_2);
```

SET PATTERN REFERENCE POINT AND VECTORS

```
procedure SET_PATTERN_REFERENCE_POINT_AND_VECTORS
  (REFERENCE_POINT     : in MC.POINT_3;
   REFERENCE_VECTORS  : in MC.VECTOR_PAIR_3);
```

SET PATTERN REFERENCE POINT

```
procedure SET_PATTERN_REFERENCE_POINT
  (REFERENCE_POINT : MC.POINT_2);
```

ADD NAMES TO SET

```
procedure ADD_NAMES_TO_SET
  (NAMES_TO_ADD : in NAME_SET);
```

REMOVE NAMES FROM SET

```
procedure REMOVE_NAMES_FROM_SET
  (NAMES_TO_REMOVE : in NAME_SET);
```

SET INDIVIDUAL ASF

```
procedure SET_INDIVIDUAL_ASF
  (ASPECT_ID       : in ASPECT;
   SOURCE_FLAG  : in ASF);
```

SET POLYLINE REPRESENTATION

```
procedure SET_POLYLINE_REPRESENTATION
  (WS                 : in WS_ID;
   POLYLINE_IND  : in POLYLINE_INDEX;
   TYPE_OF_LINE  : in LINETYPE;
   LINEWIDTH_SF  : in LINEWIDTH;
   LINE_COLOUR   : in COLOUR_INDEX);
```

SET POLYMARKER REPRESENTATION

```
procedure SET_POLYMARKER_REPRESENTATION
  (WS                : in WS_ID;
   POLYMARKER_IND    : in POLYMARKER_INDEX;
   TYPE_OF_MARKER    : in MARKER_TYPE;
   SIZE              : in MARKER_SIZE;
   MARKER_COLOUR     : in COLOUR_INDEX);
```

SET TEXT REPRESENTATION

```
procedure SET_TEXT_REPRESENTATION
  (WS                : in WS_ID;
   TEXT_IND          : in TEXT_INDEX;
   FONT              : in TEXT_FONT;
   PRECISION         : in TEXT_PRECISION;
   EXPANSION         : in CHAR_EXPANSION;
   SPACING           : in CHAR_SPACING;
   TEXT_COLOUR       : in COLOUR_INDEX);
```

SET INTERIOR REPRESENTATION

```
procedure SET_INTERIOR_REPRESENTATION
  (WS                : in WS_ID;
   INTERIOR_IND      : in INTERIOR_INDEX;
   STYLE_OF_INTERIOR : in INTERIOR_STYLE;
   STYLE_IND         : in STYLE_INDEX;
   INTERIOR_COLOUR   : in COLOUR_INDEX);
```

SET EDGE REPRESENTATION

```
procedure SET_EDGE_REPRESENTATION
  (WS                : in WS_ID;
   EDGE_IND          : in EDGE_INDEX;
   FLAG              : in EDGE_FLAG;
   TYPE_OF_EDGE      : in EDGETYPE;
   EDGEWIDTH_SF      : in EDGEWIDTH;
   EDGE_COLOUR       : in COLOUR_INDEX);
```

SET PATTERN REPRESENTATION

```
procedure SET_PATTERN_REPRESENTATION
  (WS                : in WS_ID;
   PATTERN_IND       : in PATTERN_INDEX;
   PATTERN           : in COLOUR_MATRIX);
```

SET COLOUR REPRESENTATION

```
procedure SET_COLOUR_REPRESENTATION
  (WS              : in WS_ID;
   COLOUR_IND      : in COLOUR_INDEX;
   COLOUR          : in COLOUR_REPRESENTATION);
```

SET HIGHLIGHTING FILTER

```
procedure SET_HIGHLIGHTING_FILTER
  (WS                : in WS_ID;
   HIGHLIGHTING      : in NAME_SET_FILTER);
```

SET INVISIBILITY FILTER

```
procedure SET_INVISIBILITY_FILTER
  (WS                : in WS_ID;
   INVISIBILITY      : in NAME_SET_FILTER);
```

SET COLOUR MODEL

```
procedure SET_COLOUR_MODEL
  (WS          : in WS_ID;
   MODEL       : in COLOUR_MODEL);
```

SET HLHSR IDENTIFIER

```
procedure SET_HLHSR_IDENTIFIER
  (HLHSR_IDENTIFIER : in HLHSR_ID);
```

SET HLHSR MODE

```
procedure SET_HLHSR_MODE
  (WS          : WS_ID;
   MODE        : in HLHSR_MODE);
```

5.4 Transformation and clipping functions

SET LOCAL TRANSFORMATION 3

```
procedure SET_LOCAL_TRANSFORMATION
  (MATRIX          : in TRANSFORMATION_MATRIX_3;
   HOW_APPLIED     : in COMPOSITION_TYPE);
```

SET LOCAL TRANSFORMATION

```
procedure SET_LOCAL_TRANSFORMATION
  (MATRIX          : in TRANSFORMATION_MATRIX_2;
   HOW_APPLIED     : in COMPOSITION_TYPE);
```

SET GLOBAL TRANSFORMATION 3

```
procedure SET_GLOBAL_TRANSFORMATION
  (MATRIX : in TRANSFORMATION_MATRIX_3);
```

SET GLOBAL TRANFORMATION

```
procedure SET_GLOBAL_TRANSFORMATION
  (MATRIX : in TRANSFORMATION_MATRIX_2);
```

SET MODELLING CLIPPING VOLUME

```
procedure SET_MODELLING_CLIPPING_VOLUME
  (OPERATOR      : in MODELLING_CLIP_OPERATION_TYPE;
   HALF_SPACES   : in MC.HALF_SPACE_LIST_3);
```

SET MODELLING CLIPPING VOLUME

```
procedure SET_MODELLING_CLIPPING_VOLUME
  (OPERATOR      : in MODELLING_CLIP_OPERATION_TYPE;
   HALF_SPACES   : in MC.HALF_SPACE_LIST_2);
```

SET MODELLING CLIPPING INDICATOR

```
procedure SET_MODELLING_CLIPPING_INDICATOR
  (MODELLING_CLIPPING : in CLIPPING_INDICATOR);
```

RESTORE MODELLING CLIPPING VOLUME

```
procedure RESTORE_MODELLING_CLIPPING_VOLUME;
```

SET VIEW INDEX

```
procedure SET_VIEW_INDEX
  (VIEW_IND : in VIEW_INDEX);
```

SET VIEW REPRESENTATION 3

```
procedure SET_VIEW_REPRESENTATION
  (WS                  : in WS_ID;
   VIEW_IND            : in POSITIVE_VIEW_INDEX;
   ORIENTATION_MATRIX  : in TRANSFORMATION_MATRIX_3;
   MAPPING_MATRIX      : in TRANSFORMATION_MATRIX_3;
   CLIPPING_LIMITS     : in NPC.RECTANGULAR_REGION_3;
   XY_CLIPPING         : in CLIPPING_INDICATOR;
   BACK_CLIPPING       : in CLIPPING_INDICATOR;
   FRONT_CLIPPING      : in CLIPPING_INDICATOR);
```

SET VIEW REPRESENTATION

```
procedure SET_VIEW_REPRESENTATION
  (WS                  : in WS_ID;
   VIEW_IND            : in POSITIVE_VIEW_INDEX;
   ORIENTATION_MATRIX  : in TRANSFORMATION_MATRIX_2;
   MAPPING_MATRIX      : in TRANSFORMATION_MATRIX_2;
   CLIPPING_LIMITS     : in NPC.RECTANGULAR_REGION_2;
   XY_CLIPPING         : in CLIPPING_INDICATOR);
```

SET VIEW TRANSFORMATION INPUT PRIORITY

```
procedure SET_VIEW_TRANSFORMATION_INPUT_PRIORITY
  (WS                  : in WS_ID;
   VIEW_IND            : in VIEW_INDEX;
   REFERENCE_IND       : in VIEW_INDEX;
   PRIORITY            : in RELATIVE_PRIORITY);
```

SET WORKSTATION WINDOW 3

```
procedure SET_WS_WINDOW
  (WS                  : in WS_ID;
   WINDOW_LIMITS       : in NPC.RECTANGULAR_REGION_3);
```

SET WORKSTATION WINDOW

```
procedure SET_WS_WINDOW
  (WS                : in WS_ID;
   WINDOW_LIMITS     : in NPC.RECTANGULAR_REGION_2);
```

SET WORKSTATION VIEWPORT 3

```
procedure SET_WS_VIEWPORT
  (WS                : in WS_ID;
   VIEWPORT_LIMITS   : in DC.RECTANGULAR_REGION_3);
```

SET WORKSTATION VIEWPORT

```
procedure SET_WS_VIEWPORT
  (WS                : in WS_ID;
   VIEWPORT_LIMITS   : in DC.RECTANGULAR_REGION_2);
```

TRANSLATE 3

```
procedure TRANSLATE
  (VECTOR            : in  MC.VECTOR_3;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MATRIX            : out TRANSFORMATION_MATRIX_3);
```

TRANSLATE

```
procedure TRANSLATE
  (VECTOR            : in  MC.VECTOR_2;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MATRIX            : out TRANSFORMATION_MATRIX_2);
```

SCALE 3

```
procedure SCALE
  (FACTOR            : in  MC.VECTOR_3;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MATRIX            : out TRANSFORMATION_MATRIX_3);
```

SCALE

```
procedure SCALE
  (FACTOR            : in  MC.VECTOR_2;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MATRIX            : out TRANSFORMATION_MATRIX_2);
```

ROTATE X

```
procedure ROTATE_X
  (ANGLE_X           : in  ANGLE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MATRIX            : out TRANSFORMATION_MATRIX_3);
```

ROTATE Y

```
procedure ROTATE_Y
  (ANGLE_Y           : in ANGLE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MATRIX            : out TRANSFORMATION_MATRIX_3);
```

ROTATE Z

```
procedure ROTATE_Z
  (ANGLE_Z           : in  ANGLE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MATRIX            : out TRANSFORMATION_MATRIX_3);
```

ROTATE

```
procedure ROTATE
  (ANGLE_Z           : in  ANGLE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MATRIX            : out TRANSFORMATION_MATRIX_2);
```

COMPOSE MATRIX 3

```
procedure COMPOSE_MATRIX
  (MATRIX_A          : in  TRANSFORMATION_MATRIX_3;
   MATRIX_B          : in  TRANSFORMATION_MATRIX_3;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   COMPOSED_MATRIX   : out TRANSFORMATION_MATRIX_3);
```

COMPOSE MATRIX

```
procedure COMPOSE_MATRIX
  (MATRIX_A          : in  TRANSFORMATION_MATRIX_2;
   MATRIX_B          : in  TRANSFORMATION_MATRIX_2;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   COMPOSED_MATRIX   : out TRANSFORMATION_MATRIX_2);
```

TRANSFORM POINT 3

```
procedure TRANSFORM_POINT
  (POINT                : in  MC.POINT_3;
   MATRIX               : in  TRANSFORMATION_MATRIX_3;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   TRANSFORMED_POINT    : out MC.POINT_3);
```

TRANSFORM POINT

```
procedure TRANSFORM_POINT
  (POINT                : in  MC.POINT_2;
   MATRIX               : in  TRANSFORMATION_MATRIX_2;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   TRANSFORMED_POINT    : out MC.POINT_2);
```

BUILD TRANSFORMATION MATRIX 3

```
procedure BUILD_TRANSFORMATION_MATRIX
  (FIXED_POINT          : in  MC.POINT_3;
   SHIFT_VECTOR         : in  MC.VECTOR_3;
   ANGLE_X              : in  ANGLE;
   ANGLE_Y              : in  ANGLE;
   ANGLE_Z              : in  ANGLE;
   SCALE_FACTORS        : in  MC.VECTOR_3;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   MATRIX               : out TRANSFORMATION_MATRIX_3);
```

BUILD TRANSFORMATION MATRIX

```
procedure BUILD_TRANSFORMATION_MATRIX
  (FIXED_POINT          : in  MC.POINT_2;
   SHIFT_VECTOR         : in  MC.VECTOR_2;
   ANGLE_Z              : in  ANGLE;
   SCALE_FACTORS        : in  MC.VECTOR_2;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   MATRIX               : out TRANSFORMATION_MATRIX_2);
```

COMPOSE TRANSFORMATION MATRIX 3

```
procedure COMPOSE_TRANSFORMATION_MATRIX
  (MATRIX              : in  TRANSFORMATION_MATRIX_3;
   FIXED_POINT         : in  MC.POINT_3;
   SHIFT_VECTOR        : in  MC.VECTOR_3;
   ANGLE_X             : in  ANGLE;
   ANGLE_Y             : in  ANGLE;
   ANGLE_Z             : in  ANGLE;
   SCALE_FACTORS       : in  MC.VECTOR_3;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   COMPOSED_MATRIX     : out TRANSFORMATION_MATRIX_3);
```

COMPOSE TRANSFORMATION MATRIX

```
procedure COMPOSE_TRANSFORMATION_MATRIX
  (MATRIX              : in  TRANSFORMATION_MATRIX_2;
   FIXED_POINT         : in  MC.POINT_2;
   SHIFT_VECTOR        : in  MC.VECTOR_2;
   ANGLE_Z             : in  ANGLE;
   SCALE_FACTORS       : in  MC.VECTOR_2;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   COMPOSED_MATRIX     : out TRANSFORMATION_MATRIX_2);
```

EVALUATE VIEW ORIENTATION MATRIX 3

```
procedure EVALUATE_VIEW_ORIENTATION_MATRIX
  (REFERENCE_POINT     : in  WC.POINT_3;
   NORMAL_VECTOR       : in  WC.VECTOR_3;
   UP_VECTOR           : in  WC.VECTOR_3;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   ORIENTATION_MATRIX  : out TRANSFORMATION_MATRIX_3);
```

EVALUATE VIEW ORIENTATION MATRIX

```
procedure EVALUATE_VIEW_ORIENTATION_MATRIX
  (REFERENCE_POINT     : in  WC.POINT_2;
   UP_VECTOR           : in  WC.VECTOR_2;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   ORIENTATION_MATRIX  : out TRANSFORMATION_MATRIX_2);
```

EVALUATE VIEW MAPPING MATRIX 3

```
procedure EVALUATE_VIEW_MAPPING_MATRIX
  (WINDOW_LIMITS        : in  VRC.RECTANGULAR_REGION_2;
   VIEWPORT_LIMITS      : in  NPC.RECTANGULAR_REGION_3;
   TYPE_OF_PROJECTION   : in  PROJECTION_TYPE;
   REFERENCE_POINT      : in  VRC.POINT_3;
   VIEW_DISTANCE        : in  VRC_TYPE;
   BACK_DISTANCE        : in  VRC_TYPE;
   FRONT_DISTANCE       : in  VRC_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   MAPPING_MATRIX       : out TRANSFORMATION_MATRIX_3);
```

EVALUATE VIEW MAPPING MATRIX

```
procedure EVALUATE_VIEW_MAPPING_MATRIX
  (WINDOW_LIMITS     : in  VRC.RECTANGULAR_REGION_2;
   VIEWPORT_LIMITS   : in  NPC.RECTANGULAR_REGION_2;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MAPPING_MATRIX    : out TRANSFORMATION_MATRIX_2);
```

5.5 Structure content functions

OPEN STRUCTURE

```
procedure OPEN_STRUCTURE
  (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);
```

CLOSE STRUCTURE

```
procedure CLOSE_STRUCTURE;
```

EXECUTE STRUCTURE

```
procedure EXECUTE_STRUCTURE
  (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);
```

LABEL

```
procedure LABEL
  (LABEL_IDENTIFIER : in LABEL_ID);
```

APPLICATION DATA

```
procedure APPLICATION_DATA
  (DATA : in APPLICATION_DATA_RECORD);
```

GENERALIZED STRUCTURE ELEMENT

The Generalized Structure Element (GSE) is bound in a one-to-many fashion, with a separate procedure implemented for each GSE, each with its own parameter interface. GSE names and parameters are registered in the ISO International Register of Graphical Items which is maintained by the Registration Authority. All supported registered GSE's are in a library package PHIGS_GSE.

The minimal support for GSE's required by this standard is that defined in the following specification for the PHIGS_GSE package. The minimal support provides an interface to registered and unregistered GSE's not supported by the implementation. Unsupported GSE's may be inserted in structures using the procedure GENERALIZED_GSE whose external specification is shown.

```
with PHIGS_TYPES;
use  PHIGS_TYPES;
package PHIGS_GSE is
  type GSE_INTEGER_ARRAY is array (SMALL_NATURAL range <>) of
                                   INTEGER;
  type GSE_FLOAT_ARRAY   is array (SMALL_NATURAL range <>) of
                                   GENERAL_FLOAT_PARAMETER;
  type GSE_STRING_ARRAY  is array (SMALL_NATURAL range <>) of
                                   PHIGS_STRING(1..80);

  type GSE_DATA_RECORD(NUM_OF_INTEGERS : SMALL_NATURAL := 0;
                       NUM_OF_REALS    : SMALL_NATURAL := 0;
                       NUM_OF_STRINGS  : SMALL_NATURAL := 0) is

    record
      INTEGER_ARRAY : GSE_INTEGER_ARRAY (1..NUM_OF_INTEGERS);
      REAL_ARRAY    : GSE_FLOAT_ARRAY   (1..NUM_OF_REALS);
      GSE_STRINGS   : GSE_STRING_ARRAY  (1..NUM_OF_STRINGS);
    end record;

procedure GENERALIZED_GSE
  (GSE        : GSE_ID;
   GSE_DATA   : in GSE_DATA_RECORD);

end PHIGS_GSE;
```

Each registered GSE procedure supported by the implementation will be included in package PHIGS_GSE as a separate procedure using the following naming convention which results in references to the GSE by the expression PHIGS_GSE.<name of the GSE procedure>. Specific names will be assigned when the GSE is registered.

```
procedure <name of the GSE procedure> (<parameters as required>);
```

Each unregistered GSE procedure will be a library package using the following naming convention:

```
package PHIGS_UGSE_<name of GSE procedure> is
  procedure GSE;
    -- Ada code for UGSE procedure

  -- The only procedure name used in the package
  -- will be GSE.

end PHIGS_UGSE_<name of the GSE procedure>;
```

SET EDIT MODE

```
procedure SET_EDIT_MODE
  (MODE : in EDIT_MODE);
```

COPY ALL ELEMENTS FROM STRUCTURE

```
procedure COPY_ALL_ELEMENTS_FROM_STRUCTURE
  (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);
```

SET ELEMENT POINTER

```
procedure SET_ELEMENT_POINTER
  (POSITION : in ELEMENT_POSITION);
```

OFFSET ELEMENT POINTER

```
procedure OFFSET_ELEMENT_POINTER
  (OFFSET : in ELEMENT_POSITION);
```

SET ELEMENT POINTER AT LABEL

```
procedure SET_ELEMENT_POINTER_AT_LABEL
  (LABEL_IDENTIFIER : in LABEL_ID);
```

DELETE ELEMENT

```
procedure DELETE_ELEMENT;
```

DELETE ELEMENT RANGE

```
procedure DELETE_ELEMENT_RANGE
  (POSITION_1  : in ELEMENT_POSITION;
   POSITION_2  : in ELEMENT_POSITION);
```

DELETE ELEMENTS BETWEEN LABELS

```
procedure DELETE_ELEMENTS_BETWEEN_LABELS
  (LABEL_IDENTIFIER_1 : in LABEL_ID;
   LABEL_IDENTIFIER_2 : in LABEL_ID);
```

EMPTY STRUCTURE

```
procedure EMPTY_STRUCTURE
  (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);
```

5.6 Structure manipulation functions

DELETE STRUCTURE

```
procedure DELETE_STRUCTURE
  (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);
```

DELETE STRUCTURE NETWORK

```
procedure DELETE_STRUCTURE_NETWORK
  (STRUCTURE_IDENTIFIER : in STRUCTURE_ID;
   HANDLING_FLAG        : in REFERENCE_HANDLING_FLAG);
```

DELETE ALL STRUCTURES

```
procedure DELETE_ALL_STRUCTURES;
```

CHANGE STRUCTURE IDENTIFIER

```
procedure CHANGE_STRUCTURE_IDENTIFIER
  (ORIGINAL_IDENTIFIER    : in STRUCTURE_ID;
   RESULTING_IDENTIFIER   : in STRUCTURE_ID);
```

CHANGE STRUCTURE REFERENCES

```
procedure CHANGE_STRUCTURE_REFERENCES
  (ORIGINAL_IDENTIFIER    : in STRUCTURE_ID;
   RESULTING_IDENTIFIER   : in STRUCTURE_ID);
```

CHANGE STRUCTURE IDENTIFIER AND REFERENCES

```
procedure CHANGE_STRUCTURE_IDENTIFIER_AND_REFERENCES
  (ORIGINAL_IDENTIFIER    : in STRUCTURE_ID;
   RESULTING_IDENTIFIER   : in STRUCTURE_ID);
```

5.7 Structure display functions

POST STRUCTURE

```
procedure POST_STRUCTURE
  (WS                     : in WS_ID;
   STRUCTURE_IDENTIFIER   : in STRUCTURE_ID;
   PRIORITY               : in DISPLAY_PRIORITY);
```

UNPOST STRUCTURE

```
procedure UNPOST_STRUCTURE
  (WS                     : in WS_ID;
   STRUCTURE_IDENTIFIER   : in STRUCTURE_ID);
```

UNPOST ALL STRUCTURES

```
procedure UNPOST_ALL_STRUCTURES
  (WS : in WS_ID);
```

5.8 Structure archive functions

OPEN ARCHIVE FILE

```
procedure OPEN_ARCHIVE_FILE
  (ARCHIVE_IDENTIFIER : in ARCHIVE_ID;
   ARCHIVE_FILE       : in FILE_ID);
```

CLOSE ARCHIVE FILE

```
procedure CLOSE_ARCHIVE_FILE
  (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);
```

ARCHIVE STRUCTURES

```
procedure ARCHIVE_STRUCTURES
  (ARCHIVE_IDENTIFIER       : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS  : in STRUCTURE_IDS.LIST_OF);
```

ARCHIVE STRUCTURE NETWORKS

```
procedure ARCHIVE_STRUCTURE_NETWORKS
  (ARCHIVE_IDENTIFIER       : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS  : in STRUCTURE_IDS.LIST_OF);
```

ARCHIVE ALL STRUCTURES

```
procedure ARCHIVE_ALL_STRUCTURES
  (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);
```

SET CONFLICT RESOLUTION

```
procedure SET_CONFLICT_RESOLUTION
  (ARCHIVAL_CONFLICT   : in CONFLICT_RESOLUTION;
   RETRIEVAL_CONFLICT  : in CONFLICT_RESOLUTION);
```

RETRIEVE STRUCTURE IDENTIFIERS

```
procedure RETRIEVE_STRUCTURE_IDENTIFIERS
  (ARCHIVE_IDENTIFIER       : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS  : out STRUCTURE_IDS.LIST_OF);
```

RETRIEVE STRUCTURES

```
procedure RETRIEVE_STRUCTURES
  (ARCHIVE_IDENTIFIER       : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS  : in STRUCTURE_IDS.LIST_OF);
```

RETRIEVE STRUCTURE NETWORKS

```
procedure RETRIEVE_STRUCTURE_NETWORKS
  (ARCHIVE_IDENTIFIER      : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS   : in STRUCTURE_IDS.LIST_OF);
```

RETRIEVE ALL STRUCTURES

```
procedure RETRIEVE_ALL_STRUCTURES
  (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);
```

RETRIEVE PATHS TO ANCESTORS

```
procedure RETRIEVE_PATHS_TO_ANCESTORS
  (ARCHIVE_IDENTIFIER      : in  ARCHIVE_ID;
   STRUCTURE_IDENTIFIER    : in  STRUCTURE_ID;
   ORDER                   : in  PATH_ORDER;
   DEPTH                   : in  PATH_DEPTH;
   LIST_OF_PATHS           : out REFERENCE_PATHS.LIST_OF);
```

RETRIEVE PATHS TO DESCENDANTS

```
procedure RETRIEVE_PATHS_TO_DESCENDANTS
  (ARCHIVE_IDENTIFIER      : in  ARCHIVE_ID;
   STRUCTURE_IDENTIFIER    : in  STRUCTURE_ID;
   ORDER                   : in  PATH_ORDER;
   DEPTH                   : in  PATH_DEPTH;
   LIST_OF_PATHS           : out REFERENCE_PATHS.LIST_OF);
```

DELETE STRUCTURES FROM ARCHIVE

```
procedure DELETE_STRUCTURES_FROM_ARCHIVE
  (ARCHIVE_IDENTIFIER      : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS   : in STRUCTURE_IDS.LIST_OF);
```

DELETE STRUCTURE NETWORKS FROM ARCHIVE

```
procedure DELETE_STRUCTURE_NETWORKS_FROM_ARCHIVE
  (ARCHIVE_IDENTIFIER      : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS   : in STRUCTURE_IDS.LIST_OF);
```

DELETE ALL STRUCTURES FROM ARCHIVE

```
procedure DELETE_ALL_STRUCTURES_FROM_ARCHIVE
  (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);
```

5.9 Input functions

SET PICK IDENTIFIER

```
procedure SET_PICK_IDENTIFIER
  (PICK_IDENTIFIER : in PICK_ID);
```

SET PICK FILTER

```
procedure SET_PICK_FILTER
  (WS              : in WS_ID;
   PICK_DEVICE     : in PICK_DEVICE_NUMBER;
   PICKABILITY     : in NAME_SET_FILTER);
```

INITIALIZE LOCATOR 3

```
procedure INITIALIZE_LOCATOR
  (WS                  : in WS_ID;
   DEVICE              : in LOCATOR_DEVICE_NUMBER;
   INITIAL_VIEW_IND    : in VIEW_INDEX;
   INITIAL_POSITION    : in WC.POINT_3;
   ECHO_VOLUME         : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD         : in LOCATOR_DATA_RECORD);
```

INITIALIZE LOCATOR

```
procedure INITIALIZE_LOCATOR
  (WS                  : in WS_ID;
   DEVICE              : in LOCATOR_DEVICE_NUMBER;
   INITIAL_VIEW_IND    : in VIEW_INDEX;
   INITIAL_POSITION    : in WC.POINT_2;
   ECHO_AREA           : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD         : in LOCATOR_DATA_RECORD);
```

INITIALIZE STROKE 3

```
procedure INITIALIZE_STROKE
  (WS                  : in WS_ID;
   DEVICE              : in STROKE_DEVICE_NUMBER;
   INITIAL_VIEW_IND    : in VIEW_INDEX;
   INITIAL_STROKE      : in WC.POINT_LIST_3;
   ECHO_VOLUME         : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD         : in STROKE_DATA_RECORD);
```

INITIALIZE STROKE

```
procedure INITIALIZE_STROKE
  (WS                 : in WS_ID;
   DEVICE             : in STROKE_DEVICE_NUMBER;
   INITIAL_VIEW_IND   : in VIEW_INDEX;
   INITIAL_STROKE     : in WC.POINT_LIST_2;
   ECHO_AREA          : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD        : in STROKE_DATA_RECORD);
```

INITIALIZE VALUATOR 3

```
procedure INITIALIZE_VALUATOR
  (WS                 : in WS_ID;
   DEVICE             : in VALUATOR_DEVICE_NUMBER;
   INITIAL_VALUE      : in VALUATOR_VALUE;
   ECHO_VOLUME        : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD        : in VALUATOR_DATA_RECORD);
```

INITIALIZE VALUATOR

```
procedure INITIALIZE_VALUATOR
  (WS                 : in WS_ID;
   DEVICE             : in VALUATOR_DEVICE_NUMBER;
   INITIAL_VALUE      : in VALUATOR_VALUE;
   ECHO_AREA          : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD        : in VALUATOR_DATA_RECORD);
```

INITIALIZE CHOICE 3

```
procedure INITIALIZE_CHOICE
  (WS                 : in WS_ID;
   DEVICE             : in CHOICE_DEVICE_NUMBER;
   INITIAL_STATUS     : in CHOICE_STATUS;
   INITIAL_CHOICE     : in CHOICE_NUMBER;
   ECHO_VOLUME        : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD        : in CHOICE_DATA_RECORD);
```

INITIALIZE CHOICE

```
procedure INITIALIZE_CHOICE
  (WS                 : in WS_ID;
   DEVICE             : in CHOICE_DEVICE_NUMBER;
   INITIAL_STATUS     : in CHOICE_STATUS;
   INITIAL_CHOICE     : in CHOICE_NUMBER;
   ECHO_AREA          : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD        : in CHOICE_DATA_RECORD);
```

INITIALIZE PICK 3

```
procedure INITIALIZE_PICK
  (WS              : in WS_ID;
   DEVICE          : in PICK_DEVICE_NUMBER;
   INITIAL_STATUS  : in PICK_STATUS;
   INITIAL_PATH    : in PICK_PATH;
   ECHO_VOLUME     : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD     : in PICK_DATA_RECORD;
   ORDER           : in PATH_ORDER);
```

INITIALIZE PICK

```
procedure INITIALIZE_PICK
  (WS              : in WS_ID;
   DEVICE          : in PICK_DEVICE_NUMBER;
   INITIAL_STATUS  : in PICK_STATUS;
   INITIAL_PATH    : in PICK_PATH;
   ECHO_AREA       : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD     : in PICK_DATA_RECORD;
   ORDER           : in PATH_ORDER);
```

INITIALIZE STRING 3

```
procedure INITIALIZE_STRING
  (WS              : in WS_ID;
   DEVICE          : in STRING_DEVICE_NUMBER;
   INITIAL_STRING  : in PHIGS_STRING;
   ECHO_VOLUME     : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD     : in STRING_DATA_RECORD);
```

INITIALIZE STRING

```
procedure INITIALIZE_STRING
  (WS              : in WS_ID;
   DEVICE          : in STRING_DEVICE_NUMBER;
   INITIAL_STRING  : in PHIGS_STRING;
   ECHO_AREA       : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD     : in STRING_DATA_RECORD);
```

SET LOCATOR MODE

```
procedure SET_LOCATOR_MODE
  (WS      : in WS_ID;
   DEVICE  : in LOCATOR_DEVICE_NUMBER;
   MODE    : in OPERATING_MODE;
   SWITCH  : in ECHO_SWITCH);
```

SET STROKE MODE

```
procedure SET_STROKE_MODE
  (WS        : in WS_ID;
   DEVICE    : in STROKE_DEVICE_NUMBER;
   MODE      : in OPERATING_MODE;
   SWITCH    : in ECHO_SWITCH);
```

SET VALUATOR MODE

```
procedure SET_VALUATOR_MODE
  (WS        : in WS_ID;
   DEVICE    : in VALUATOR_DEVICE_NUMBER;
   MODE      : in OPERATING_MODE;
   SWITCH    : in ECHO_SWITCH);
```

SET CHOICE MODE

```
procedure SET_CHOICE_MODE
  (WS        : in WS_ID;
   DEVICE    : in CHOICE_DEVICE_NUMBER;
   MODE      : in OPERATING_MODE;
   SWITCH    : in ECHO_SWITCH);
```

SET PICK MODE

```
procedure SET_PICK_MODE
  (WS        : in WS_ID;
   DEVICE    : in PICK_DEVICE_NUMBER;
   MODE      : in OPERATING_MODE;
   SWITCH    : in ECHO_SWITCH);
```

SET STRING MODE

```
procedure SET_STRING_MODE
  (WS        : in WS_ID;
   DEVICE    : in STRING_DEVICE_NUMBER;
   MODE      : in OPERATING_MODE;
   SWITCH    : in ECHO_SWITCH);
```

REQUEST LOCATOR 3

```
procedure REQUEST_LOCATOR
  (WS          : in  WS_ID;
   DEVICE      : in  LOCATOR_DEVICE_NUMBER;
   STATUS      : out INPUT_STATUS;
   VIEW_IND    : out VIEW_INDEX;
   POSITION    : out WC.POINT_3);
```

REQUEST LOCATOR

```
procedure REQUEST_LOCATOR
  (WS          : in  WS_ID;
   DEVICE      : in  LOCATOR_DEVICE_NUMBER;
   STATUS      : out INPUT_STATUS;
   VIEW_IND    : out VIEW_INDEX;
   POSITION    : out WC.POINT_2);
```

REQUEST STROKE 3

```
procedure REQUEST_STROKE
  (WS                : in  WS_ID;
   DEVICE            : in  STROKE_DEVICE_NUMBER;
   STATUS            : out INPUT_STATUS;
   VIEW_IND          : out VIEW_INDEX;
   STROKE_POINTS     : out WC.ACCESS_POINT_LIST_3);
```

REQUEST STROKE

```
procedure REQUEST_STROKE
  (WS                : in  WS_ID;
   DEVICE            : in  STROKE_DEVICE_NUMBER;
   STATUS            : out INPUT_STATUS;
   VIEW_IND          : out VIEW_INDEX;
   STROKE_POINTS     : out WC.ACCESS_POINT_LIST_2);
```

REQUEST VALUATOR

```
procedure REQUEST_VALUATOR
  (WS          : in  WS_ID;
   DEVICE      : in  VALUATOR_DEVICE_NUMBER;
   STATUS      : out INPUT_STATUS;
   VALUE       : out VALUATOR_VALUE);
```

REQUEST CHOICE

```
procedure REQUEST_CHOICE
  (WS       : in  WS_ID;
   DEVICE   : in  CHOICE_DEVICE_NUMBER;
   STATUS   : out CHOICE_REQUEST_STATUS;
   CHOICE   : out CHOICE_NUMBER);
```

REQUEST PICK

```
procedure REQUEST_PICK
  (WS                   : in  WS_ID;
   DEVICE               : in  PICK_DEVICE_NUMBER;
   DEPTH_TO_RETURN      : in  PATH_DEPTH;
   STATUS               : out PICK_REQUEST_STATUS;
   PATH                 : out PICK_PATH);
```

REQUEST STRING

```
procedure REQUEST_STRING
  (WS                   : in  WS_ID;
   DEVICE               : in  STRING_DEVICE_NUMBER;
   STATUS               : out INPUT_STATUS;
   CHAR_STRING          : out INPUT_STRING);
```

SAMPLE LOCATOR 3

```
procedure SAMPLE_LOCATOR
  (WS           : in  WS_ID;
   DEVICE       : in  LOCATOR_DEVICE_NUMBER;
   VIEW_IND     : out VIEW_INDEX;
   POSITION     : out WC.POINT_3);
```

SAMPLE LOCATOR

```
procedure SAMPLE_LOCATOR
  (WS           : in  WS_ID;
   DEVICE       : in  LOCATOR_DEVICE_NUMBER;
   VIEW_IND     : out VIEW_INDEX;
   POSITION     : out WC.POINT_2);
```

SAMPLE STROKE 3

```
procedure SAMPLE_STROKE
  (WS                : in  WS_ID;
   DEVICE            : in  STROKE_DEVICE_NUMBER;
   VIEW_IND          : out VIEW_INDEX;
   STROKE_POINTS     : out WC.ACCESS_POINT_LIST_3);
```

SAMPLE STROKE

```
procedure SAMPLE_STROKE
  (WS                : in  WS_ID;
   DEVICE            : in  STROKE_DEVICE_NUMBER;
   VIEW_IND          : out VIEW_INDEX;
   STROKE_POINTS     : out WC.ACCESS_POINT_LIST_2);
```

SAMPLE VALUATOR

```
procedure SAMPLE_VALUATOR
  (WS       : in  WS_ID;
   DEVICE   : in  VALUATOR_DEVICE_NUMBER;
   VALUE    : out VALUATOR_VALUE);
```

SAMPLE CHOICE

```
procedure SAMPLE_CHOICE
  (WS       : in  WS_ID;
   DEVICE   : in  CHOICE_DEVICE_NUMBER;
   STATUS   : out CHOICE_STATUS;
   CHOICE   : out CHOICE_NUMBER);
```

SAMPLE PICK

```
procedure SAMPLE_PICK
  (WS                : in  WS_ID;
   DEVICE            : in  PICK_DEVICE_NUMBER;
   DEPTH_TO_RETURN   : in  PATH_DEPTH;
   STATUS            : out PICK_STATUS;
   PATH              : out PICK_PATH);
```

SAMPLE STRING

```
procedure SAMPLE_STRING
  (WS                : in  WS_ID;
   DEVICE            : in  STRING_DEVICE_NUMBER;
   CHAR_STRING       : out INPUT_STRING);
```

AWAIT EVENT

```
procedure AWAIT_EVENT
  (TIMEOUT  : in DURATION;
   WS       : out WS_ID;
   DEVICE   : out EVENT_DEVICE_NUMBER);
```

FLUSH DEVICE EVENTS

```
procedure FLUSH_DEVICE_EVENTS
  (WS       : in WS_ID;
   DEVICE   : in EVENT_QUEUE_DEVICE_NUMBER);
```

GET LOCATOR 3

```
procedure GET_LOCATOR
  (VIEW_IND : out VIEW_INDEX;
   POSITION : out WC.POINT_3);
```

GET LOCATOR

```
procedure GET_LOCATOR
  (VIEW_IND : out VIEW_INDEX;
   POSITION : out WC.POINT_2);
```

GET STROKE 3

```
procedure GET_STROKE
  (VIEW_IND      : out VIEW_INDEX;
   STROKE_POINTS : out WC.ACCESS_POINT_LIST_3);
```

GET STROKE

```
procedure GET_STROKE
  (VIEW_IND      : out VIEW_INDEX;
   STROKE_POINTS : out WC.ACCESS_POINT_LIST_2);
```

GET VALUATOR

```
procedure GET_VALUATOR
  (VALUE : out VALUATOR_VALUE);
```

GET CHOICE

```
procedure GET_CHOICE
  (STATUS  : out CHOICE_STATUS;
   CHOICE  : out CHOICE_NUMBER);
```

GET PICK

```
procedure GET_PICK
  (DEPTH_TO_RETURN : in  PATH_DEPTH;
   STATUS          : out PICK_STATUS;
   PATH            : out PICK_PATH);
```

GET STRING

```
procedure GET_STRING
  (CHAR_STRING : out INPUT_STRING);
```

5.10 Metafile functions

WRITE ITEM TO METAFILE

```
procedure WRITE_ITEM_TO_METAFILE
  (WS    : in WS_ID;
   ITEM  : in METAFILE_DATA_RECORD);
```

GET ITEM TYPE FROM METAFILE

```
procedure GET_ITEM_TYPE_FROM_METAFILE
  (WS              : in  WS_ID;
   TYPE_OF_ITEM    : out METAFILE_ITEM_TYPE;
   LENGTH          : out METAFILE_ITEM_LENGTH);
```

READ ITEM FROM METAFILE

```
procedure READ_ITEM_FROM_METAFILE
  (WS              : in  WS_ID;
   MAX_LENGTH      : in  METAFILE_ITEM_LENGTH;
   ITEM            : out METAFILE_DATA_RECORD);
```

INTERPRET ITEM

```
procedure INTERPRET_ITEM
  (ITEM : in METAFILE_DATA_RECORD);
```

5.11 Inquiry functions

INQUIRE SYSTEM STATE VALUE

```
procedure INQ_SYSTEM_STATE_VALUE
  (STATE_VALUE : out SYSTEM_STATE);
```

INQUIRE WORKSTATION STATE VALUE

```
procedure INQ_WS_STATE_VALUE
  (STATE_VALUE : out WS_STATE);
```

INQUIRE STRUCTURE STATE VALUE

```
procedure INQ_STRUCTURE_STATE_VALUE
  (STATE_VALUE : out STRUCTURE_STATE);
```

INQUIRE ARCHIVE STATE VALUE

```
procedure INQ_ARCHIVE_STATE_VALUE
  (STATE_VALUE : out ARCHIVE_STATE);
```

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES

```
procedure INQ_LIST_OF_AVAILABLE_WS_TYPES
  (ERROR_INDICATOR  : out ERROR_NUMBER;
   LIST_OF_TYPES    : out WS_TYPES.LIST_OF);
```

INQUIRE PHIGS FACILITIES

```
procedure INQ_PHIGS_FACILITIES
  (ERROR_INDICATOR                : out ERROR_NUMBER;
   MAX_SIMUL_OPEN_WS              : out PHIGS_POSITIVE;
   MAX_SIMUL_OPEN_ARCHIVES        : out PHIGS_POSITIVE;
   NUMBER_AVAIL_NAMES             : out PHIGS_POSITIVE;
   AVAIL_CHAR_SETS                : out CHAR_SETS.LIST_OF;
   MAX_ISS_NORMAL_FILTER_LIST     : out PHIGS_POSITIVE;
   MAX_ISS_INVERTED_FILTER_LIST   : out PHIGS_POSITIVE);
```

INQUIRE GENERALIZED STRUCTURE ELEMENT FACILITIES

```
procedure INQ_GSE_FACILITIES
  (ERROR_INDICATOR        : out ERROR_NUMBER;
   LIST_AVAIL_GSES        : out GSE_IDS.LIST_OF;
   LIST_WS_DEPENDENCIES   : out WS_DEPENDENCIES.LIST_OF);
```

INQUIRE MODELLING CLIPPING FACILITIES

```
procedure INQ_MODELLING_CLIPPING_FACILITIES
  (ERROR_INDICATOR          : out ERROR_NUMBER;
   NUMBER_DISTINCT_PLANES   : out PHIGS_POSITIVE;
   LIST_OF_OPERATIONS       : out MODELLING_CLIP_OPERATION_TYPES.LIST_OF);
```

INQUIRE EDIT MODE

```
procedure INQ_EDIT_MODE
  (ERROR_INDICATOR : out ERROR_NUMBER;
   MODE            : out EDIT_MODE);
```

INQUIRE SET OF OPEN WORKSTATIONS

```
procedure INQ_SET_OF_OPEN_WS
  (ERROR_INDICATOR  : out ERROR_NUMBER;
   OPEN_WS          : out WS_IDS.LIST_OF);
```

INQUIRE STRUCTURE IDENTIFIERS

```
procedure INQ_STRUCTURE_IDENTIFIERS
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   LIST_OF_STRUCTURES   : out STRUCTURE_IDS.LIST_OF);
```

INQUIRE ARCHIVE FILES

```
procedure INQ_ARCHIVE_FILES
  (ERROR_INDICATOR           : out ERROR_NUMBER;
   LIST_ARCHIVE_IDENTIFIERS  : out ARCHIVE_IDS.LIST_OF;
   LIST_ARCHIVE_FILES        : out VARIABLE_FILE_IDS.LIST_OF);
```

INQUIRE_CONFLICT_RESOLUTION

```
procedure INQ_CONFLICT_RESOLUTION
  (ERROR_INDICATOR     : out ERROR_NUMBER;
   ARCHIVAL_CONFLICT   : out CONFLICT_RESOLUTION;
   RETRIEVAL_CONFLICT  : out CONFLICT_RESOLUTION);
```

INQUIRE ALL CONFLICTING STRUCTURES

```
procedure INQ_ALL_CONFLICTING_STRUCTURES
  (ARCHIVE_IDENTIFIER  : in  ARCHIVE_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   LIST_OF_STRUCTURES  : out STRUCTURE_IDS.LIST_OF);
```

INQUIRE CONFLICTING STRUCTURES IN NETWORK

```
procedure INQ_CONFLICTING_STRUCTURES_IN_NETWORK
  (ARCHIVE_IDENTIFIER      : in  ARCHIVE_ID;
   STRUCTURE_IDENTIFIER    : in  STRUCTURE_ID;
   SOURCE                  : in  STRUCTURE_NETWORK_SOURCE;
   ERROR_INDICATOR         : out ERROR_NUMBER;
   LIST_OF_STRUCTURES      : out STRUCTURE_IDS.LIST_OF);
```

INQUIRE MORE SIMULTANEOUS EVENTS

```
procedure INQ_MORE_SIMULTANEOUS_EVENTS
  (ERROR_INDICATOR  : out ERROR_NUMBER;
   EVENTS           : out MORE_EVENTS);
```

INQUIRE WORKSTATION CONNECTION AND TYPE

```
procedure INQ_WS_CONNECTION_AND_TYPE
  (WS                : in  WS_ID;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   CONNECTION        : out VARIABLE_CONNECTION_ID;
   TYPE_OF_WS        : out WS_TYPE);
```

INQUIRE LIST OF VIEW INDICES

```
procedure INQ_LIST_OF_VIEW_INDICES
  (WS                    : in  WS_ID;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   DEFINED_VIEW_LIST     : out VIEW_INDICES.LIST_OF);
```

INQUIRE VIEW REPRESENTATION

```
procedure INQ_VIEW_REPRESENTATION
  (WS                                 : in  WS_ID;
   VIEW_IND                           : in  VIEW_INDEX;
   ERROR_INDICATOR                    : out ERROR_NUMBER;
   UPDATE                             : out UPDATE_STATE;
   REQUESTED_ORIENTATION_MATRIX       : out TRANSFORMATION_MATRIX_3;
   CURRENT_ORIENTATION_MATRIX         : out TRANSFORMATION_MATRIX_3;
   REQUESTED_MAPPING_MATRIX           : out TRANSFORMATION_MATRIX_3;
   CURRENT_MAPPING_MATRIX             : out TRANSFORMATION_MATRIX_3;
   REQUESTED_CLIPPING_LIMITS          : out NPC.RECTANGULAR_REGION_3;
   CURRENT_CLIPPING_LIMITS            : out NPC.RECTANGULAR_REGION_3;
   REQUESTED_XY_CLIPPING              : out CLIPPING_INDICATOR;
   CURRENT_XY_CLIPPING                : out CLIPPING_INDICATOR;
   REQUESTED_BACK_CLIPPING            : out CLIPPING_INDICATOR;
   CURRENT_BACK_CLIPPING              : out CLIPPING_INDICATOR;
   REQUESTED_FRONT_CLIPPING           : out CLIPPING_INDICATOR;
   CURRENT_FRONT_CLIPPING             : out CLIPPING_INDICATOR);
```

INQUIRE HLHSR MODE

```
procedure INQ_HLHSR_MODE
  (WS                    : in  WS_ID;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   UPDATE                : out UPDATE_STATE;
   CURRENT_MODE          : out HLHSR_MODE;
   REQUESTED_MODE        : out HLHSR_MODE);
```

INQUIRE POSTED STRUCTURES

```
procedure INQ_POSTED_STRUCTURES
  (WS                    : in  WS_ID;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   LIST_OF_STRUCTURES    : out POSTED_STRUCTURES.LIST_OF);
```

INQUIRE DISPLAY UPDATE STATE

```
procedure INQ_DISPLAY_UPDATE_STATE
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   DEFERRAL            : out DEFERRAL_MODE;
   MODIFICATION        : out MODIFICATION_MODE;
   EMPTINESS           : out DISPLAY_SURFACE_EMPTY;
   VISUAL_STATE        : out VISUAL_REPRESENTATION_STATE);
```

INQUIRE LIST OF POLYLINE INDICES

```
procedure INQ_LIST_OF_POLYLINE_INDICES
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   INDICES             : out POLYLINE_INDICES.LIST_OF);
```

INQUIRE POLYLINE REPRESENTATION

```
procedure INQ_POLYLINE_REPRESENTATION
  (WS                  : in  WS_ID;
   POLYLINE_IND        : in  POLYLINE_INDEX;
   RETURNED_VALUES     : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   TYPE_OF_LINE        : out LINETYPE;
   LINEWIDTH_SF        : out LINEWIDTH;
   LINE_COLOUR         : out COLOUR_INDEX);
```

INQUIRE LIST OF POLYMARKER INDICES

```
procedure INQ_LIST_OF_POLYMARKER_INDICES
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   INDICES             : out POLYMARKER_INDICES.LIST_OF);
```

INQUIRE POLYMARKER REPRESENTATION

```
procedure INQ_POLYMARKER_REPRESENTATION
  (WS                  : in  WS_ID;
   POLYMARKER_IND      : in  POLYMARKER_INDEX;
   RETURNED_VALUES     : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   TYPE_OF_MARKER      : out MARKER_TYPE;
   SIZE                : out MARKER_SIZE;
   MARKER_COLOUR       : out COLOUR_INDEX);
```

INQUIRE LIST OF TEXT INDICES

```
procedure INQ_LIST_OF_TEXT_INDICES
  (WS                : in  WS_ID;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   INDICES           : out TEXT_INDICES.LIST_OF);
```

INQUIRE TEXT REPRESENTATION

```
procedure INQ_TEXT_REPRESENTATION
  (WS                : in  WS_ID;
   TEXT_IND          : in  TEXT_INDEX;
   RETURNED_VALUES   : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   FONT              : out TEXT_FONT;
   PRECISION         : out TEXT_PRECISION;
   EXPANSION         : out CHAR_EXPANSION;
   SPACING           : out CHAR_SPACING;
   TEXT_COLOUR       : out COLOUR_INDEX);
```

INQUIRE LIST OF INTERIOR INDICES

```
procedure INQ_LIST_OF_INTERIOR_INDICES
  (WS                : in  WS_ID;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   INDICES           : out INTERIOR_INDICES.LIST_OF);
```

INQUIRE INTERIOR REPRESENTATION

```
procedure INQ_INTERIOR_REPRESENTATION
  (WS                : in  WS_ID;
   INTERIOR_IND      : in  INTERIOR_INDEX;
   RETURNED_VALUES   : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   STYLE_OF_INTERIOR : out INTERIOR_STYLE;
   STYLE_IND         : out STYLE_INDEX;
   INTERIOR_COLOUR   : out COLOUR_INDEX);
```

INQUIRE LIST OF EDGE INDICES

```
procedure INQ_LIST_OF_EDGE_INDICES
  (WS                : in  WS_ID;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   INDICES           : out EDGE_INDICES.LIST_OF);
```

INQUIRE EDGE REPRESENTATION

```
procedure INQ_EDGE_REPRESENTATION
  (WS                  : in  WS_ID;
   EDGE_IND            : in  EDGE_INDEX;
   RETURNED_VALUES     : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   FLAG                : out EDGE_FLAG;
   TYPE_OF_EDGE        : out EDGETYPE;
   EDGEWIDTH_SF        : out EDGEWIDTH;
   EDGE_COLOUR         : out COLOUR_INDEX);
```

INQUIRE LIST OF PATTERN INDICES

```
procedure INQ_LIST_OF_PATTERN_INDICES
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   INDICES             : out PATTERN_INDICES.LIST_OF);
```

INQUIRE PATTERN REPRESENTATION

```
procedure INQ_PATTERN_REPRESENTATION
  (WS                  : in  WS_ID;
   PATTERN_IND         : in  PATTERN_INDEX;
   RETURNED_VALUES     : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   PATTERN             : out ACCESS_COLOUR_MATRIX);
```

INQUIRE COLOUR MODEL

```
procedure INQ_COLOUR_MODEL
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MODEL               : out COLOUR_MODEL);
```

INQUIRE LIST OF COLOUR INDICES

```
procedure INQ_LIST_OF_COLOUR_INDICES
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   INDICES             : out COLOUR_INDICES.LIST_OF);
```

INQUIRE COLOUR REPRESENTATION

```
procedure INQ_COLOUR_REPRESENTATION
  (WS                  : in  WS_ID;
   COLOUR_IND          : in  COLOUR_INDEX;
   RETURNED_VALUES     : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   COLOUR              : out COLOUR_REPRESENTATION);
```

INQUIRE HIGHLIGHTING FILTER

```
procedure INQ_HIGHLIGHTING_FILTER
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   HIGHLIGHTING        : out NAME_SET_FILTER);
```

INQUIRE INVISIBILITIY FILTER

```
procedure INQ_INVISIBILITY_FILTER
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   INVISIBILITY        : out NAME_SET_FILTER);
```

INQUIRE WORKSTATION TRANSFORMATION 3

```
procedure INQ_WS_TRANSFORMATION
  (WS                          : in  WS_ID;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   UPDATE                      : out UPDATE_STATE;
   REQUESTED_WINDOW_LIMITS     : out NPC.RECTANGULAR_REGION_3;
   CURRENT_WINDOW_LIMITS       : out NPC.RECTANGULAR_REGION_3;
   REQUESTED_VIEWPORT_LIMITS   : out DC.RECTANGULAR_REGION_3;
   CURRENT_VIEWPORT_LIMITS     : out DC.RECTANGULAR_REGION_3);
```

INQUIRE WORKSTATION TRANSFORMATION

```
procedure INQ_WS_TRANSFORMATION
  (WS                          : in  WS_ID;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   UPDATE                      : out UPDATE_STATE;
   REQUESTED_WINDOW_LIMITS     : out NPC.RECTANGULAR_REGION_2;
   CURRENT_WINDOW_LIMITS       : out NPC.RECTANGULAR_REGION_2;
   REQUESTED_VIEWPORT_LIMITS   : out DC.RECTANGULAR_REGION_2;
   CURRENT_VIEWPORT_LIMITS     : out DC.RECTANGULAR_REGION_2);
```

INQUIRE LOCATOR DEVICE STATE 3

```
procedure INQ_LOCATOR_DEVICE_STATE
  (WS                  : in  WS_ID;
   DEVICE              : in  LOCATOR_DEVICE_NUMBER;
   RETURNED_VALUES     : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MODE                : out OPERATING_MODE;
   SWITCH              : out ECHO_SWITCH;
   INITIAL_VIEW_IND    : out VIEW_INDEX;
   INITIAL_POSITION    : out WC.POINT_3;
   ECHO_VOLUME         : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD         : out LOCATOR_DATA_RECORD);
```

INQUIRE LOCATOR DEVICE STATE

```
procedure INQ_LOCATOR_DEVICE_STATE
  (WS                  : in  WS_ID;
   DEVICE              : in  LOCATOR_DEVICE_NUMBER;
   RETURNED_VALUES     : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MODE                : out OPERATING_MODE;
   SWITCH              : out ECHO_SWITCH;
   INITIAL_VIEW_IND    : out VIEW_INDEX;
   INITIAL_POSITION    : out WC.POINT_2;
   ECHO_AREA           : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD         : out LOCATOR_DATA_RECORD);
```

INQUIRE STROKE DEVICE STATE 3

```
procedure INQ_STROKE_DEVICE_STATE
  (WS                  : in  WS_ID;
   DEVICE              : in  STROKE_DEVICE_NUMBER;
   RETURNED_VALUES     : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MODE                : out OPERATING_MODE;
   SWITCH              : out ECHO_SWITCH;
   INITIAL_VIEW_IND    : out VIEW_INDEX;
   INITIAL_STROKE_POINTS : out WC.ACCESS_POINT_LIST_3;
   ECHO_VOLUME         : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD         : out STROKE_DATA_RECORD);
```

INQUIRE STROKE DEVICE STATE

```
procedure INQ_STROKE_DEVICE_STATE
  (WS                    : in  WS_ID;
   DEVICE                : in  STROKE_DEVICE_NUMBER;
   RETURNED_VALUES       : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   MODE                  : out OPERATING_MODE;
   SWITCH                : out ECHO_SWITCH;
   INITIAL_VIEW_IND      : out VIEW_INDEX;
   INITIAL_STROKE_POINTS : out WC.ACCESS_POINT_LIST_2;
   ECHO_AREA             : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD           : out STROKE_DATA_RECORD);
```

INQUIRE VALUATOR DEVICE STATE 3

```
procedure INQ_VALUATOR_DEVICE_STATE
  (WS                : in  WS_ID;
   DEVICE            : in  VALUATOR_DEVICE_NUMBER;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MODE              : out OPERATING_MODE;
   SWITCH            : out ECHO_SWITCH;
   INITIAL_VALUE     : out VALUATOR_VALUE;
   ECHO_VOLUME       : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD       : out VALUATOR_DATA_RECORD);
```

INQUIRE VALUATOR DEVICE STATE

```
procedure INQ_VALUATOR_DEVICE_STATE
  (WS                : in  WS_ID;
   DEVICE            : in  VALUATOR_DEVICE_NUMBER;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MODE              : out OPERATING_MODE;
   SWITCH            : out ECHO_SWITCH;
   INITIAL_VALUE     : out VALUATOR_VALUE;
   ECHO_AREA         : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD       : out VALUATOR_DATA_RECORD);
```

INQUIRE CHOICE DEVICE STATE 3

```
procedure INQ_CHOICE_DEVICE_STATE
  (WS                : in  WS_ID;
   DEVICE            : in  CHOICE_DEVICE_NUMBER;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   MODE              : out OPERATING_MODE;
   SWITCH            : out ECHO_SWITCH;
   INITIAL_STATUS    : out CHOICE_STATUS;
   INITIAL_CHOICE    : out CHOICE_NUMBER;
   ECHO_VOLUME       : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD       : out CHOICE_DATA_RECORD);
```

INQUIRE CHOICE DEVICE STATE

```
procedure INQ_CHOICE_DEVICE_STATE
  (WS                 : in  WS_ID;
   DEVICE             : in  CHOICE_DEVICE_NUMBER;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   MODE               : out OPERATING_MODE;
   SWITCH             : out ECHO_SWITCH;
   INITIAL_STATUS     : out CHOICE_STATUS;
   INITIAL_CHOICE     : out CHOICE_NUMBER;
   ECHO_AREA          : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD        : out CHOICE_DATA_RECORD);
```

INQUIRE PICK DEVICE STATE 3

```
procedure INQ_PICK_DEVICE_STATE
  (WS                 : in  WS_ID;
   DEVICE             : in  PICK_DEVICE_NUMBER;
   RETURNED_VALUES    : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   MODE               : out OPERATING_MODE;
   SWITCH             : out ECHO_SWITCH;
   PICKABILITY        : out NAME_SET_FILTER;
   INITIAL_STATUS     : out PICK_STATUS;
   INITIAL_PATH       : out PICK_PATH;
   ECHO_VOLUME        : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD        : out PICK_DATA_RECORD;
   ORDER              : out PATH_ORDER);
```

INQUIRE PICK DEVICE STATE

```
procedure INQ_PICK_DEVICE_STATE
  (WS                 : in  WS_ID;
   DEVICE             : in  PICK_DEVICE_NUMBER;
   RETURNED_VALUES    : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   MODE               : out OPERATING_MODE;
   SWITCH             : out ECHO_SWITCH;
   PICKABILITY        : out NAME_SET_FILTER;
   INITIAL_STATUS     : out PICK_STATUS;
   INITIAL_PATH       : out PICK_PATH;
   ECHO_AREA          : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD        : out PICK_DATA_RECORD;
   ORDER              : out PATH_ORDER);
```

INQUIRE STRING DEVICE STATE 3

```
procedure INQ_STRING_DEVICE_STATE
  (WS                    : in  WS_ID;
   DEVICE                : in  STRING_DEVICE_NUMBER;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   MODE                  : out OPERATING_MODE;
   SWITCH                : out ECHO_SWITCH;
   INITIAL_CHAR_STRING   : out INPUT_STRING;
   ECHO_VOLUME           : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD           : out STRING_DATA_RECORD);
```

INQUIRE STRING DEVICE STATE

```
procedure INQ_STRING_DEVICE_STATE
  (WS                    : in  WS_ID;
   DEVICE                : in  STRING_DEVICE_NUMBER;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   MODE                  : out OPERATING_MODE;
   SWITCH                : out ECHO_SWITCH;
   INITIAL_CHAR_STRING   : out INPUT_STRING;
   ECHO_AREA             : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD           : out STRING_DATA_RECORD);
```

INQUIRE WORKSTATION CATEGORY

```
procedure INQ_WS_CATEGORY
  (TYPE_OF_WS        : in  WS_TYPE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   CATEGORY          : out WS_CATEGORY);
```

INQUIRE DISPLAY SPACE SIZE 3

```
procedure INQ_DISPLAY_SPACE_SIZE
  (TYPE_OF_WS              : in  WS_TYPE;
   ERROR_INDICATOR         : out ERROR_NUMBER;
   UNITS                   : out DC_UNITS;
   MAX_DC_UNIT_SIZE        : out DC.SIZE_3;
   MAX_RASTER_UNIT_SIZE    : out RASTER_UNIT_SIZE_3);
```

INQUIRE DISPLAY SPACE SIZE

```
procedure INQ_DISPLAY_SPACE_SIZE
  (TYPE_OF_WS              : in  WS_TYPE;
   ERROR_INDICATOR         : out ERROR_NUMBER;
   UNITS                   : out DC_UNITS;
   MAX_DC_UNIT_SIZE        : out DC.SIZE_2;
   MAX_RASTER_UNIT_SIZE    : out RASTER_UNIT_SIZE_2);
```

INQUIRE HLHSR FACILITIES

```
procedure INQ_HLHSR_FACILITIES
  (TYPE_OF_WS            : in  WS_TYPE;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   LIST_OF_IDENTIFIERS   : out HLHSR_IDS.LIST_OF;
   LIST_OF_MODES         : out HLHSR_MODES.LIST_OF);
```

INQUIRE VIEW FACILITIES

```
procedure INQ_VIEW_FACILITIES
  (TYPE_OF_WS            : in  WS_TYPE;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   NUMBER_OF_INDICES     : out PHIGS_POSITIVE);
```

INQUIRE PREDEFINED VIEW REPRESENTATION

```
procedure INQ_PREDEFINED_VIEW_REPRESENTATION
  (TYPE_OF_WS            : in  WS_TYPE;
   VIEW_IND              : in  VIEW_INDEX;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   ORIENTATION_MATRIX    : out TRANSFORMATION_MATRIX_3;
   MAPPING_MATRIX        : out TRANSFORMATION_MATRIX_3;
   CLIPPING_LIMITS       : out NPC.RECTANGULAR_REGION_3;
   XY_CLIPPING           : out CLIPPING_INDICATOR;
   BACK_CLIPPING         : out CLIPPING_INDICATOR;
   FRONT_CLIPPING        : out CLIPPING_INDICATOR);
```

INQUIRE WORKSTATION CLASSIFICATION

```
procedure INQ_WS_CLASSIFICATION
  (TYPE_OF_WS            : in  WS_TYPE;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   CLASS                 : out DISPLAY_CLASS);
```

INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES

```
procedure INQ_DYNAMICS_OF_WS_ATTRIBUTES
  (TYPE_OF_WS                    : in  WS_TYPE;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   POLYLINE_REPRESENTATION       : out DYNAMIC_MODIFICATION;
   POLYMARKER_REPRESENTATION     : out DYNAMIC_MODIFICATION;
   TEXT_REPRESENTATION           : out DYNAMIC_MODIFICATION;
   INTERIOR_REPRESENTATION       : out DYNAMIC_MODIFICATION;
   EDGE_REPRESENTATION           : out DYNAMIC_MODIFICATION;
   PATTERN_REPRESENTATION        : out DYNAMIC_MODIFICATION;
   COLOUR_REPRESENTATION         : out DYNAMIC_MODIFICATION;
   VIEW_REPRESENTATION           : out DYNAMIC_MODIFICATION;
   WS_TRANSFORMATION             : out DYNAMIC_MODIFICATION;
   HIGHLIGHTING_FILTER           : out DYNAMIC_MODIFICATION;
   INVISIBILITY_FILTER           : out DYNAMIC_MODIFICATION;
   HLHSR_MODE_CHANGE             : out DYNAMIC_MODIFICATION);
```

INQUIRE DEFAULT DISPLAY UPDATE STATE

```
procedure INQ_DEFAULT_DISPLAY_UPDATE_STATE
  (TYPE_OF_WS          : in  WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   DEFERRAL            : out DEFERRAL_MODE;
   MODIFICATION        : out MODIFICATION_MODE);
```

INQUIRE POLYLINE FACILITIES

```
procedure INQ_POLYLINE_FACILITIES
  (TYPE_OF_WS          : in  WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   NUMBER_OF_TYPES     : out PHIGS_INTEGER;
   LIST_OF_TYPES       : out LINETYPES.LIST_OF;
   NUMBER_OF_WIDTHS    : out PHIGS_NATURAL;
   NOMINAL_WIDTH       : out DC.MAGNITUDE;
   RANGE_OF_WIDTHS     : out DC.RANGE_OF_MAGNITUDES;
   NUMBER_OF_INDICES   : out PHIGS_NATURAL);
```

INQUIRE PREDEFINED POLYLINE REPRESENTATION

```
procedure INQ_PREDEFINED_POLYLINE_REPRESENTATION
  (TYPE_OF_WS          : in  WS_TYPE;
   POLYLINE_IND        : in  POLYLINE_INDEX;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   TYPE_OF_LINE        : out LINETYPE;
   LINEWIDTH_SF        : out LINEWIDTH;
   LINE_COLOUR         : out COLOUR_INDEX);
```

INQUIRE POLYMARKER FACILITIES

```
procedure INQ_POLYMARKER_FACILITIES
  (TYPE_OF_WS            : in WS_TYPE;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   NUMBER_OF_TYPES       : out PHIGS_INTEGER;
   LIST_OF_TYPES         : out MARKER_TYPES.LIST_OF;
   NUMBER_OF_SIZES       : out PHIGS_NATURAL;
   NOMINAL_SIZE          : out DC.MAGNITUDE;
   RANGE_OF_SIZES        : out DC.RANGE_OF_MAGNITUDES;
   NUMBER_OF_INDICES     : out PHIGS_NATURAL);
```

INQUIRE PREDEFINED POLYMARKER REPRESENTATION

```
procedure INQ_PREDEFINED_POLYMARKER_REPRESENTATION
  (TYPE_OF_WS            : in WS_TYPE;
   POLYMARKER_IND        : in POLYMARKER_INDEX;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   TYPE_OF_MARKER        : out MARKER_TYPE;
   SIZE                  : out MARKER_SIZE;
   MARKER_COLOUR         : out COLOUR_INDEX);
```

INQUIRE TEXT FACILITIES

```
procedure INQ_TEXT_FACILITIES
  (TYPE_OF_WS                    : in WS_TYPE;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   LIST_OF_FONT_PRECISION_PAIRS  : out TEXT_FONT_PRECISIONS.LIST_OF;
   NUMBER_OF_HEIGHTS             : out PHIGS_NATURAL;
   RANGE_OF_HEIGHTS              : out DC.RANGE_OF_MAGNITUDES;
   NUMBER_OF_EXPANSIONS          : out PHIGS_NATURAL;
   RANGE_OF_EXPANSIONS           : out DC.RANGE_OF_MAGNITUDES;
   NUMBER_OF_INDICES             : out PHIGS_NATURAL);
```

INQUIRE PREDEFINED TEXT REPRESENTATION

```
procedure INQ_PREDEFINED_TEXT_REPRESENTATION
  (TYPE_OF_WS            : in WS_TYPE;
   TEXT_IND              : in TEXT_INDEX;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   FONT                  : out TEXT_FONT;
   PRECISION             : out TEXT_PRECISION;
   EXPANSION             : out CHAR_EXPANSION;
   SPACING               : out CHAR_SPACING;
   TEXT_COLOUR           : out COLOUR_INDEX);
```

INQUIRE ANNOTATION FACILITIES

```
procedure INQ_ANNOTATION_FACILITIES
  (TYPE_OF_WS                  : in WS_TYPE;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   LIST_OF_ANNOTATION_STYLES   : out ANNOTATION_STYLES.LIST_OF;
   NUMBER_OF_CHAR_HEIGHTS      : out PHIGS_NATURAL;
   RANGE_OF_CHAR_HEIGHTS       : out DC.RANGE_OF_MAGNITUDES);
```

INQUIRE TEXT EXTENT

```
procedure INQ_TEXT_EXTENT
  (TYPE_OF_WS                  : in  WS_TYPE;
   FONT                        : in  TEXT_FONT;
   EXPANSION                   : in  CHAR_EXPANSION;
   SPACING                     : in  CHAR_SPACING;
   HEIGHT                      : in  MC.MAGNITUDE;
   PATH                        : in  TEXT_PATH;
   ALIGNMENT                   : in  TEXT_ALIGNMENT;
   CHAR_STRING                 : in  PHIGS_STRING;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   EXTENT_RECTANGLE            : out MC.RECTANGULAR_REGION_2;
   CONCATENATION_OFFSET        : out MC.VECTOR_2);

-- This procedure using MC coordinates is for use with text primitives.
```

INQUIRE TEXT EXTENT

```
procedure INQ_TEXT_EXTENT
  (TYPE_OF_WS                  : in  WS_TYPE;
   FONT                        : in  TEXT_FONT;
   EXPANSION                   : in  CHAR_EXPANSION;
   SPACING                     : in  CHAR_SPACING;
   HEIGHT                      : in  NPC.MAGNITUDE;
   PATH                        : in  TEXT_PATH;
   ALIGNMENT                   : in  TEXT_ALIGNMENT;
   CHAR_STRING                 : in  PHIGS_STRING;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   EXTENT_RECTANGLE            : out NPC.RECTANGULAR_REGION_2;
   CONCATENATION_OFFSET        : out NPC.VECTOR_2);

-- This procedure using NPC coordinates is for use with annotation text
-- primitives.
```

INQUIRE INTERIOR FACILITIES

```
procedure INQ_INTERIOR_FACILITIES
  (TYPE_OF_WS                  : in  WS_TYPE;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   LIST_OF_INTERIOR_STYLES     : out INTERIOR_STYLES.LIST_OF;
   NUMBER_OF_HATCH_STYLES      : out PHIGS_NATURAL;
   LIST_OF_HATCH_STYLES        : out HATCH_STYLES.LIST_OF;
   NUMBER_OF_INDICES           : out PHIGS_NATURAL);
```

INQUIRE PREDEFINED INTERIOR REPRESENTATION

```
procedure INQ_PREDEFINED_INTERIOR_REPRESENTATION
  (TYPE_OF_WS          : in  WS_TYPE;
   INTERIOR_IND        : in  INTERIOR_INDEX;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   STYLE_OF_INTERIOR   : out INTERIOR_STYLE;
   STYLE_IND           : out STYLE_INDEX;
   INTERIOR_COLOUR     : out COLOUR_INDEX);
```

INQUIRE EDGE FACILITIES

```
procedure INQ_EDGE_FACILITIES
  (TYPE_OF_WS          : in  WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   NUMBER_OF_TYPES     : out PHIGS_INTEGER;
   LIST_OF_TYPES       : out EDGETYPES.LIST_OF;
   NUMBER_OF_WIDTHS    : out PHIGS_NATURAL;
   NOMINAL_WIDTH       : out DC.MAGNITUDE;
   RANGE_OF_WIDTHS     : out DC.RANGE_OF_MAGNITUDES;
   NUMBER_OF_INDICES   : out PHIGS_NATURAL);
```

INQUIRE PREDEFINED EDGE REPRESENTATION

```
procedure INQ_PREDEFINED_EDGE_REPRESENTATION
  (TYPE_OF_WS          : in  WS_TYPE;
   EDGE_IND            : in  EDGE_INDEX;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   FLAG                : out EDGE_FLAG;
   TYPE_OF_EDGE        : out EDGETYPE;
   EDGEWIDTH_SF        : out EDGEWIDTH;
   EDGE_COLOUR         : out COLOUR_INDEX);
```

INQUIRE PATTERN FACILITIES

```
procedure INQ_PATTERN_FACILITIES
  (TYPE_OF_WS          : in  WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   NUMBER_OF_INDICES   : out PHIGS_NATURAL);
```

INQUIRE PREDEFINED PATTERN REPRESENTATION

```
procedure INQ_PREDEFINED_PATTERN_REPRESENTATION
  (TYPE_OF_WS          : in  WS_TYPE;
   PATTERN_IND         : in  PATTERN_INDEX;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   PATTERN             : out ACCESS_COLOUR_MATRIX);
```

INQUIRE COLOUR MODEL FACILITIES

```
procedure INQ_COLOUR_MODEL_FACILITIES
  (TYPE_OF_WS          : in  WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   LIST_OF_MODELS      : out COLOUR_MODELS.LIST_OF;
   DEFAULT_MODEL       : out COLOUR_MODEL);
```

INQUIRE COLOUR FACILITIES

```
procedure INQ_COLOUR_FACILITIES
  (TYPE_OF_WS                : in  WS_TYPE;
   ERROR_INDICATOR           : out ERROR_NUMBER;
   NUMBER_OF_COLOURS         : out PHIGS_NATURAL;
   AVAILABILITY_OF_COLOUR    : out COLOUR_AVAILABLE;
   NUMBER_OF_INDICES         : out PHIGS_NATURAL;
   PRIMARY_COLOURS           : out CHROMATICITY_COEFFICIENT_SET);
```

INQUIRE PREDEFINED COLOUR REPRESENTATION

```
procedure INQ_PREDEFINED_COLOUR_REPRESENTATION
  (TYPE_OF_WS          : in  WS_TYPE;
   INDEX               : in  COLOUR_INDEX;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   COLOUR              : out COLOUR_REPRESENTATION);
```

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3

```
procedure INQ_LIST_OF_AVAILABLE_GDP
  (TYPE_OF_WS          : in  WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   LIST_OF_GDP_3       : out GDP_3_IDS.LIST_OF);
```

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

```
procedure INQ_LIST_OF_AVAILABLE_GDP
  (TYPE_OF_WS          : in  WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   LIST_OF_GDP         : out GDP_IDS.LIST_OF);
```

INQUIRE GENERALIZED DRAWING PRIMITIVE 3

```
procedure INQ_GDP
  (TYPE_OF_WS                    : in  WS_TYPE;
   GDP                           : in  GDP_3_ID;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   LIST_OF_ATTRIBUTE_SETS_USED   : out ATTRIBUTES_USED.LIST_OF);
```

INQUIRE GENERALIZED DRAWING PRIMITIVE

```
procedure INQ_GDP
  (TYPE_OF_WS                    : in  WS_TYPE;
   GDP                           : in  GDP_ID;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   LIST_OF_ATTRIBUTE_SETS_USED   : out ATTRIBUTES_USED.LIST_OF);
```

INQUIRE LIST OF AVAILABLE GENERALIZED STRUCTURE ELEMENTS

```
procedure INQ_LIST_OF_AVAILABLE_GSE
  (TYPE_OF_WS          : in  WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   LIST_OF_GSE         : out GSE_IDS.LIST_OF);
```

INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED

```
procedure INQ_NUMBER_OF_DISPLAY_PRIORITIES_SUPPORTED
  (TYPE_OF_WS                : in  WS_TYPE;
   ERROR_INDICATOR           : out ERROR_NUMBER;
   NUMBER_OF_PRIORITIES      : out PHIGS_NATURAL);
```

INQUIRE WORKSTATION STATE TABLE LENGTHS

```
procedure INQ_WS_STATE_TABLE_LENGTHS
  (TYPE_OF_WS                    : in  WS_TYPE;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   MAX_POLYLINE_ENTRIES          : out PHIGS_NATURAL;
   MAX_POLYMARKER_ENTRIES        : out PHIGS_NATURAL;
   MAX_TEXT_ENTRIES              : out PHIGS_NATURAL;
   MAX_INTERIOR_ENTRIES          : out PHIGS_NATURAL;
   MAX_EDGE_ENTRIES              : out PHIGS_NATURAL;
   MAX_PATTERN_INDICES           : out PHIGS_NATURAL;
   MAX_COLOUR_INDICES            : out PHIGS_NATURAL;
   MAX_VIEW_INDICES              : out PHIGS_NATURAL);
```

INQUIRE DYNAMICS OF STRUCTURES

```
procedure INQ_DYNAMICS_OF_STRUCTURES
  (TYPE_OF_WS                 : in  WS_TYPE;
   ERROR_INDICATOR            : out ERROR_NUMBER;
   CONTENT_MODIFICATION       : out DYNAMIC_MODIFICATION;
   POST                       : out DYNAMIC_MODIFICATION;
   UNPOST                     : out DYNAMIC_MODIFICATION;
   DELETE                     : out DYNAMIC_MODIFICATION;
   REFERENCE_MODIFICATION     : out DYNAMIC_MODIFICATION);
```

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

```
procedure INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES
  (TYPE_OF_WS          : in  WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   LOCATOR             : out PHIGS_NATURAL;
   STROKE              : out PHIGS_NATURAL;
   VALUATOR            : out PHIGS_NATURAL;
   CHOICE              : out PHIGS_NATURAL;
   PICK                : out PHIGS_NATURAL;
   STRING              : out PHIGS_NATURAL);
```

INQUIRE DEFAULT LOCATOR DEVICE DATA 3

```
procedure INQ_DEFAULT_LOCATOR_DEVICE_DATA
  (TYPE_OF_WS                 : in  WS_TYPE;
   DEVICE                     : in  LOCATOR_DEVICE_NUMBER;
   ERROR_INDICATOR            : out ERROR_NUMBER;
   INITIAL_POSITION           : out WC.POINT_3;
   LIST_OF_PROMPT_ECHO_TYPES  : out LOCATOR_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_VOLUME                : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD                : out LOCATOR_DATA_RECORD);
```

INQUIRE DEFAULT LOCATOR DEVICE DATA

```
procedure INQ_DEFAULT_LOCATOR_DEVICE_DATA
  (TYPE_OF_WS                 : in  WS_TYPE;
   DEVICE                     : in  LOCATOR_DEVICE_NUMBER;
   ERROR_INDICATOR            : out ERROR_NUMBER;
   INITIAL_POSITION           : out WC.POINT_2;
   LIST_OF_PROMPT_ECHO_TYPES  : out LOCATOR_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_AREA                  : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD                : out LOCATOR_DATA_RECORD);
```

INQUIRE DEFAULT STROKE DEVICE DATA 3

```
procedure INQ_DEFAULT_STROKE_DEVICE_DATA
  (TYPE_OF_WS                    : in  WS_TYPE;
   DEVICE                        : in  STROKE_DEVICE_NUMBER;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   MAX_BUFFER_SIZE               : out PHIGS_NATURAL;
   LIST_OF_PROMPT_ECHO_TYPES     : out STROKE_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_VOLUME                   : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD                   : out STROKE_DATA_RECORD);
```

INQUIRE DEFAULT STROKE DEVICE DATA

```
procedure INQ_DEFAULT_STROKE_DEVICE_DATA
  (TYPE_OF_WS                    : in  WS_TYPE;
   DEVICE                        : in  STROKE_DEVICE_NUMBER;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   MAX_BUFFER_SIZE               : out PHIGS_NATURAL;
   LIST_OF_PROMPT_ECHO_TYPES     : out STROKE_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_AREA                     : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD                   : out STROKE_DATA_RECORD);
```

INQUIRE DEFAULT VALUATOR DEVICE DATA 3

```
procedure INQ_DEFAULT_VALUATOR_DEVICE_DATA
  (TYPE_OF_WS                    : in  WS_TYPE;
   DEVICE                        : in  VALUATOR_DEVICE_NUMBER;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   INITIAL_VALUE                 : out VALUATOR_VALUE;
   LIST_OF_PROMPT_ECHO_TYPES     : out VALUATOR_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_VOLUME                   : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD                   : out VALUATOR_DATA_RECORD);
```

INQUIRE DEFAULT VALUATOR DEVICE DATA

```
procedure INQ_DEFAULT_VALUATOR_DEVICE_DATA
  (TYPE_OF_WS                    : in  WS_TYPE;
   DEVICE                        : in  VALUATOR_DEVICE_NUMBER;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   INITIAL_VALUE                 : out VALUATOR_VALUE;
   LIST_OF_PROMPT_ECHO_TYPES     : out VALUATOR_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_AREA                     : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD                   : out VALUATOR_DATA_RECORD);
```

INQUIRE DEFAULT CHOICE DEVICE DATA 3

```
procedure INQ_DEFAULT_CHOICE_DEVICE_DATA
  (TYPE_OF_WS                    : in  WS_TYPE;
   DEVICE                        : in  CHOICE_DEVICE_NUMBER;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   MAX_CHOICES                   : out CHOICE_NUMBER;
   LIST_OF_PROMPT_ECHO_TYPES     : out CHOICE_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_VOLUME                   : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD                   : out CHOICE_DATA_RECORD);
```

INQUIRE DEFAULT CHOICE DEVICE DATA

```
procedure INQ_DEFAULT_CHOICE_DEVICE_DATA
  (TYPE_OF_WS                    : in  WS_TYPE;
   DEVICE                        : in  CHOICE_DEVICE_NUMBER;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   MAX_CHOICES                   : out CHOICE_NUMBER;
   LIST_OF_PROMPT_ECHO_TYPES     : out CHOICE_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_AREA                     : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD                   : out CHOICE_DATA_RECORD);
```

INQUIRE DEFAULT PICK DEVICE DATA 3

```
procedure INQ_DEFAULT_PICK_DEVICE_DATA
  (TYPE_OF_WS                    : in  WS_TYPE;
   DEVICE                        : in  PICK_DEVICE_NUMBER;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   LIST_OF_PROMPT_ECHO_TYPES     : out PICK_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_VOLUME                   : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD                   : out PICK_DATA_RECORD);
```

INQUIRE DEFAULT PICK DEVICE DATA

```
procedure INQ_DEFAULT_PICK_DEVICE_DATA
  (TYPE_OF_WS                    : in  WS_TYPE;
   DEVICE                        : in  PICK_DEVICE_NUMBER;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   LIST_OF_PROMPT_ECHO_TYPES     : out PICK_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_AREA                     : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD                   : out PICK_DATA_RECORD);
```

INQUIRE DEFAULT STRING DEVICE DATA 3

```
procedure INQ_DEFAULT_STRING_DEVICE_DATA
  (TYPE_OF_WS                    : in  WS_TYPE;
   DEVICE                        : in  STRING_DEVICE_NUMBER;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   MAX_STRING_BUFFER_SIZE        : out PHIGS_NATURAL;
   LIST_OF_PROMPT_ECHO_TYPES     : out STRING_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_VOLUME                   : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD                   : out STRING_DATA_RECORD);
```

INQUIRE DEFAULT STRING DEVICE DATA

```
procedure INQ_DEFAULT_STRING_DEVICE_DATA
  (TYPE_OF_WS                    : in  WS_TYPE;
   DEVICE                        : in  STRING_DEVICE_NUMBER;
   ERROR_INDICATOR               : out ERROR_NUMBER;
   MAX_STRING_BUFFER_SIZE        : out PHIGS_NATURAL;
   LIST_OF_PROMPT_ECHO_TYPES     : out STRING_PROMPT_ECHO_TYPES.LIST_OF;
   ECHO_AREA                     : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD                   : out STRING_DATA_RECORD);
```

INQUIRE SET OF WORKSTATIONS TO WHICH POSTED

```
procedure INQ_SET_OF_WS_TO_WHICH_POSTED
  (STRUCTURE_IDENTIFIER : in  STRUCTURE_ID;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   LIST_OF_WS           : out WS_IDS.LIST_OF);
```

INQUIRE OPEN STRUCTURE

```
procedure INQ_OPEN_STRUCTURE
  (ERROR_INDICATOR      : out ERROR_NUMBER;
   STATUS               : out OPEN_STRUCTURE_STATUS;
   STRUCTURE_IDENTIFIER : out STRUCTURE_ID);
```

INQUIRE ELEMENT POINTER

```
procedure INQ_ELEMENT_POINTER
  (ERROR_INDICATOR : out ERROR_NUMBER;
   POSITION        : out RETURNED_ELEMENT_POSITION);
```

INQUIRE CURRENT ELEMENT TYPE AND SIZE

```
procedure INQ_CURRENT_ELEMENT_TYPE_AND_SIZE
  (ERROR_INDICATOR : out ERROR_NUMBER;
   ELEMENT_TYPE    : out STRUCTURE_ELEMENT_TYPE;
   SIZE            : out PHIGS_NATURAL);
```

INQUIRE CURRENT ELEMENT CONTENT

```
procedure INQ_CURRENT_ELEMENT_CONTENT
  (ERROR_INDICATOR : out ERROR_NUMBER;
   ELEMENT_RECORD  : in out STRUCTURE_ELEMENT_RECORD);
```

INQUIRE ELEMENT TYPE AND SIZE

```
procedure INQ_ELEMENT_TYPE_AND_SIZE
  (STRUCTURE_IDENTIFIER : in  STRUCTURE_ID;
   POSITION             : in  ELEMENT_POSITION;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   ELEMENT_TYPE         : out STRUCTURE_ELEMENT_TYPE;
   SIZE                 : out PHIGS_NATURAL);
```

INQUIRE ELEMENT CONTENT

```
procedure INQ_ELEMENT_CONTENT
  (STRUCTURE_IDENTIFIER : in     STRUCTURE_ID;
   POSITION             : in     ELEMENT_POSITION;
   ERROR_INDICATOR      :    out ERROR_NUMBER;
   ELEMENT_RECORD       : in out STRUCTURE_ELEMENT_RECORD);
```

INQUIRE STRUCTURE STATUS

```
procedure INQ_STRUCTURE_STATUS
  (STRUCTURE_IDENTIFIER : in  STRUCTURE_ID;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   STATUS               : out STRUCTURE_STATUS);
```

INQUIRE PATHS TO ANCESTORS

```
procedure INQ_PATHS_TO_ANCESTORS
  (STRUCTURE_IDENTIFIER : in  STRUCTURE_ID;
   ORDER                : in  PATH_ORDER;
   DEPTH                : in  PATH_DEPTH;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   LIST_OF_PATHS        : out REFERENCE_PATHS.LIST_OF);
```

INQUIRE PATHS TO DESCENDANTS

```
procedure INQ_PATHS_TO_DESCENDANTS
  (STRUCTURE_IDENTIFIER : in STRUCTURE_ID;
   ORDER                : in PATH_ORDER;
   DEPTH                : in PATH_DEPTH;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   LIST_OF_PATHS        : out REFERENCE_PATHS.LIST_OF);
```

ELEMENT SEARCH

```
procedure ELEMENT_SEARCH
  (STRUCTURE_IDENTIFIER : in STRUCTURE_ID;
   START_POSITION       : in ELEMENT_POSITION;
   DIRECTION            : in SEARCH_DIRECTION;
   INCLUSION_SET        : in ELEMENT_TYPES.LIST_OF;
   EXCLUSION_SET        : in ELEMENT_TYPES.LIST_OF;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   SEARCH_STATUS        : out SEARCH_STATUS_INDICATOR;
   FOUND_POSITION       : out RETURNED_ELEMENT_POSITION);
```

INCREMENTAL SPATIAL SEARCH 3

```
procedure INCREMENTAL_SPATIAL_SEARCH
  (SEARCH_REFERENCE_POINT : in WC.POINT_3;
   SEARCH_DISTANCE        : in WC.MAGNITUDE;
   STARTING_PATH          : in REFERENCE_PATH;
   MODELLING_CLIPPING     : in CLIPPING_INDICATOR;
   SEARCH_CEILING_INDEX   : in ELEMENT_POSITION;
   LIST_OF_NORMAL_FILTERS : in NAME_SET_FILTERS.LIST_OF;
   LIST_OF_INVERTED_FILTERS : in NAME_SET_FILTERS.LIST_OF;
   ERROR_INDICATOR        : out ERROR_NUMBER;
   FOUND_PATH             : out REFERENCE_PATH);
```

INCREMENTAL SPATIAL SEARCH

```
procedure INCREMENTAL_SPATIAL_SEARCH
  (SEARCH_REFERENCE_POINT : in WC.POINT_2;
   SEARCH_DISTANCE        : in WC.MAGNITUDE;
   STARTING_PATH          : in REFERENCE_PATH;
   MODELLING_CLIPPING     : in CLIPPING_INDICATOR;
   SEARCH_CEILING_INDEX   : in ELEMENT_POSITION;
   LIST_OF_NORMAL_FILTERS : in NAME_SET_FILTERS.LIST_OF;
   LIST_OF_INVERTED_FILTERS : in NAME_SET_FILTERS.LIST_OF;
   ERROR_INDICATOR        : out ERROR_NUMBER;
   FOUND_PATH             : out REFERENCE_PATH);
```

INQUIRE INPUT QUEUE OVERFLOW

```
procedure INQ_INPUT_QUEUE_OVERFLOW
  (ERROR_INDICATOR : out ERROR_NUMBER;
   WS              : out WS_ID;
   DEVICE          : out EVENT_QUEUE_DEVICE_NUMBER);
```

INQUIRE ERROR HANDLING MODE

```
procedure INQ_ERROR_HANDLING_MODE
  (ERROR_INDICATOR : out ERROR_NUMBER;
   MODE            : out ERROR_HANDLING_MODE);
```

5.12 Error control functions

EMERGENCY CLOSE PHIGS

```
procedure EMERGENCY_CLOSE_PHIGS;
```

ERROR HANDLING

```
procedure ERROR_HANDLING
  (ERROR_INDICATOR : in ERROR_NUMBER;
   SUBPROGRAM      : in SUBPROGRAM_NAME;
   ERROR_FILE      : in FILE_ID := PHIGS_STRING(DEFAULT_ERROR_FILE));

   -- The procedure ERROR_HANDLING is defined as a library
   -- subprogram, and is not declared within Package PHIGS.
```

ERROR LOGGING

```
procedure ERROR_LOGGING
  (ERROR_INDICATOR : in ERROR_NUMBER;
   SUBPROGRAM      : in SUBPROGRAM_NAME;
   ERROR_FILE      : in FILE_ID := PHIGS_STRING(DEFAULT_ERROR_FILE));
```

SET ERROR HANDLING MODE

```
procedure SET_ERROR_HANDLING_MODE
  (MODE : in ERROR_HANDLING_MODE);
```

5.13 Special interface functions

ESCAPE

Escape functions are bound in Ada as separate procedures for each unique type of escape provided by the implementation, each with a formal parameter list appropriate to the procedure implemented. ESCAPE names and parameters are registered in the ISO International Register of Graphical Items which is maintained by the registration authority. All supported registered ESCAPE procedures will be in a library package named PHIGS_ESCAPE.

The minimal support for ESCAPE's required by this standard is that defined in the following specification for the PHIGS_ESCAPE package. This package is part of package PHIGS and shall be visible. The minimal support provides an interface to registered and unregistered ESCAPE's not supported by the implementation. Unsupported ESCAPE's may be invoked using the procedure GENERALIZED_ESC whose external specification is shown.

```
with PHIGS_TYPES;
use  PHIGS_TYPES;
package PHIGS_ESCAPE is
  type ESC_INTEGER_ARRAY is array (SMALL_NATURAL range <>) of
                                  PHIGS_INTEGER;
  type ESC_FLOAT_ARRAY   is array (SMALL_NATURAL range <>) of
                                  GENERAL_FLOAT_PARAMETER;
  type ESC_STRING_ARRAY  is array (SMALL_NATURAL range <>) of
                                  PHIGS_STRING(1..80);

  type ESC_DATA_RECORD (NUM_OF_INTEGERS : SMALL_NATURAL := 0;
                        NUM_OF_REALS    : SMALL_NATURAL := 0;
                        NUM_OF_STRINGS  : SMALL_NATURAL := 0) is
~@

    record
      INTEGER_ARRAY : ESC_INTEGER_ARRAY (1..NUM_OF_INTEGERS);
      REAL_ARRAY    : ESC_FLOAT_ARRAY   (1..NUM_OF_REALS);
      ESC_STRINGS   : ESC_STRING_ARRAY  (1..NUM_OF_STRINGS);
    end record;


  procedure GENERALIZED_ESC
    (ESC_NAME      : in  ESCAPE_ID;
     ESC_DATA_IN   : in  ESC_DATA_RECORD;
     ESC_DATA_OUT  : out ESC_DATA_RECORD);

end PHIGS_ESCAPE;
```

Each registered ESCAPE procedure supported by the implementation will be included in package PHIGS_ESCAPE (which is part of package PHIGS) as a separate procedure using the following naming convention which results in references to the ESCAPE by the expression PHIGS_ESCAPE.<name of the ESCAPE procedure>. Specific names will be assigned when the ESCAPE is registered.

```
procedure <name of the ESCAPE procedure> (<parameters as required>);
```

Each unregistered ESCAPE procedure will be a library package using the following naming convention:

```
package PHIGS_UESC_<name of the escape procedure> is
  procedure ESC;
    -- Ada code for UESC procedure

  -- The only procedure name used in the package
  -- will be ESC

end PHIGS_UESC_<name of the escape procedure>;
```

5.14 Additional Functions

5.14.1 Subprograms for Manipulating Input Data Records

The procedures and functions defined in this section are those that are necessary for constructing and inquiring the input data records, declared as private types in this binding for each of the six classes of input devices defined by the PHIGS specification -- the Locator, Stroke, Valuator, Choice, Pick and String logical devices. The procedures listed here are used to construct the data records for each of the registered prompt and echo types of a device class to be used for initializing a particular input device. Assorted functions are also provided so that an application may examine the parts of the data record which are defined by PHIGS. Any implementation specific information in the data records is kept private and unavailable.

A PHIGS_ERROR is generated if any of the below procedures are used incorrectly. For example, if an illegal prompt and echo type is used for a build procedure, then error number 2500 is logged onto the error file.

To implement implementation dependent and registered items an implementation may provide additional overloaded versions of the BUILD procedures in this section and additional functions for extracting information from the private data records.

-- Locator Data Record Operations

```
procedure BUILD_LOCATOR_DATA_RECORD
  (PROMPT_ECHO_TYPE : in LOCATOR_PROMPT_ECHO_TYPE;
   DATA_RECORD      : out LOCATOR_DATA_RECORD);

-- Constructs and returns a locator data record for the
-- locator prompt and echo types 1, 2, 3, and 6.

procedure BUILD_LOCATOR_DATA_RECORD
  (PROMPT_ECHO_TYPE : in LOCATOR_PROMPT_ECHO_TYPE;
   CONTENTS         : in LINE_DATA;
   DATA_RECORD      : out LOCATOR_DATA_RECORD);

-- Constructs and returns a locator data record for prompt and
-- echo type 4 or for echo type 5 when its attributes are
-- specified by Polyline attributes.

procedure BUILD_LOCATOR_DATA_RECORD
  (PROMPT_ECHO_TYPE : in LOCATOR_PROMPT_ECHO_TYPE;
   CONTENTS         : in INTERIOR_DATA;
```

```
      DATA_RECORD          : out LOCATOR_DATA_RECORD);

   -- Constructs and returns a locator data record for the locator
   -- prompt and echo type 5 when its attributes are specified only by
   -- interior attributes.

   procedure BUILD_LOCATOR_DATA_RECORD
     (PROMPT_ECHO_TYPE  : in  LOCATOR_PROMPT_ECHO_TYPE;
      INTERIOR_CONTENTS : in  INTERIOR_DATA;
      EDGE_CONTENTS     : in  EDGE_DATA;
      DATA_RECORD       : out LOCATOR_DATA_RECORD);

   -- Constructs and returns a locator data record for the locator
   -- prompt and echo type 5 when its attributes are specified by
   -- both interior and edge attributes.

   function LOCATOR_ATTRIBUTES_USED
               (DATA_RECORD  : in LOCATOR_DATA_RECORD)
     return POLYLINE_FILL_AREA_CONTROL_FLAG;

   -- Returns which attribute sets are stored in the data record for
   -- locator prompt and echo type 5.

   function LINE_ATTRIBUTES (DATA_RECORD : in LOCATOR_DATA_RECORD)
     return LINE_DATA;

   -- Returns the Polyline attribute information stored in the data
   -- record for locator prompt and echo types 4 or 5.

   function INTERIOR_ATTRIBUTES (DATA_RECORD : in  LOCATOR_DATA_RECORD)
     return INTERIOR_DATA;

   -- Returns the Interior attribute information stored in the data
   -- record for locator prompt and echo type 5.

   function EDGE_ATTRIBUTES (DATA_RECORD : in  LOCATOR_DATA_RECORD)
     return EDGE_DATA;

   -- Returns the Edge attribute information stored in the data
   -- record for locator prompt and echo type 5.

-- Stroke Data Record Operations

   procedure BUILD_STROKE_DATA_RECORD
     (PROMPT_ECHO_TYPE : in  STROKE_PROMPT_ECHO_TYPE;
      BUFFER_SIZE      : in  PHIGS_POSITIVE;
      POSITION         : in  PHIGS_POSITIVE;
      INTERVAL         : in  WC.SIZE_2;
      TIME             : in  DURATION;
      DATA_RECORD      : out STROKE_DATA_RECORD);

   -- Constructs and returns a stroke data record for the stroke
   -- prompt and echo types 1 and 2.

   procedure BUILD_STROKE_DATA_RECORD
     (PROMPT_ECHO_TYPE : in  STROKE_PROMPT_ECHO_TYPE;
      BUFFER_SIZE      : in  PHIGS_POSITIVE;
      POSITION         : in  PHIGS_POSITIVE;
      INTERVAL         : in  WC.SIZE_2;
```

```
      TIME                  : in  DURATION;
      CONTENTS              : in  MARKER_DATA;
      DATA_RECORD           : out STROKE_DATA_RECORD);

   -- Constructs and returns a stroke data record for the stroke
   -- prompt and echo type 3.

   procedure BUILD_STROKE_DATA_RECORD
     (PROMPT_ECHO_TYPE : in  STROKE_PROMPT_ECHO_TYPE;
      BUFFER_SIZE      : in  PHIGS_POSITIVE;
      POSITION         : in  PHIGS_POSITIVE;
      INTERVAL         : in  WC.SIZE_2;
      TIME             : in  DURATION;
      CONTENTS         : in  LINE_DATA;
      DATA_RECORD      : out STROKE_DATA_RECORD);

   -- Constructs and returns a stroke data record for the stroke
   -- prompt and echo type 4.

   function BUFFER_SIZE (DATA_RECORD : in STROKE_DATA_RECORD)
     return PHIGS_POSITIVE;

   -- Returns the size of the input stroke buffer stored in the data
   -- record for the stroke prompt and echo types 1, 2, 3, and 4.

   function POSITION (DATA_RECORD : in STROKE_DATA_RECORD)
     return PHIGS_POSITIVE;

   -- Returns the editing position within the stroke input buffer
   -- stored in the data record for the stroke prompt and echo
   -- types 1, 2, 3, and 4.

   function INTERVAL (DATA_RECORD : in STROKE_DATA_RECORD)
     return WC.SIZE_2;

   -- Returns the interval value stored in the stroke data record
   -- for the prompt and echo types 1, 2, 3, and 4.

   function TIME (DATA_RECORD : in STROKE_DATA_RECORD)
     return DURATION;

   -- Returns the measuring time for sampling stroke input stored in
   -- the stroke data record for the stroke prompt and echo types
   -- 1, 2, 3, and 4.

   function MARKER_ATTRIBUTES (DATA_RECORD : in STROKE_DATA_RECORD)
     return MARKER_DATA;

   -- Returns the Polymarker attributes used to echo the stroke input
   -- stored in the data record for prompt and echo type 3.

   function LINE_ATTRIBUTES (DATA_RECORD : in STROKE_DATA_RECORD)
     return LINE_DATA;

   -- Returns the Polyline attributes used to echo the stroke input
   -- stored in the data record for stroke prompt and echo type 4.

-- Valuator Data Record Operations
```

```
   procedure BUILD_VALUATOR_DATA_RECORD
     (PROMPT_ECHO_TYPE  : in VALUATOR_PROMPT_ECHO_TYPE;
      LOW_VALUE         : in VALUATOR_VALUE;
      HIGH_VALUE        : in VALUATOR_VALUE;
      DATA_RECORD       : out VALUATOR_DATA_RECORD);

   -- Constructs and returns a valuator data record for valuator
   -- prompt and echo type 1.

   function HIGH_VALUE (DATA_RECORD : in VALUATOR_DATA_RECORD)
     return VALUATOR_VALUE;

   -- Returns the high value for the valuator stored in the valuator
   -- data record for the prompt and echo types 1, 2, and 3.

   function LOW_VALUE (DATA_RECORD : in VALUATOR_DATA_RECORD)
     return VALUATOR_VALUE;

   -- Returns the low value for the valuator stored in the valuator
   -- data record for the prompt and echo types 1, 2, and 3.

-- Choice Data Record Operations

   procedure BUILD_CHOICE_DATA_RECORD
     (PROMPT_ECHO_TYPE  : in CHOICE_PROMPT_ECHO_TYPE;
      DATA_RECORD       : out CHOICE_DATA_RECORD);

   -- Constructs and returns a choice data record for the choice
   -- prompt and echo type 1.

   procedure BUILD_CHOICE_DATA_RECORD
     (PROMPT_ECHO_TYPE  : in CHOICE_PROMPT_ECHO_TYPE;
      ARRAY_OF_PROMPTS  : in CHOICE_PROMPTS.LIST_OF;
      DATA_RECORD       : out CHOICE_DATA_RECORD);

   -- Constructs and returns a choice data record for the choice
   -- prompt and echo type 2.

   procedure BUILD_CHOICE_DATA_RECORD
     (PROMPT_ECHO_TYPE  : in CHOICE_PROMPT_ECHO_TYPE;
      ARRAY_OF_STRINGS  : in CHOICE_PROMPT_STRING_LIST;
      DATA_RECORD       : out CHOICE_DATA_RECORD);

   -- Constructs and returns a choice data record for the choice
   -- prompt and echo types 3 and 4.

   procedure BUILD_CHOICE_DATA_RECORD
     (PROMPT_ECHO_TYPE      : in CHOICE_PROMPT_ECHO_TYPE;
      STRUCTURE_IDENTIFIER  : in STRUCTURE_ID;
      LIST_OF_PICK_IDS      : in PICK_IDS.LIST_OF;
      DATA_RECORD           : out CHOICE_DATA_RECORD);

   -- Constructs and returns a choice data record for the choice
   -- prompt and echo type 5.

   function ARRAY_OF_PROMPTS (DATA_RECORD : in CHOICE_DATA_RECORD)
     return CHOICE_PROMPTS.LIST_OF;

   -- Returns the array of prompts stored in the choice data record
```

```
      -- for the prompt and echo type 2.

      function ARRAY_OF_STRINGS (DATA_RECORD : in CHOICE_DATA_RECORD)
        return CHOICE_PROMPT_STRING_LIST;

      -- Returns the array of prompt strings stored in the choice data
      -- record for the prompt and echo types 3 and 4.

      function STRUCTURE_IDENTIFIER (DATA_RECORD : in CHOICE_DATA_RECORD)
        return STRUCTURE_ID;

      -- Returns the structure identifier stored in the choice data
      -- record for the prompt and echo type 5.

      function LIST_OF_PICK_IDS (DATA_RECORD : in CHOICE_DATA_RECORD)
        return PICK_IDS.LIST_OF;

      -- Returns the list of pick identifiers stored in the choice data
      -- record for the prompt and echo type 5.

-- Pick Data Record Operations.

      procedure BUILD_PICK_DATA_RECORD
        (PROMPT_ECHO_TYPE : in PICK_PROMPT_ECHO_TYPE;
         DATA_RECORD      : out PICK_DATA_RECORD);

      -- Constructs and returns a pick data record for the pick
      -- prompt and echo types 1, 2, and 3.

-- String Data Record Operations

      procedure BUILD_STRING_DATA_RECORD
        (PROMPT_ECHO_TYPE       : in STRING_PROMPT_ECHO_TYPE;
         INPUT_BUFFER_SIZE      : in PHIGS_NATURAL;
         INITIAL_CURSOR_POSITION: in PHIGS_NATURAL;
         DATA_RECORD            : out STRING_DATA_RECORD);

      -- Constructs and returns a string data record for the string
      -- prompt and echo type 1.

      function INPUT_BUFFER_SIZE (DATA_RECORD : in STRING_DATA_RECORD)
        return PHIGS_NATURAL;

      -- Returns the size of the buffer used for storing string input
      -- stored in the string data record for prompt and echo type 1.

      function INITIAL_CURSOR_POSITION (DATA_RECORD : in STRING_DATA_RECORD)
        return PHIGS_NATURAL;

      -- Returns the initial cursor position for string input stored in
      -- the string data record for prompt and echo type 1.
```

5.14.2 PHIGS Generic Coordinate System Package

The generic package declared in this section is the specification of a generic Cartesian Coordinate System for PHIGS. This package is instantiated in package PHIGS_TYPES once for each of Modelling Coordinates,

World Coordinates, View Reference Coordinates, Normalized Projection Coordinates, and Device Coordinates. The package defines the representation of the following data types:

POINT_3
POINT_2
POINT_LIST_3
POINT_LIST_2
ACCESS_POINT_LIST_3
ACCESS_POINT_LIST_2
LIST_OF_POINT_LIST_3
LIST_OF_POINT_LIST_2
ACCESS_LIST_OF_POINT_LIST_3
ACCESS_LIST_OF_POINT_LIST_2
VECTOR_3
VECTOR_PAIR_3
VECTOR_2
VECTOR_PAIR_2
RECTANGULAR_REGION_3
RECTANGULAR_REGION_2
HALF_SPACE_3
HALF_SPACE_2
HALF_SPACE_LIST_3
HALF_SPACE_LIST_2
ACCESS_HALF_SPACE_LIST_3
ACCESS_HALF_SPACE_LIST_2

Also defined is a MAGNITUDE type for measuring lengths within a coordinate space. The type SIZE measures lengths parallel to both the axes, and the RANGE_OF_MAGNITUDES type specifies two lengths within a coordinate system, a minimum and maximum for values such as the range of Character Heights available on a device.

```
generic

   type COORDINATE_COMPONENT_TYPE is digits <>;

package PHIGS_COORDINATE_SYSTEM is


   type POINT_3 is
     record
       X : COORDINATE_COMPONENT_TYPE;
       Y : COORDINATE_COMPONENT_TYPE;
       Z : COORDINATE_COMPONENT_TYPE;
     end record;


   type POINT_2 is
     record
       X : COORDINATE_COMPONENT_TYPE;
       Y : COORDINATE_COMPONENT_TYPE;
     end record;

   type POINT_LIST_3 is array (POSITIVE range <>) of POINT_3;

   type POINT_LIST_2 is array (POSITIVE range <>) of POINT_2;
```

```
type ACCESS_POINT_LIST_3 is access POINT_LIST_3;

type ACCESS_POINT_LIST_2 is access POINT_LIST_3;


type LIST_OF_POINT_LIST_3 is
  array (POSITIVE range <>) of ACCESS_POINT_LIST_3;

type LIST_OF_POINT_LIST_2 is
  array (POSITIVE range <>) of ACCESS_POINT_LIST_2;


type ACCESS_LIST_OF_POINT_LIST_3 is access LIST_OF_POINT_LIST_3;

type ACCESS_LIST_OF_POINT_LIST_2 is access LIST_OF_POINT_LIST_2;


type VECTOR_3 is new POINT_3;

type VECTOR_PAIR_3 is array (1..2) of VECTOR_3;

type VECTOR_2 is new POINT_2;

type VECTOR_PAIR_2 is array (1..2) of VECTOR_2;

type RECTANGULAR_REGION_3 is
  record
    XMIN  : COORDINATE_COMPONENT_TYPE;
    XMAX  : COORDINATE_COMPONENT_TYPE;
    YMIN  : COORDINATE_COMPONENT_TYPE;
    YMAX  : COORDINATE_COMPONENT_TYPE;
    ZMIN  : COORDINATE_COMPONENT_TYPE;
    ZMAX  : COORDINATE_COMPONENT_TYPE;
  end record;


type RECTANGULAR_REGION_2 is
  record
    XMIN  : COORDINATE_COMPONENT_TYPE;
    XMAX  : COORDINATE_COMPONENT_TYPE;
    YMIN  : COORDINATE_COMPONENT_TYPE;
    YMAX  : COORDINATE_COMPONENT_TYPE;
  end record;


type HALF_SPACE_3 is
  record
    REFERENCE_POSITION  : POINT_3;
    ACCEPTANCE_NORMAL : VECTOR_3;
  end record;


type HALF_SPACE_2 is
  record
    REFERENCE_POSITION  : POINT_2;
    ACCEPTANCE_NORMAL : VECTOR_2;
  end record;
```

```
  type HALF_SPACE_LIST_3 is
    array (POSITIVE range <>) of HALF_SPACE_3;

  type HALF_SPACE_LIST_2 is
    array (POSITIVE range <>) of HALF_SPACE__2;


  type ACCESS_HALF_SPACE_LIST_3 is access HALF_SPACE_LIST_3;

  type ACCESS_HALF_SPACE_LIST_2 is access HALF_SPACE_LIST_2;


  type MAGNITUDE_BASE_TYPE is digits PHIGS_PRECISION;

  subtype MAGNITUDE is
    MAGNITUDE_BASE_TYPE range
        COORDINATE_COMPONENT_TYPE'SAFE_SMALL ..
        COORDINATE_COMPONENT_TYPE'SAFE_LARGE;


  type SIZE_3 is
    record
      XAXIS  : MAGNITUDE;
      YAXIS  : MAGNITUDE;
      ZAXIS  : MAGNITUDE;
    end record;


  type SIZE_2 is
    record
      XAXIS  : MAGNITUDE;
      YAXIS  : MAGNITUDE;
    end record;


  type RANGE_OF_MAGNITUDES is
    record
      MIN   : MAGNITUDE;
      MAX   : MAGNITUDE;
    end record;

end PHIGS_COORDINATE_SYSTEM;
```

5.14.3 PHIGS Generic List Utility Package

The generic package PHIGS_LIST_UTILITIES is instantiated in the PHIGS_TYPES package once for each of several LIST_OF types and their manipulation subprograms. Each LIST_OF type contains different element type values.

The LIST_OF type is declared as a private type in PHIGS_LIST_UTILITIES to restrict the operations on the LIST_OF type that are available to outside program units. The LIST_OF private type declaration includes a discriminant part that defines the current size of the lists. LIST_OF objects are declared as unconstrained objects (by using the default discriminant value) to allow dynamic modification of the list size.

A LIST_OF object is a sequence of element type values. Each element type value is associated with an index. Index values begin at one and increase in steps of one.

The size of a LIST_OF object is the number of element type values stored within it. A single element type value may be stored more than onece within a LIST_OF object. A LIST_OF object may be empty. The size of an empty LIST_OF object is zero. The maximum size of a LIST_OF object is given by the MAX_LIST_SIZE generic parameter. If this parameter is not specified in the instantiation, an implementation dependent value is used.

An adequate package specification is shown below. The comments within the package describe the functionality of the manipulation subprograms:

```
generic
   type ELEMENT_TYPE is private;

   MAX_LIST_SIZE : NATURAL := PHIGS_CONFIGURATION.DEFAULT_LIST_SIZE;


package PHIGS_LIST_UTILITIES is

   subtype LIST_SIZE is NATURAL range 0..MAX_LIST_SIZE;

   type LIST_OF (SIZE : LIST_SIZE := 0) is limited private;

   type LIST_VALUES is ARRAY (POSITIVE range <>) of ELEMENT_TYPE;


   function NULL_LIST return LIST_OF;

      -- This function returns an empty LIST_OF object. This list is
      -- intended primarily for use by implementors.


   function SIZE_OF_LIST (LIST : in LIST_OF) return NATURAL;

      -- This function returns the number of element type values stored
      -- in the list object.


   function IS_IN_LIST (ELEMENT  : in ELEMENT_TYPE;
                        LIST     : in LIST_OF) return BOOLEAN;

      -- This function returns the value TRUE if the element parameter
      -- value is in the list object. Otherwise it returns the value
      -- FALSE.


   function LIST_ELEMENT (INDEX  : in POSITIVE;
                          LIST   : in  LIST_OF) return ELEMENT_TYPE;

      -- This function returns the element value in the list object that
      -- has an associated index value equal to the index parameter. The
      -- PHIGS_ERROR 2502 is generated if the index parameter exceeds the
      -- current size of the list parameter object.
```

```
   function LIST (VALUES : in LIST_VALUES) return LIST_OF;

      -- This function returns a valid LIST_OF object.  If the VALUES
      -- parameter is a null array, an empty LIST_OF object is returned.
      -- If the values parameter is not null, this function returns a
      -- LIST_OF object containing all the values in the VALUES parameter.
      -- The PHIGS_ERROR 2502 is generated if the number of element values
      -- exceeds the specified maximum size of the LIST_OF object.


   procedure ADD_TO_LIST (ELEMENT  : in ELEMENT_TYPE;
                          LIST     : in out LIST_OF);

      -- This procedure stores the element parameter value in the list
      -- parameter object, and increases the size of the list object
      -- by one.  An index value equal to the incremented list size
      -- is associated with the stored element value.  The ADD_TO_LIST
      -- procedure will generate PHIGS_ERROR 2502 if it is called when
      -- the list parameter has a size equal to the specified maximum
      -- size.  If desired, the user can ensure duplicate values are not
      -- stored.  This is accomplised by calling ADD_TO_LIST with a
      -- particular element value only if the function IS_IN_LIST returns
      -- false for that element value.


   procedure DELETE_FROM_LIST (ELEMENT  : in ELEMENT_TYPE;
                               LIST     : in out LIST_OF);

      -- If the list parameter object does not contain the element
      -- parameter value, this procedure has no effect.  Otherwise, the
      -- first occurrence of the element value is deleted.  The size of
      -- the list object is decreased by one, and the indices associated
      -- with the remaining element values are adjusted so that the indices
      -- begin at one and increment in steps of one.  If desired, the user
      -- can delete all occurrences of an element value.  This is accomplished
      -- by calling DELETE_FROM_LIST repeatedly with a particular element
      -- value while the function IS_IN_LIST returns TRUE for that value.


private

   -- The declaration of the LIST_OF type is implementation dependent.
   -- However, the operations implicitly declared by the LIST_OF declaration,
   -- including both assignment and the predefined comparison for equality
   -- and inequality, shall produce the correct results.  This requirement
   -- precludes the use of access types for the implementation of the
   -- LIST_OF type.  The recommended implementation is given below:
   --
   --       type LIST_OF (SIZE : LIST_SIZE := 0) is
   --         record
   --           ELEMENTS : LIST_VALUES(1..SIZE);
   --         end record;
   --
   -- Note that declaring unconstrained LIST_OF objects by using the default
   -- discriminant value allows dynamic modification of the size of the
   -- element array.

end PHIGS_LIST_UTILITIES;
```

5.14.4 PHIGS Name Set Facility Package

The types, procedures, and functions defined in this section are those necessary for constructing and inquiring name sets, declared as private types in this binding. The procedures listed here are used to construct the data items of type NAME_SET and NAME_SET_FILTER and to extract information about the classes contained included in NAME_SET data elements. Assorted functions are also provided so that an application may obtain information about the characteristics of one or more name sets.

The procedure COPY_NAME_SET and the function IS_EQUAL are provided as replacements for the normal Ada assignment and equality operators for NAME_SET objects. Conforming programs shall use these replacement functions. This allows the use of Ada access types to implement name sets at the discretion of the implementation.

PHIGS_ERROR 2501 is generated if any of the below procedures are used incorrectly.

```
with PHIGS_CONFIGURATION, PHIGS_LIST_UTILITIES;

package PHIGS_NAME_SET_FACILITY is

   -- PHIGS_CONFIGURATION.MAX_NAME_SUPPORTED

      -- An implementation dependent value which defines the maximum
      -- name set class supported.

   -- NAME SET

   type NAME_SET is private;

      -- Defines a name set data element.

   -- NAME_SET_ACCEPTANCE

   type NAME_SET_ACCEPTANCE is (REJECTED, ACCEPTED);

      -- Used to indicate the results of applying a name set
      -- against a filter pair.

   -- NAME

   type NAME is range 0..PHIGS_CONFIGURATION.MAX_NAME_SUPPORTED;

      -- Provides for names for each of the name set classes.


   -- NAMES

   package NAMES is
      new PHIGS_LIST_UTILITIES (NAME);

      -- Provides for lists of names.
```

```
-- NAME_SET_FILTER

type NAME_SET_FILTER is
  record
    INCLUSION   : NAME_SET;
    EXCLUSION   : NAME_SET;
  end record;

  -- Used to define name set pairs used as filters.


-- NAME_SET_MEMBERSHIP

type NAME_SET_MEMBERSHIP is (NON_MEMBER, MEMBER);

   -- Provides for indicating whether a name is a member of a
   -- name set.


-- Subprograms for Manipulating Name Sets

procedure BUILD_NAME_SET
  (MEMBERS  : in  NAMES.LIST_OF;
   SET      : out NAME_SET);

  -- Constructs a name set from a list of names.


procedure BUILD_NAME_SET_UNION
  (LEFT_SET   : in  NAME_SET;
   RIGHT_SET  : in  NAME_SET;
   UNION      : out NAME_SET);

  -- Constructs the logical union of two name sets.


procedure BUILD_NAME_SET_INTERSECTION
  (LEFT_SET       : in  NAME_SET;
   RIGHT_SET      : in  NAME_SET;
   INTERSECTION   : out NAME_SET);

  -- Constructs the logical intersection of two name sets.


procedure BUILD_NAME_SET_DIFFERENCE
  (LEFT_SET     : in  NAME_SET;
   RIGHT_SET    : in  NAME_SET;
   DIFFERENCE   : out NAME_SET);

  -- Constructs the logical difference of two name sets.


procedure COPY_NAME_SET
  (ORIGINAL  : in  NAME_SET;
   COPY      : out NAME_SET);

  -- Duplicates the provided name set.
```

```
function IS_EQUAL
  (LEFT_SET   : in NAME_SET;
   RIGHT_SET  : in NAME_SET)
  return BOOLEAN;

  -- Compares two name sets.


procedure EXTRACT_NAMES
  (SET         : in  NAME_SET;
   MEMBERS     : out NAMES.LIST_OF);

  -- Extracts the names from the provided name set.


procedure ADD_NAMES
  (MEMBERS  : in     NAMES.LIST_OF;
   SET      : in out NAME_SET);

  -- Adds names to the name set if not already there.


procedure REMOVE_NAMES
  (MEMBERS  : in     NAMES.LIST_OF;
   SET      : in out NAME_SET);

  -- Removes the names from the name set if they are there.


function APPLY_FILTER
  (SET      : in NAME_SET;
   FILTER   : in NAME_SET_FILTER)
  return NAME_SET_ACCEPTANCE;

  -- Applies a filter to a name set and indicates whether the name
  -- set passes through the filter.


function INQUIRE_MEMBERSHIP
  (MEMBER   : in NAME;
   SET      : in NAME_SET)
  return NAME_SET_MEMBERSHIP;

  -- Returns an indication of whether the indicated name is a member
  -- of the provided name set.


procedure DEALLOCATE
  (OBJECT   : in out NAME_SET);

  -- Deallocates name set objects.
```

```
private

   -- The following types define the specifications for private
   -- types used in the PHIGS_NAME_SET_FACILITY. Null records are
   -- used solely for the purposes of demonstrating compilability.
   -- An implementation would use some more appropriate declaration.

   type NAME_SET is
     record
       null;
     end record;

end PHIGS_NAME_SET_FACILITY;
```

5.14.5 Deallocation of structure element records

The contents of objects of type STRUCTURE_ELEMENT_RECORD and the various private types may contain data allocated from heap space. For example, objects of type STRUCTURE_ELEMENT_RECORD are used to store definitions of structure elements. The data types of these elements differ and vary according to the type of structure element. Some elements are as complex as variable length lists of variable length lists of points. The size and dimension of such lists is known only by the implementation at the time an application invokes the inquiry functions which use the STRUCTURE_ELEMENT_RECORD type. Similarly, objects of the various private types may also contain allocated data space which represent variable length lists. The implementation can allocate the exact amount of memory needed to return the variable length lists.

To avoid requiring that an application provide cumbersome code to deallocate all of the various possible cases, deallocation functions for each such data type are provided. These functions are all named DEALLOCATE and may be used by either the application or the implementation to free the memory space used by the corresponding records. For example, the implementation uses the DEALLOCATE function for objects of type STRUCTURE_ELEMENT_RECORD on successive calls to either INQ_ELEMENT_CONTENT or INQ_CURRENT_ELEMENT_CONTENT to free the space which may have been allocated in a previous call with the same object. The application will use the function to free memory after the last invocation of these inquiry functions. Once the memory has been released, the access variables will be set to the null value.

The REQUEST_STROKE, SAMPLE_STROKE, GET_STROKE, INQ_PATTERN_REPRESENTATION, INQ_STROKE_DEVICE_STATE, and INQ_PREDEFINED_PATTERN_REPRESENTATION functions also return data in the form of access types. Since the space for this data can easily be deallocated by the application by using the Ada predefined generic library procedure UNCHECKED_DEALLOCATION, no additional functions are provided in this binding.

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out CHOICE_DATA_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out ESCAPE_DATA_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out GDP_3_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out GDP_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out GSE_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out LOCATOR_DATA_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out METAFILE_DATA_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out NAME_SET)
  renames PHIGS_NAME_SET_FACILITY.DEALLOCATE;
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out PICK_DATA_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out STRING_DATA_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out STROKE_DATA_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out STRUCTURE_ELEMENT_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE
  (OBJECT : in out VALUATOR_DATA_RECORD);
```

5.14.6 Metafile Function Utilities

Metafile item data records are complex and are dependent on the encoding for the specified workstation. Record length depends on the number of data elements. PHIGS defines that the format is implementation defined. The item data record type should be private to allow direct manipulation of the record contents in order to have them efficiently processed.

The application programmer shall be able to write non-graphical data into the metafile. This can be provided by allowing character strings to be output. Numeric data can be converted to a string prior to output. A function is provided as a means to convert item data records into strings.

BUILD NEW METAFILE DATA RECORD

```
procedure BUILD_NEW_METAFILE_DATA_RECORD
  (TYPE_OF_ITEM : in METAFILE_ITEM_TYPE;
   ITEM_DATA    : in PHIGS_STRING;
   ITEM         : out METAFILE_DATA_RECORD);
```

ITEM DATA RECORD STRING

```
function ITEM_DATA_RECORD_STRING
  (ITEM : in METAFILE_DATA_RECORD) return ACCESS_STRING;
```

5.15 Conformal Variants

Since no subsets or supersets of the Ada language are allowed, PHIGS/Ada has no conformal variants. Furthermore, this binding does not require the use of any Ada language feature for which support of that feature is implementation-dependent.

Annex A

(informative)

Compilable PHIGS Specification

This annex is intended to show one example of a compilable PHIGS binding to Ada. It is not the only possible such example. It is expected that implementations of a PHIGS binding to Ada will need to add additional constructs to the various packages or rearrange the constructs in a manner suitable to the implementation. However, all names defined in this binding shall be visible as dictated in this standard.

-- **PHIGS Configuration Package**

```
package PHIGS_CONFIGURATION is

-- The following define example values for the PHIGS configuration names.
-- Note that the values specified are implementation dependent. The values
-- shown below are arbitrary and have been chosen solely as examples. More
-- appropriate values should be chosen by an implementation. Note also that
-- only the technique of defining the names as constants was used in this
-- example. An implementation is free to choose the most appropriate means
-- of providing the value as discussed in 4.2.5.

   DEFAULT_ERROR_FILE                          : constant STRING := "ERROR.FIL";
   DEFAULT_LIST_SIZE                           : constant := 256;
   DEFAULT_MEMORY_UNITS                        : constant := 0;
   MAX_ANNOTATION_STYLES_SUPPORTED             : constant := 2;
   MAX_APPLICATION_DATA                        : constant := 256;
   MAX_ARCHIVE_IDS_SUPPORTED                   : constant := 4;
   MAX_CHAR_SETS_SUPPORTED                     : constant := 1;
   MAX_CHOICE_PROMPTS_SUPPORTED                : constant := 32;
   MAX_CHOICE_PROMPT_ECHO_TYPES_SUPPORTED      : constant := 10;
   MAX_CHOICE_SMALL_NATURAL                    : constant := 64;
   MAX_COLOUR_COEFFICIENTS                     : constant := 4;
   MAX_COLOUR_INDICES_SUPPORTED                : constant := 4096;
   MAX_COLOUR_MATRIX_SMALL_NATURAL             : constant := 128;
   MAX_COLOUR_MODELS_SUPPORTED                 : constant := 2;
   MAX_EDGE_INDICES_SUPPORTED                  : constant := 5;
   MAX_EDGETYPES_SUPPORTED                     : constant := 1;
   MAX_ESCAPE_IDS_SUPPORTED                    : constant := 16;
   MAX_FILE_IDS_SUPPORTED                      : constant := 10;
   MAX_FILE_SMALL_NATURAL                      : constant := 80;
   MAX_GDP_3_IDS_SUPPORTED                     : constant := 24;
   MAX_GDP_IDS_SUPPORTED                       : constant := 24;
   MAX_GSE_IDS_SUPPORTED                       : constant := 8;
   MAX_HATCH_STYLES_SUPPORTED                  : constant := 64;
   MAX_HLHSR_IDS_SUPPORTED                     : constant := 128;
   MAX_HLHSR_MODES_SUPPORTED                   : constant := 4;
```

```
   MAX_INPUT_STRING_SMALL_NATURAL                    : constant := 80;
   MAX_INTERIOR_INDICES_SUPPORTED                    : constant := 5;
   MAX_LINETYPES_SUPPORTED                           : constant := 5;
   MAX_LOCATOR_PROMPT_ECHO_TYPES_SUPPORTED           : constant := 8;
   MAX_MARKER_TYPES_SUPPORTED                        : constant := 256;
   MAX_MEMORY_UNITS                                  : constant := 0;
   MAX_METAFILE_ITEM_LENGTH                          : constant := 1024;
   MAX_METAFILE_ITEM_TYPE                            : constant := 256;
   MAX_MODELLING_CLIP_OPERATION_TYPES_SUPPORTED      : constant := 2;
   MAX_MODELLING_CLIP_DISTINCT_PLANES_SUPPORTED      : constant := 6;
   MAX_NAME_SUPPORTED                                : constant := 63;
   MAX_NAME_SET_FILTER_LIST_LENGTH_SUPPORTED         : constant := 4;
   MAX_OPEN_WS_SUPPORTED                             : constant := 4;
   MAX_PATH_DEPTH_SUPPORTED                          : constant := 64;
   MAX_PATTERN_INDICES_SUPPORTED                     : constant := 4;
   MAX_PICK_IDS_SUPPORTED                            : constant := 64536;
   MAX_PICK_PROMPT_ECHO_TYPES_SUPPORTED              : constant := 2;
   MAX_POLYLINE_INDICES_SUPPORTED                    : constant := 5;
   MAX_POLYMARKER_INDICES_SUPPORTED                  : constant := 5;
   MAX_POSTED_STRUCTURES_SUPPORTED                   : constant := 4096;
   MAX_REFERENCE_PATHS_SUPPORTED                     : constant := 64;
   MAX_SMALL_NATURAL                                 : constant := 255;
   MAX_STRING_PROMPT_ECHO_TYPES_SUPPORTED            : constant := 2;
   MAX_STRING_SMALL_NATURAL                          : constant := 255;
   MAX_STROKE_PROMPT_ECHO_TYPES_SUPPORTED            : constant := 2;
   MAX_STRUCTURE_IDS_SUPPORTED                       : constant := 65535;
   MAX_TEXT_FONT_PRECISION_PAIRS_SUPPORTED           : constant := 32;
   MAX_TEXT_INDICES_SUPPORTED                        : constant := 5;
   MAX_VALUATOR_PROMPT_ECHO_TYPES_SUPPORTED          : constant := 4;
   MAX_VIEW_INDICES_SUPPORTED                        : constant := 20;
   MAX_WS_IDS_SUPPORTED                              : constant := 16383;
   MAX_WS_TYPES_SUPPORTED                            : constant := 64;
   MIN_METAFILE_ITEM_TYPE                            : constant := 0;
   MIN_INTEGER_VALUE                                 : constant := -2147483648;
   MAX_INTEGER_VALUE                                 : constant := 2147483647;
   PHIGS_PRECISION                                   : constant := 7;

end PHIGS_CONFIGURATION;
```

```
-- PHIGS List Utilities Generic Package

with PHIGS_CONFIGURATION;

generic

   type ELEMENT_TYPE is private;

   MAX_LIST_SIZE : NATURAL := PHIGS_CONFIGURATION.DEFAULT_LIST_SIZE;

package PHIGS_LIST_UTILITIES is

   subtype LIST_SIZE is NATURAL range 0..MAX_LIST_SIZE;

   type LIST_OF (SIZE : LIST_SIZE := 0) is private;

   type LIST_VALUES is ARRAY (POSITIVE range <>) of ELEMENT_TYPE;

   function NULL_LIST return LIST_OF;

      -- This function returns an empty LIST_OF object.  This list is
      -- intended primarily for use by implementors.

   function SIZE_OF_LIST (LIST : in LIST_OF) return NATURAL;

      -- This function returns the number of element type values stored
      -- in the list object.

   function IS_IN_LIST (ELEMENT  : in ELEMENT_TYPE;
                        LIST     : in LIST_OF) return BOOLEAN;

      -- This function returns the value TRUE if the element parameter
      -- value is in the list object.  Otherwise it returns the value
      -- FALSE.

   function LIST_ELEMENT (INDEX : in POSITIVE;
                          LIST  : in  LIST_OF) return ELEMENT_TYPE;

      -- This function returns the element value in the list object that
      -- has an associated index value equal to the index parameter.  The
      -- PHIGS_ERROR 2502 is generated if the index parameter exceeds the
      -- current size of the list parameter object.

   function LIST (VALUES : in LIST_VALUES) return LIST_OF;

      -- This function returns a valid LIST_OF object.  If the VALUES
      -- parameter is a null array, an empty LIST_OF object is returned.
      -- If the values parameter is not null, this function returns a
      -- LIST_OF object containing all the values in the VALUES parameter.
      -- The PHIGS_ERROR 2502 is generated if the number of element values
      -- exceeds the specified maximum size of the LIST_OF object.

   procedure ADD_TO_LIST (ELEMENT  : in ELEMENT_TYPE;
                          LIST     : in out LIST_OF);

      -- This procedure stores the element parameter value in the list
      -- parameter object, and increases the size of the list object
      -- by one.  An index value equal to the incremented list size
```

```
    -- is associated with the stored element value.  The ADD_TO_LIST
    -- procedure will generate PHIGS_ERROR 2502 if it is called when
    -- the list parameter has a size equal to the specified maximum
    -- size.  If desired, the user can ensure duplicate values are not
    -- stored.  This is accomplised by calling ADD_TO_LIST with a
    -- particular element value only if the function IS_IN_LIST returns
    -- false for that element value.

  procedure DELETE_FROM_LIST (ELEMENT  : in ELEMENT_TYPE;
                              LIST     : in out LIST_OF);

    -- If the list parameter object does not contain the element
    -- parameter value, this procedure has no effect.  Otherwise, the
    -- first occurrence of the element value is deleted.  The size of
    -- the list object is decreased by one, and the indices associated
    -- with the remaining element values are adjusted so that the indices
    -- begin at one and increment in steps of one.  If desired, the user
    -- can delete all occurrences of an element value.  This is accomplished
    -- by calling DELETE_FROM_LIST repeatedly with a particular element
    -- value while the function IS_IN_LIST returns TRUE for that value.

private

  -- The declaration of the LIST_OF type is implementation dependent.
  -- However, the operations implicitly declared by the LIST_OF declaration,
  -- including both assignment and the predefined comparison for equality
  -- and inequality, shall produce the correct results.  This requirement
  -- precludes the use of access types for the implementation of the
  -- LIST_OF type.  The recommended implementation is given below:
  --
  type LIST_OF (SIZE : LIST_SIZE := 0) is
    record
      ELEMENTS : LIST_VALUES(1..SIZE);
    end record;
  --
  -- Note that declaring unconstrained LIST_OF objects by using the default
  -- discriminant value allows dynamic modification of the size of the
  -- element array.

end PHIGS_LIST_UTILITIES;

package body PHIGS_LIST_UTILITIES is

  -- This package body contains stubs for each of the subprograms
  -- contained in the PHIGS_LIST_UTILITIES package.  The purpose of
  -- these stubs is to show compilability for the PHIGS/Ada binding.
  -- An implementation would replace these stubs with code appropriate
  -- to implement the functionality.

  function NULL_LIST return LIST_OF is
    EMPTY_LIST : LIST_OF;
  begin
    return EMPTY_LIST;
  end NULL_LIST;

  function SIZE_OF_LIST (LIST : in LIST_OF) return NATURAL is
  begin
    return 0;
  end SIZE_OF_LIST;
```

```
   function IS_IN_LIST (ELEMENT : in ELEMENT_TYPE;
                        LIST    : in LIST_OF) return BOOLEAN is
   begin
      return FALSE;
   end IS_IN_LIST;

   function LIST_ELEMENT (INDEX : in POSITIVE;
                          LIST  : in  LIST_OF) return ELEMENT_TYPE is
   EXAMPLE_ELEMENT : ELEMENT_TYPE;
   begin
     return EXAMPLE_ELEMENT;
   end LIST_ELEMENT;

   function LIST (VALUES : in LIST_VALUES) return LIST_OF is
   begin
      return NULL_LIST;
   end LIST;

   procedure ADD_TO_LIST (ELEMENT : in ELEMENT_TYPE;
                          LIST    : in out LIST_OF) is
   begin
      null;
   end ADD_TO_LIST;

   procedure DELETE_FROM_LIST (ELEMENT : in ELEMENT_TYPE;
                               LIST    : in out LIST_OF) is
   begin
      null;
   end DELETE_FROM_LIST;

end PHIGS_LIST_UTILITIES;
```

```
-- PHIGS Name Set Facility Package

with PHIGS_CONFIGURATION, PHIGS_LIST_UTILITIES;

package PHIGS_NAME_SET_FACILITY is

   -- PHIGS_CONFIGURATION.MAX_NAME_SUPPORTED

   -- An implementation dependent constant which defines the maximum
   -- name supported. The minimum value allowed by the PHIGS
   -- specification is 63 thus supporting 64 classes. A larger
   -- value is strongly encouraged.

   type NAME_SET is private;

   -- Defines a name set data element.

   -- NAME_SET_ACCEPTANCE

   type NAME_SET_ACCEPTANCE is (REJECTED, ACCEPTED);

   -- Used to indicate the results of applying a name set
   -- against a filter pair.

   -- NAME

   type NAME is
            range 0..PHIGS_CONFIGURATION.MAX_NAME_SUPPORTED;

   -- Provides for names for each of the name set classes.

   -- NAMES

   package NAMES is
               new PHIGS_LIST_UTILITIES (NAME);

   -- Provides for lists of names.

   -- NAME_SET_FILTER

   type NAME_SET_FILTER is
     record
       INCLUSION : NAME_SET;
       EXCLUSION : NAME_SET;
     end record;

     -- Used to define name set pairs used as filters.

   -- NAME_SET_MEMBERSHIP

   type NAME_SET_MEMBERSHIP is (NON_MEMBER, MEMBER);

   -- Provides for indicating whether a name is a member of a
   -- name set.

   -- Subprograms for Manipulating Name Sets

   procedure BUILD_NAME_SET
```

```
    (MEMBERS : in   NAMES.LIST_OF;
     SET            : out NAME_SET);

  -- Constructs a name set from a list of names.

  procedure BUILD_NAME_SET_UNION
    (LEFT_SET    : in  NAME_SET;
     RIGHT_SET   : in  NAME_SET;
     UNION       : out NAME_SET);

    -- Constructs the logical union of two name sets.

  procedure BUILD_NAME_SET_INTERSECTION
    (LEFT_SET         : in  NAME_SET;
     RIGHT_SET        : in  NAME_SET;
     INTERSECTION     : out NAME_SET);

    -- Constructs the logical intersection of two name sets.

  procedure BUILD_NAME_SET_DIFFERENCE
    (LEFT_SET        : in  NAME_SET;
     RIGHT_SET       : in  NAME_SET;
     DIFFERENCE      : out NAME_SET);

    -- Constructs the logical difference of two name sets.

  procedure COPY_NAME_SET
    (ORIGINAL  : in  NAME_SET;
     COPY           : out NAME_SET);

    -- Duplicates the provided name set.

  function IS_EQUAL
    (LEFT_SET    : in NAME_SET;
     RIGHT_SET   : in NAME_SET)
    return BOOLEAN;

    -- Compares two name sets.

  procedure EXTRACT_NAMES
    (SET            : in  NAME_SET;
     MEMBERS        : out NAMES.LIST_OF);

  -- Extracts the names from the provided name set.

  procedure ADD_NAMES
    (MEMBERS : in      NAMES.LIST_OF;
     SET             : in out NAME_SET);

  -- Adds names to the name set if not already there.

  procedure REMOVE_NAMES
    (MEMBERS : in      NAMES.LIST_OF;
     SET             : in out NAME_SET);

  -- Removes names from the name set if they are there.

  function APPLY_FILTER
    (SET        : in NAME_SET;
```

```
      FILTER  : in NAME_SET_FILTER)
    return NAME_SET_ACCEPTANCE;

  -- Applies a filter to a name set and indicates whether the name
  -- set passes through the filter.

  function INQUIRE_MEMBERSHIP
    (MEMBER : in NAME;
     SET          : in NAME_SET)
    return NAME_SET_MEMBERSHIP;

  -- Returns an indication of whether the indicated name is a member
  -- of the provided name set.

  procedure DEALLOCATE
    (OBJECT  : in out NAME_SET);

    -- Deallocates name set objects.

  -- PRIVATE TYPE DEFINITIONS

private

  -- The following types define the specifications for private
  -- types used in the PHIGS_NAME_SET_FACILITY.  Null records are
  -- used soley for the purposes of demonstrating compilability.
  -- An implementation would use some more appropriate declaration.

  type NAME_SET is
    record
      null;
    end record;

end PHIGS_NAME_SET_FACILITY;
```

```
-- The PHIGS Types Package

with PHIGS_CONFIGURATION, PHIGS_LIST_UTILITIES, PHIGS_NAME_SET_FACILITY;

package PHIGS_TYPES is

-- This package contains all the data type definitions used to define
-- the Ada binding to PHIGS.

-- Configuration names

use PHIGS_CONFIGURATION;


-- Basic type definitions

-- BASE_TYPES

    package BASE_TYPES is

    -- This package is used to encapsulate the base derived types of PHIGS
    -- since they are used as the parent of several other derived types.
    -- In Ada, if the parent of a derived type is itself a derived type,
    -- then this parent type cannot be declared immediately in the visible
    -- part of the same package.

    -- PHIGS_INTEGER

    type PHIGS_INTEGER is range MIN_INTEGER_VALUE .. MAX_INTEGER_VALUE;

    --  Base type for PHIGS integer types.

    -- PHIGS_NATURAL

    subtype PHIGS_NATURAL is
                PHIGS_INTEGER range 0..PHIGS_INTEGER'last;

    --  Base type for PHIGS natural types.

    -- PHIGS POSITIVE

    subtype PHIGS_POSITIVE is
                PHIGS_INTEGER range 1..PHIGS_INTEGER'last;

    --  Base type for PHIGS positive types.

    -- SCALE_FACTOR

    type SCALE_FACTOR is digits PHIGS_PRECISION;

    -- The type used for unitless scaling factors.

    -- PHIGS_STRING

    type PHIGS_STRING is
            array (PHIGS_INTEGER range <>) of CHARACTER;

    -- Base type for PHIGS string types.
```

```
end BASE_TYPES;

subtype PHIGS_INTEGER is BASE_TYPES.PHIGS_INTEGER;
subtype PHIGS_NATURAL is BASE_TYPES.PHIGS_NATURAL;
subtype PHIGS_POSITIVE is BASE_TYPES.PHIGS_POSITIVE;
subtype PHIGS_STRING is BASE_TYPES.PHIGS_STRING;
subtype SCALE_FACTOR is BASE_TYPES.SCALE_FACTOR;

--  Make BASE_TYPES visible externally.

-- SMALL_NATURAL

subtype SMALL_NATURAL is
        PHIGS_NATURAL range 0..MAX_SMALL_NATURAL;

-- This is a subtype declaration which allows for unconstrained record
-- objects for various record types without causing the exception
-- STORAGE_ERROR to be raised.

-- CHOICE_SMALL_NATURAL

subtype CHOICE_SMALL_NATURAL is
        PHIGS_NATURAL range 0..MAX_CHOICE_SMALL_NATURAL;

-- This is a subtype declaration which allows for
-- unconstrained record objects for CHOICE_PROMPT_STRING_LIST
-- type without causing the exception STORAGE_ERROR to be raised.

-- COLOUR_MATRIX_SMALL_NATURAL

subtype COLOUR_MATRIX_SMALL_NATURAL is
        PHIGS_NATURAL range 0..MAX_COLOUR_MATRIX_SMALL_NATURAL;

-- This is a subtype declaration which allows for unconstrained record
-- objects of VARIABLE_COLOUR_MATRIX type without causing the
-- exception STORAGE_ERROR to be raised.

-- FILE_SMALL_NATURAL

subtype FILE_SMALL_NATURAL is
        PHIGS_NATURAL range 0..MAX_FILE_SMALL_NATURAL;

-- This is a subtype declaration which allows for
-- unconstrained record objects for VARIABLE_FILE_ID type
-- without causing the exception STORAGE_ERROR to be raised.

-- INPUT_STRING_SMALL_NATURAL

subtype INPUT_STRING_SMALL_NATURAL is
        PHIGS_NATURAL range 0..MAX_INPUT_STRING_SMALL_NATURAL;

-- This is a subtype declaration which allows for
-- unconstrained record objects for INPUT_STRING type
-- without causing the exception STORAGE_ERROR to be raised.

-- STRING_SMALL_NATURAL

subtype STRING_SMALL_NATURAL is
```

```
            PHIGS_NATURAL range 0..MAX_STRING_SMALL_NATURAL;

-- This is a subtype declaration which allows for
-- unconstrained record objects for various string record
-- types without causing the exception STORAGE_ERROR to be
-- raised.

generic

  type COORDINATE_COMPONENT_TYPE is digits <>;

  package PHIGS_COORDINATE_SYSTEM is

    type POINT_3 is
      record
        X : COORDINATE_COMPONENT_TYPE;
        Y : COORDINATE_COMPONENT_TYPE;
        Z : COORDINATE_COMPONENT_TYPE;
      end record;

    type POINT_2 is
      record
        X : COORDINATE_COMPONENT_TYPE;
        Y : COORDINATE_COMPONENT_TYPE;
      end record;

    type POINT_LIST_3 is array (POSITIVE range <>) of POINT_3;

    type POINT_LIST_2 is array (POSITIVE range <>) of POINT_2;

    type ACCESS_POINT_LIST_3 is access POINT_LIST_3;

    type ACCESS_POINT_LIST_2 is access POINT_LIST_2;

    type LIST_OF_POINT_LIST_3 is array (POSITIVE range <>) of
                                    ACCESS_POINT_LIST_3;

    type LIST_OF_POINT_LIST_2 is array (POSITIVE range <>) of
                                    ACCESS_POINT_LIST_2;

    type ACCESS_LIST_OF_POINT_LIST_3 is access LIST_OF_POINT_LIST_3;

    type ACCESS_LIST_OF_POINT_LIST_2 is access LIST_OF_POINT_LIST_2;

    type VECTOR_3 is new POINT_3;

    type VECTOR_PAIR_3 is array (1..2) of VECTOR_3;

    type VECTOR_2 is new POINT_2;

    type VECTOR_PAIR_2 is array (1..2) of VECTOR_2;

    type RECTANGULAR_REGION_3 is
      record
        XMIN  : COORDINATE_COMPONENT_TYPE;
        XMAX  : COORDINATE_COMPONENT_TYPE;
        YMIN  : COORDINATE_COMPONENT_TYPE;
        YMAX  : COORDINATE_COMPONENT_TYPE;
        ZMIN  : COORDINATE_COMPONENT_TYPE;
```

```
      ZMAX  : COORDINATE_COMPONENT_TYPE;
    end record;

  type RECTANGULAR_REGION_2 is
    record
      XMIN  : COORDINATE_COMPONENT_TYPE;
      XMAX  : COORDINATE_COMPONENT_TYPE;
      YMIN  : COORDINATE_COMPONENT_TYPE;
      YMAX  : COORDINATE_COMPONENT_TYPE;
    end record;

  type HALF_SPACE_3 is
    record
      REFERENCE_POSITION   : POINT_3;
      ACCEPTANCE_NORMAL  : VECTOR_3;
  end record;

  type HALF_SPACE_2 is
    record
      REFERENCE_POSITION   : POINT_2;
      ACCEPTANCE_NORMAL  : VECTOR_2;
  end record;

  type HALF_SPACE_LIST_3 is
    array (POSITIVE range <>) of HALF_SPACE_3;

  type HALF_SPACE_LIST_2 is
    array (POSITIVE range <>) of HALF_SPACE_2;

  type ACCESS_HALF_SPACE_LIST_3 is access HALF_SPACE_LIST_3;

  type ACCESS_HALF_SPACE_LIST_2 is access HALF_SPACE_LIST_2;

  type MAGNITUDE_BASE_TYPE is digits PHIGS_PRECISION;

  subtype MAGNITUDE is MAGNITUDE_BASE_TYPE range
    COORDINATE_COMPONENT_TYPE'SAFE_SMALL ..
    COORDINATE_COMPONENT_TYPE'SAFE_LARGE;

  type SIZE_3 is
    record
      XAXIS  : MAGNITUDE;
      YAXIS  : MAGNITUDE;
      ZAXIS  : MAGNITUDE;
    end record;

  type SIZE_2 is
    record
      XAXIS  : MAGNITUDE;
      YAXIS  : MAGNITUDE;
    end record;

  type RANGE_OF_MAGNITUDES is
    record
      MIN   : MAGNITUDE;
      MAX  : MAGNITUDE;
    end record;

end PHIGS_COORDINATE_SYSTEM;
```

```
-- MC_TYPE

type MC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of coordinate in the Modeling Coordinate
-- System.

-- MC

package MC is new PHIGS_COORDINATE_SYSTEM (MC_TYPE);

-- Defines the Modeling Coordinate System.

-- WC_TYPE

type WC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of coordinate in the World Coordinate
-- System.

-- WC

package WC is new PHIGS_COORDINATE_SYSTEM (WC_TYPE);

-- Defines the World Coordinate System.

-- VRC_TYPE

type VRC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of coordinate in the View Reference
-- Coordinate System.

-- VRC

package VRC is new PHIGS_COORDINATE_SYSTEM (VRC_TYPE);

-- Defines the viewing coordinate system.

-- NPC_TYPE

type NPC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of a coordinate in the Normalized
-- Projection Coordinate System.

-- NPC

package NPC is new PHIGS_COORDINATE_SYSTEM (NPC_TYPE);

-- Defines the Normalized Projection Coordinate System.

-- DC_TYPE

type DC_TYPE is digits PHIGS_PRECISION;

-- The type of a coordinate component in the Device Coordinate
-- System.
```

```
-- DC

package DC is new PHIGS_COORDINATE_SYSTEM (DC_TYPE);

-- Defines the Device Coordinate System.

-- GENERAL_FLOAT_PARAMETER

type GENERAL_FLOAT_PARAMETER is digits PHIGS_PRECISION;

-- Defines a type for specifying implementation-specific real
-- data in GDP, GSE, and ESCAPE data records.

-- OFF_ON

type OFF_ON is (OFF,
                ON);

-- Generic type for off/on switches.

-- ANGLE

type ANGLE is digits PHIGS_PRECISION;

-- Values used in the modeling transformation utility functions
-- for specifying angles of rotation in radians. Positive
-- indicates a counterclockwise direction.

-- ANNOTATION_STYLE

type ANNOTATION_STYLE is new PHIGS_INTEGER;

-- Defines the annotation styles for annotation primitives.

-- ANNOTATION_STYLES
package ANNOTATION_STYLES is
        new PHIGS_LIST_UTILITIES (ANNOTATION_STYLE,
                                  MAX_ANNOTATION_STYLES_SUPPORTED);

-- Provides for lists of annotation styles.

-- APPLICATION_DATA_RANGE

subtype APPLICATION_DATA_RANGE is
        PHIGS_NATURAL range 0..MAX_APPLICATION_DATA;

-- Defines range of lengths for application data objects.

-- APPLICATION DATA RECORD

type APPLICATION_DATA_RECORD (LENGTH : APPLICATION_DATA_RANGE := 0) is
  record
    DATA : PHIGS_STRING (1..LENGTH);
  end record;

-- Defines type for defining application data objects.

-- ARCHIVE_ID
```

```
type ARCHIVE_ID is new PHIGS_NATURAL;

-- Provides for an application specified pointer to archive
-- files separate from the file identifier.

-- ARCHIVE_IDS

package ARCHIVE_IDS is new PHIGS_LIST_UTILITIES (ARCHIVE_ID,
                                                 MAX_ARCHIVE_IDS_SUPPORTED);

-- Provides for lists of archive identifiers.

-- ARCHIVE_STATE

type ARCHIVE_STATE is (ARCL,
                       AROP);

--  The type used to return the archive state.

-- ASF

type ASF is (BUNDLED,
             INDIVIDUAL);

-- This type defines an aspect source flag whose
-- value indicates whether an aspect of a primitive
-- should be set from a bundle table or from an
-- individual attribute.

-- ASPECT

type ASPECT is (TYPE_OF_LINE,
                LINEWIDTH_SF,
                LINE_COLOUR,
                TYPE_OF_MARKER,
                SIZE,
                MARKER_COLOUR,
                FONT,
                PRECISION,
                EXPANSION,
                SPACING,
                TEXT_COLOUR,
                STYLE_OF_INTERIOR,
                STYLE_IND,
                INTERIOR_COLOUR,
                FLAG,
                TYPE_OF_EDGE,
                EDGEWIDTH_SF,
                EDGE_COLOUR);

-- This type lists the aspects for which an aspect source
-- flag exists.

-- ATTRIBUTES_USED_TYPE

type ATTRIBUTES_USED_TYPE is (POLYLINE_ATTRIBUTES,
                              POLYMARKER_ATTRIBUTES,
                              TEXT_ATTRIBUTES,
                              INTERIOR_ATTRIBUTES,
```

```
                                        EDGE_ATTRIBUTES);

-- The types of attributes which may be used in generating
-- output for a GDP and in generating prompt and echo
-- information for certain prompt and echo types of certain
-- classes of input devices.

-- ATTRIBUTES_USED

package ATTRIBUTES_USED is
   new PHIGS_LIST_UTILITIES
         (ATTRIBUTES_USED_TYPE,
          ATTRIBUTES_USED_TYPE'POS(ATTRIBUTES_USED_TYPE'LAST)+1);

-- Provides for a list of the attributes used which is returned
-- by the INQ_GDP and LOCATOR_ATTRIBUTES_USED functions.

-- CHAR_EXPANSION

type CHAR_EXPANSION is new SCALE_FACTOR range
  SCALE_FACTOR'SAFE_SMALL..SCALE_FACTOR'LAST;

-- Defines a character expansion factor.

-- CHAR_SET

type CHAR_SET is new PHIGS_NATURAL;

-- Provides identifications for the available character
-- sets.  ISO 646 character set maps to value zero.

-- CHAR_SETS

package CHAR_SETS is new PHIGS_LIST_UTILITIES (CHAR_SET,
                                               MAX_CHAR_SETS_SUPPORTED);

-- Provides for lists of character set identifications.

-- CHAR_SPACING

type CHAR_SPACING is new SCALE_FACTOR;

-- Defines a character spacing factor.  The factors are
-- unitless.  A positive value indicates the amount of extra
-- space between character boxes in a text string, and a
-- negative value indicates the amount of overlap between
-- character boxes in a text string.

-- CHOICE_PROMPT_ECHO_TYPE

type CHOICE_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the choice prompt and echo type.

-- CHOICE_PROMPT_ECHO_TYPES

package CHOICE_PROMPT_ECHO_TYPES is
   new PHIGS_LIST_UTILITIES (CHOICE_PROMPT_ECHO_TYPE,
                             MAX_CHOICE_PROMPT_ECHO_TYPES_SUPPORTED);
```

```
-- Provides for lists of choice prompt and echo types.

-- DEVICE_NUMBER

subtype DEVICE_NUMBER is PHIGS_POSITIVE;

--  Defines the base type for input device numbers.

-- CHOICE_DEVICE_NUMBER

type CHOICE_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for choice device identifiers.

-- CHOICE_NUMBER

type CHOICE_NUMBER is new PHIGS_POSITIVE;

-- Defines the choice numbers available on an implementation.

-- CHOICE_PROMPT

type CHOICE_PROMPT is new OFF_ON;

-- Indicates whether a specified choice prompt is to be displayed
-- or not.

-- CHOICE_PROMPTS

package CHOICE_PROMPTS is
            new PHIGS_LIST_UTILITIES (CHOICE_PROMPT,
                                      MAX_CHOICE_PROMPTS_SUPPORTED);

-- Provides for lists of choice prompts.

-- CHOICE_PROMPT_STRING

type CHOICE_PROMPT_STRING (LENGTH : STRING_SMALL_NATURAL := 0) is
  record
    CONTENTS : PHIGS_STRING (1..LENGTH);
  end record;

-- Provides for a variable length prompt.  Objects of this type
-- should be declared unconstrained to allow for dynamic
-- modification of the length.

-- CHOICE_PROMPT_STRING_ARRAY

type CHOICE_PROMPT_STRING_ARRAY is array (PHIGS_POSITIVE range <>)
                                    of CHOICE_PROMPT_STRING;

-- Provides for an array of prompt strings.

-- CHOICE_PROMPT_STRING_LIST

type CHOICE_PROMPT_STRING_LIST
  (LENGTH : CHOICE_SMALL_NATURAL := 0) is
      record
        LIST : CHOICE_PROMPT_STRING_ARRAY (1..LENGTH);
```

```
        end record;

-- Provides for lists of prompt strings.

-- CHOICE_REQUEST_STATUS

type CHOICE_REQUEST_STATUS is (OK,
                               NOCHOICE,
                               NONE);

-- Defines the status of a choice input operation for the
-- request function.

-- CHOICE_STATUS

subtype CHOICE_STATUS is CHOICE_REQUEST_STATUS range OK..NOCHOICE;

-- Indicates if a choice was made by the operator for the
-- sample, get, and inquiry functions.

-- CLIPPING_INDICATOR

type CLIPPING_INDICATOR is (CLIP,
                            NOCLIP);

-- Indicates whether or not clipping is to be performed.

-- COLOUR_AVAILABLE

type COLOUR_AVAILABLE is (COLOUR,
                          MONOCHROME);

-- Indicates whether colour output is available on a
-- workstation.

-- COLOUR_COMPONENTS

type COLOUR_COMPONENTS is range 1..MAX_COLOUR_COEFFICIENTS;

-- Defines a type for the size of colour component arrays.

-- COLOUR_INDEX

type COLOUR_INDEX is new PHIGS_NATURAL;

-- Indices into colour tables are of this type.

-- COLOUR_INDICES

package COLOUR_INDICES is
        new PHIGS_LIST_UTILITIES (COLOUR_INDEX,
                                  MAX_COLOUR_INDICES_SUPPORTED);

-- Provides for a set of colour indices which are available
-- on a particular workstation.

-- COLOUR_MATRIX

type COLOUR_MATRIX is array (COLOUR_MATRIX_SMALL_NATURAL range <>,
```

```
                                    COLOUR_MATRIX_SMALL_NATURAL range <>)
                                of COLOUR_INDEX;

-- Provides for matrices containing colour indices corresponding
-- to a cell array or pattern array.

-- COLOUR_MODEL

type COLOUR_MODEL is new PHIGS_INTEGER;

-- Indicates the colour models available in PHIGS.

-- COLOUR_MODELS

package COLOUR_MODELS is
        new PHIGS_LIST_UTILITIES (COLOUR_MODEL,
                                  MAX_COLOUR_MODELS_SUPPORTED);

-- Provides for lists of colour models.

-- The following constants define the colour models specified by PHIGS:
RGB     : constant COLOUR_MODEL := 1;
CIELUV  : constant COLOUR_MODEL := 2;
HSV     : constant COLOUR_MODEL := 3;
HLS     : constant COLOUR_MODEL := 4;

-- COLOUR_COEFFICIENT

type COLOUR_COEFFICIENT is digits PHIGS_PRECISION;

-- Defines a general type for colour components.

-- COLOUR_COEFFICIENT_ARRAY

type COLOUR_COEFFICIENT_ARRAY is
  array (COLOUR_COMPONENTS) of COLOUR_COEFFICIENT;

-- Defines an array type for colour components.

-- COLOUR_COEFFICIENTS

package COLOUR_COEFFICIENTS is
   new PHIGS_LIST_UTILITIES (COLOUR_COEFFICIENT,
                             MAX_COLOUR_COEFFICIENTS);

-- Provides for a lists of colour coefficients for the generalized
-- colour model.

-- INTENSITY

subtype INTENSITY is COLOUR_COEFFICIENT range 0.0..1.0;

-- Defines the restricted range of colour components required
-- by some colour models.

-- COLOUR_REPRESENTATION

type COLOUR_REPRESENTATION (MODEL : COLOUR_MODEL := RGB) is
  record
```

```
    case MODEL is

      when RGB = >
         RED_RGB                  : INTENSITY;
         GREEN_RGB                : INTENSITY;
         BLUE_RGB                 : INTENSITY;

      when CIELUV = >
         L_STAR_CIE               : COLOUR_COEFFICIENT;
         U_STAR_CIE               : COLOUR_COEFFICIENT;
         V_STAR_CIE               : COLOUR_COEFFICIENT;

      when HSV = >
         HUE_HSV                  : INTENSITY;
         SATURATION_HSV           : INTENSITY;
         VALUE_HSV                : INTENSITY;

      when HLS = >
         HUE_HLS                  : INTENSITY;
         LIGHTNESS_HLS            : INTENSITY;
         SATURATION_HLS           : INTENSITY;

      when others = >
         GENERIC_COMPONENTS  : COLOUR_COEFFICIENT_ARRAY;
    end case;
  end record;

-- Defines the representation of a colour as a combination of three
-- components whose meaning depends on the colour model.  Additional
-- variants may be included when additional registered or unregistered
-- colour models are supported by an implementation.

-- CHROMATICITY_COEFFICIENT

type CHROMATICITY_COEFFICIENT is
  record
    X : COLOUR_COEFFICIENT;
    Y : COLOUR_COEFFICIENT;
  end record;

-- Defines a CIE colour coefficient pair.

-- CHROMATICITY_COEFFICIENT_SET

type CHROMATICITY_COEFFICIENT_SET is
  record
    R : CHROMATICITY_COEFFICIENT;
    G : CHROMATICITY_COEFFICIENT;
    B : CHROMATICITY_COEFFICIENT;
  end record;

-- Defines a set of CIE primary colour chromaticity coefficients.

-- COMPOSITION_TYPE

type COMPOSITION_TYPE is (PRECONCATENATE,
                          POSTCONCATENATE,
                          REPLACE);
```

```
-- Defines the type of matrix composition allowed between
-- modeling transformations.

-- CONFLICT_RESOLUTION

type CONFLICT_RESOLUTION is (MAINTAIN,
                             ABANDON,
                             UPDATE);

-- Defines a type for specifying the conflict resolution mode.

-- CONNECTION_ID

type CONNECTION_ID is new PHIGS_STRING;

--  Provides for specifying connection identifiers.

-- CONTROL_FLAG

type CONTROL_FLAG is (CONDITIONALLY,
                      ALWAYS);

-- The control flag is used to indicate the conditions under
-- which the display surface should be cleared.

-- DC_UNITS

type DC_UNITS is (METRES,
                  OTHER);

-- Device coordinate units for a particular workstation
-- may be in metres, or some other unit (such as inches).

-- DEFERRAL_MODE

type DEFERRAL_MODE is (ASAP,
                       BNIG,
                       BNIL,
                       ASTI,
                       WAIT);

-- Defines the five PHIGS deferral modes.

-- DISPLAY_CLASS

type DISPLAY_CLASS is (VECTOR_DISPLAY,
                       RASTER_DISPLAY,
                       OTHER_DISPLAY);

-- The classification of a workstation of category OUTPUT
-- or OUTIN.

-- DISPLAY_SURFACE_EMPTY

type DISPLAY_SURFACE_EMPTY is (EMPTY,
                               NOTEMPTY);

-- Indicates whether there is graphical output on the display
-- surface.
```

```
-- DYNAMIC_MODIFICATION

type DYNAMIC_MODIFICATION is (IRG,
                              IMM,
                              CBS);

-- Indicates whether an update to the state list is performed
-- immediately (IMM), requires implicit regeneration (IRG), or
-- can be simulated (CBS).

-- ECHO_SWITCH

type ECHO_SWITCH is (NOECHO,
                     ECHO);

-- Indicates whether or not echoing of the measure is
-- performed.

-- EDIT_MODE

type EDIT_MODE is (INSERT,
                   REPLACE);

-- Defines a type for specifying the mode in which editing
-- takes place.

-- ELEMENT_POSITION

type ELEMENT_POSITION is new PHIGS_INTEGER;

-- Base type for element pointer values and offsets.

-- RETURNED_ELEMENT_POSITION

subtype RETURNED_ELEMENT_POSITION is
  ELEMENT_POSITION range 0..ELEMENT_POSITION'LAST;

-- Defines a type for returning element pointer values.

-- ELEMENT_TYPE

type ELEMENT_TYPE is (ALL_ELEMENT_TYPES,
                      NIL,

                      POLYLINE_3,
                      POLYLINE,
                      POLYMARKER_3,
                      POLYMARKER,
                      TEXT_3,
                      TEXT,
                      ANNOTATION_TEXT_RELATIVE_3,
                      ANNOTATION_TEXT_RELATIVE,
                      FILL_AREA_3,
                      FILL_AREA,
                      FILL_AREA_SET_3,
                      FILL_AREA_SET,
                      CELL_ARRAY_3,
                      CELL_ARRAY,
```

```
    GDP_3,
    GDP,

    SET_POLYLINE_INDEX,
    SET_POLYMARKER_INDEX,
    SET_TEXT_INDEX,
    SET_INTERIOR_INDEX,
    SET_EDGE_INDEX,

    SET_LINETYPE,
    SET_LINEWIDTH_SCALE_FACTOR,
    SET_POLYLINE_COLOUR_INDEX,

    SET_MARKER_TYPE,
    SET_MARKER_SIZE_SCALE_FACTOR,
    SET_POLYMARKER_COLOUR_INDEX,

    SET_TEXT_FONT,
    SET_TEXT_PRECISION,
    SET_CHAR_EXPANSION_FACTOR,
    SET_CHAR_SPACING,
    SET_TEXT_COLOUR_INDEX,
    SET_CHAR_HEIGHT,
    SET_CHAR_UP_VECTOR,
    SET_TEXT_PATH,
    SET_TEXT_ALIGNMENT,

    SET_ANNOTATION_TEXT_CHAR_HEIGHT,
    SET_ANNOTATION_TEXT_CHAR_UP_VECTOR,
    SET_ANNOTATION_TEXT_PATH,
    SET_ANNOTATION_TEXT_ALIGNMENT,
    SET_ANNOTATION_STYLE,

    SET_INTERIOR_STYLE,
    SET_INTERIOR_STYLE_INDEX,
    SET_INTERIOR_COLOUR_INDEX,

    SET_EDGE_FLAG,
    SET_EDGETYPE,
    SET_EDGEWIDTH_SCALE_FACTOR,
    SET_EDGE_COLOUR_INDEX,

    SET_PATTERN_SIZE,
    SET_PATTERN_REFERENCE_POINT_AND_VECTORS,
    SET_PATTERN_REFERENCE_POINT,

    ADD_NAMES_TO_SET,
    REMOVE_NAMES_FROM_SET,

    SET_INDIVIDUAL_ASF,
    SET_HLHSR_IDENTIFIER,

    SET_LOCAL_TRANSFORMATION_3,
    SET_LOCAL_TRANSFORMATION,
    SET_GLOBAL_TRANSFORMATION_3,
    SET_GLOBAL_TRANSFORMATION,
    SET_MODELLING_CLIPPING_VOLUME_3,
    SET_MODELLING_CLIPPING_VOLUME,
    SET_MODELLING_CLIPPING_INDICATOR,
```

```
                                RESTORE_MODELLING_CLIPPING_VOLUME,
                                SET_VIEW_INDEX,

                                EXECUTE_STRUCTURE,

                                LABEL,
                                APPLICATION_DATA,
                                GSE,
                                SET_PICK_IDENTIFIER);

-- Provides for specification of element types.

-- ELEMENT_TYPES

package ELEMENT_TYPES is
        new PHIGS_LIST_UTILITIES (ELEMENT_TYPE,
                                  ELEMENT_TYPE'POS(ELEMENT_TYPE'LAST)+1);

--  Provides for lists of element types.

-- ERROR_HANDLING_MODE

type ERROR_HANDLING_MODE is new OFF_ON;

-- Type for inquiring the error handling mode.

-- ERROR_NUMBER

type ERROR_NUMBER is new PHIGS_INTEGER;

-- Defines the type for error indicator values.

-- ESCAPE_ID

type ESCAPE_ID is new PHIGS_INTEGER;

-- Defines a type for identifying different escapes.

-- ESCAPE_IDS

package ESCAPE_IDS is
        new PHIGS_LIST_UTILITIES (ESCAPE_ID,
                                  MAX_ESCAPE_IDS_SUPPORTED);

-- Provides for lists of Escape ID's.

-- ESCAPE_DATA_RECORD

type ESCAPE_DATA_RECORD (ESCAPE : ESCAPE_ID := 0) is
  record
    case ESCAPE is

      -- For each ESCAPE which needs a record, an implementation
      -- dependent entry is defined.  For others, a null record
      -- is defined.

      when others = >
         null;
    end case;
```

```
  end record;

-- Provides for the definition of ESCAPE data records.

-- FILE_ID

subtype FILE_ID is PHIGS_STRING;

-- Provides for file identifiers.

-- VARIABLE_FILE_ID

type VARIABLE_FILE_ID (LENGTH : FILE_SMALL_NATURAL := 0) is
  record
    NAME : FILE_ID (1..LENGTH);
  end record;

-- Provides for variable length file identifiers.

-- VARIABLE_FILE_IDS

package VARIABLE_FILE_IDS is
   new PHIGS_LIST_UTILITIES (VARIABLE_FILE_ID,
                             MAX_FILE_IDS_SUPPORTED);

-- This type is used to define lists of file identifiers.

-- GDP_3_ID

type GDP_3_ID is new PHIGS_INTEGER;

-- Defines a type for selecting a Generalized Drawing Primitive.

-- GDP_3_IDS

package GDP_3_IDS is
          new PHIGS_LIST_UTILITIES (GDP_3_ID,
                                    MAX_GDP_3_IDS_SUPPORTED);

-- Provides for lists of 3D Generalized Drawing Primitive ID's.

-- GDP_3_RECORD

type GDP_3_RECORD (GDP : GDP_3_ID := 0) is
  record
    case GDP is

      -- For each supported GDP_3 which needs a record, an implementation
      -- dependent entry is defined.  For others, a generic record
      -- is defined.

      when others =>
         GDP_INFO : APPLICATION_DATA_RECORD;
    end case;
  end record;

-- Provides for the definition of GDP_3 data records.

-- GDP_ID
```

```
type GDP_ID is new PHIGS_INTEGER;

-- Defines a type for selecting a Generalized Drawing Primitive.

-- GDP_IDS

package GDP_IDS is
            new PHIGS_LIST_UTILITIES (GDP_ID,
                                      MAX_GDP_IDS_SUPPORTED);

-- Provides for lists of Generalized Drawing Primitive ID's.

-- GDP_RECORD

type GDP_RECORD (GDP : GDP_ID := 0) is
  record
    case GDP is

      -- For each supported GDP which needs a record, an implementation
      -- dependent entry is defined.  For others, a generic record
      -- is defined.

      when others =>
         GDP_INFO : APPLICATION_DATA_RECORD;
    end case;
  end record;

-- Provides for the definition of GDP data records.

-- GSE_ID

type GSE_ID is new PHIGS_INTEGER;

--  Defines a type for selecting a Generalized Structure Element.

-- GSE_IDS

package GSE_IDS is new PHIGS_LIST_UTILITIES (GSE_ID,
                                             MAX_GSE_IDS_SUPPORTED);

--  Provides for lists of Generalized Structure Element ID's.

-- GSE_RECORD

type GSE_RECORD (GSE : GSE_ID := 0) is
  record
    case GSE is

      -- For each supported GSE which needs a record, an implementation
      -- dependent entry is defined.  For others, a generic record
      -- is defined.

      when others =>
         GSE_INFO : APPLICATION_DATA_RECORD;
    end case;
  end record;

--  Provides for the definition of GSE data records.
```

```
-- STYLE_INDEX

type STYLE_INDEX is new PHIGS_INTEGER;

-- Defines an interior style index.

-- HATCH_STYLE

subtype HATCH_STYLE is STYLE_INDEX;

-- Defines the interior hatch styles type.

-- HATCH_STYLES

package HATCH_STYLES is
   new PHIGS_LIST_UTILITIES (HATCH_STYLE,
                             MAX_HATCH_STYLES_SUPPORTED);


-- Provides for lists of hatch styles.

-- HLHSR_ID

type HLHSR_ID is new PHIGS_INTEGER;

--  A number used as an attribute for workstation independent
--  HLHSR factors.

-- HLHSR_IDS

package HLHSR_IDS is new PHIGS_LIST_UTILITIES (HLHSR_ID,
   MAX_HLHSR_IDS_SUPPORTED);

--  Provides for lists of HLHSR identifiers.

-- HLHSR_MODE

type HLHSR_MODE is new PHIGS_INTEGER;

--  A number used for controlling workstation dependent
--  HLHSR factors.

-- HLHSR_MODES

package HLHSR_MODES is new PHIGS_LIST_UTILITIES (HLHSR_MODE,
                                                 MAX_HLHSR_MODES_SUPPORTED);

--  Provides for lists of HLHSR modes.

-- HORIZONTAL_ALIGNMENT

type HORIZONTAL_ALIGNMENT is (NORMAL,
                              LEFT,
                              CENTRE,
                              RIGHT);

-- The alignment of the text extent rectangle with respect to
-- horizontal positioning of the text.
```

```
-- INPUT_CLASS

type INPUT_CLASS is (NONE,
                     LOCATOR_INPUT,
                     STROKE_INPUT,
                     VALUATOR_INPUT,
                     CHOICE_INPUT,
                     PICK_INPUT,
                     STRING_INPUT);

-- Defines the input device classifications for workstations
-- of category INPUT or OUTIN.

-- INPUT_QUEUE_CLASS

subtype INPUT_QUEUE_CLASS is
  INPUT_CLASS range LOCATOR_INPUT .. STRING_INPUT;

-- Defines the input device classifications for workstations
-- of category INPUT and OUTIN that do not return a NONE
-- classification.

-- INPUT_STATUS

type INPUT_STATUS is (OK,
                      NONE);

-- Defines the status of a locator, stroke, valuator, or
-- string operation.

-- INPUT_STRING

type INPUT_STRING (LENGTH : INPUT_STRING_SMALL_NATURAL := 0) is
  record
    CONTENTS : PHIGS_STRING (1..LENGTH);
  end record;

-- Provides a variable length input string.  Objects of this type
-- should be declared unconstrained to allow for dynamic modification
-- of the length.

-- INTERIOR_INDEX

type INTERIOR_INDEX is new PHIGS_POSITIVE;

-- Defines interior bundle table indices.

-- INTERIOR_INDICES

package INTERIOR_INDICES is
  new PHIGS_LIST_UTILITIES (INTERIOR_INDEX,
                            MAX_INTERIOR_INDICES_SUPPORTED);

-- Provides for lists of interior bundle table indices.

-- INTERIOR_STYLE

type INTERIOR_STYLE is (HOLLOW,
```

```
                                SOLID,
                                PATTERN,
                                HATCH,
                                EMPTY);

-- Defines the interior styles for filled areas.

-- INTERIOR_STYLES

package INTERIOR_STYLES is
   new PHIGS_LIST_UTILITIES (INTERIOR_STYLE,
                             INTERIOR_STYLE'POS(INTERIOR_STYLE'LAST)+1);

-- Provides for lists of interior styles.

-- INTERIOR_DATA

type INTERIOR_DATA is
  record
    STYLE_OF_INTERIOR_ASF : ASF;
    STYLE_IND_ASF         : ASF;
    INTERIOR_COLOUR_ASF   : ASF;
    INTERIOR_IND          : INTERIOR_INDEX;
    STYLE_OF_INTERIOR     : INTERIOR_STYLE;
    STYLE_IND             : STYLE_INDEX;
    INTERIOR_COLOUR       : COLOUR_INDEX;
  end record;

-- A record containing information needed to specify the
-- appearance of a filled area interior.

-- INVALID_VALUES_INDICATOR

type INVALID_VALUES_INDICATOR is (ABSENT,
                                  PRESENT);

-- Indicates whether invalid values are contained in a colour
-- array or matrix.

-- LABEL_ID

type LABEL_ID is new PHIGS_INTEGER;

-- Defines a type for specifying structure element labels.

-- POLYLINE_INDEX

type POLYLINE_INDEX is new PHIGS_POSITIVE;

-- Defines the range of polyline indices.

-- POLYLINE_INDICES

package POLYLINE_INDICES is
   new PHIGS_LIST_UTILITIES (POLYLINE_INDEX,
                             MAX_POLYLINE_INDICES_SUPPORTED);

-- Provides for lists of polyline indices.
```

```
-- LINETYPE

type LINETYPE is new PHIGS_INTEGER;

-- Defines the types of line styles provided by PHIGS.

-- LINETYPES

package LINETYPES is new PHIGS_LIST_UTILITIES (LINETYPE,
                                               MAX_LINETYPES_SUPPORTED);

-- Provides for lists of linetypes.

-- LINEWIDTH

type LINEWIDTH is new SCALE_FACTOR range 0.0..SCALE_FACTOR'LAST;

-- The width of a line is indicated by a scale factor.

-- LINE_DATA

type LINE_DATA is
  record
    TYPE_OF_LINE_ASF  : ASF;
    WIDTH_ASF         : ASF;
    LINE_COLOUR_ASF   : ASF;
    POLYLINE_IND      : POLYLINE_INDEX;
    TYPE_OF_LINE      : LINETYPE;
    WIDTH             : LINEWIDTH;
    LINE_COLOUR       : COLOUR_INDEX;
  end record;

-- A record containing information needed to specify the
-- appearance of a line for input data records.

-- EDGE_FLAG

type EDGE_FLAG is new OFF, ON;

-- Defines a type for indicating whether or not to depict edges
-- on fill area set and related primitives.

-- EDGE_INDEX

type EDGE_INDEX is new PHIGS_POSITIVE;

-- Defines the edge bundle table indices.

-- EDGE_INDICES

package EDGE_INDICES is new PHIGS_LIST_UTILITIES (EDGE_INDEX,
                                                  MAX_EDGE_INDICES_SUPPORTED);

-- Provides for a list of edge indices.

-- EDGETYPE

type EDGETYPE is new PHIGS_INTEGER;
```

```
-- Defines a type for describing the form of edges.

-- EDGETYPES

package EDGETYPES is new PHIGS_LIST_UTILITIES (EDGETYPE,
                                               MAX_EDGETYPES_SUPPORTED);

-- Provides for lists of edgetypes.

-- EDGEWIDTH

type EDGEWIDTH is new SCALE_FACTOR range 0.0..SCALE_FACTOR'LAST;

-- Defines a type for describing the width scale factor of edges.

-- EDGE_DATA

type EDGE_DATA is
  record
    FLAG_ASF              : ASF;
    TYPE_OF_EDGE_ASF      : ASF;
    EDGEWIDTH_SF_ASF      : ASF;
    EDGE_COLOUR_ASF       : ASF;
    FLAG                  : EDGE_FLAG;
    TYPE_OF_EDGE          : EDGETYPE;
    EDGEWIDTH_SF          : EDGEWIDTH;
    EDGE_COLOUR           : COLOUR_INDEX;
  end record;

-- A record containing information needed to specify the
-- appearance of a fill area set edge.

-- LOCATOR_DEVICE_NUMBER

type LOCATOR_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for locator device identifiers.

-- LOCATOR_PROMPT_ECHO_TYPE

type LOCATOR_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the locator prompt and echo types supported by
-- the implementation.

-- LOCATOR_PROMPT_ECHO_TYPES

package LOCATOR_PROMPT_ECHO_TYPES is
  new PHIGS_LIST_UTILITIES (LOCATOR_PROMPT_ECHO_TYPE,
                            MAX_LOCATOR_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of locator prompt and echo types.

-- POLYMARKER_INDEX

type POLYMARKER_INDEX is new PHIGS_POSITIVE;

-- Defines the range of polymarker bundle table indices.
```

```
-- POLYMARKER_INDICES

package POLYMARKER_INDICES is
   new PHIGS_LIST_UTILITIES (POLYMARKER_INDEX,
                             MAX_POLYMARKER_INDICES_SUPPORTED);

-- Provides for lists of polymarker indices.

-- MARKER_SIZE

type MARKER_SIZE is new SCALE_FACTOR;

-- The size of a marker is indicated by a scale factor.

-- MARKER_TYPE

type MARKER_TYPE is new PHIGS_INTEGER;

-- Defines the type for markers provided by PHIGS.

-- MARKER_TYPES

package MARKER_TYPES is
          new PHIGS_LIST_UTILITIES (MARKER_TYPE,
                                    MAX_MARKER_TYPES_SUPPORTED);


-- Provides for lists of marker types.

-- MARKER_DATA

type MARKER_DATA is
  record
    TYPE_OF_MARKER_ASF  : ASF;
    SIZE_ASF            : ASF;
    MARKER_COLOUR_ASF   : ASF;
    POLYMARKER_IND      : POLYMARKER_INDEX;
    TYPE_OF_MARKER      : MARKER_TYPE;
    SIZE                : MARKER_SIZE;
    MARKER_COLOUR       : COLOUR_INDEX;
  end record;

-- A record containing information needed to specify the
-- appearance of a marker.

-- MEMORY_UNITS

type MEMORY_UNITS is range 0..MAX_MEMORY_UNITS;

-- Defines the type for units of memory that may be allocated
-- by PHIGS.

-- METAFILE_ITEM_TYPE

type METAFILE_ITEM_TYPE is
        new PHIGS_NATURAL
              range MIN_METAFILE_ITEM_TYPE..MAX_METAFILE_ITEM_TYPE;

--  The type of an item contained in a metafile.
```

```
-- METAFILE_ITEM_LENGTH

type METAFILE_ITEM_LENGTH is
        new PHIGS_NATURAL range 0..MAX_METAFILE_ITEM_LENGTH;

--  The length of an item contained in a metafile.

-- MODELLING_CLIP_OPERATION_TYPE

type MODELLING_CLIP_OPERATION_TYPE is new PHIGS_INTEGER;

--  This type provides for specifying modelling clip operation
--  types.

-- MODELLING_CLIP_OPERATION_TYPES

package MODELLING_CLIP_OPERATION_TYPES is
   new PHIGS_LIST_UTILITIES
         (MODELLING_CLIP_OPERATION_TYPE,
          MAX_MODELLING_CLIP_OPERATION_TYPES_SUPPORTED);

--  This type provides for lists of modelling clip operation types.

-- MODIFICATION_MODE

type MODIFICATION_MODE is (NIVE,
                           UWOR,
                           UQUM);

-- Defines type for specifying modification modes.

-- MORE_EVENTS

type MORE_EVENTS is (NOMORE,
                     MORE);

-- Indicates whether more simultaneous events are contained in
-- the input queue.

-- NAME_SET

subtype NAME_SET is PHIGS_NAME_SET_FACILITY.NAME_SET;

--  Used to define name sets.  The internal structure of
--  a name set is implementation-dependent.  "Build" functions
--  contained in the package PHIGS_NAME_SET_FACILITIES shall be
--  used to create and manipulate name sets (See 5.14.4).

-- NAME_SET_ACCEPTANCE

subtype NAME_SET_ACCEPTANCE is PHIGS_NAME_SET_FACILITY.NAME_SET_ACCEPTANCE;

--  Used to indicate the results of applying a name set
--  against a filter pair.  See 5.14.4.

-- NAME

subtype NAME is PHIGS_NAME_SET_FACILITY.NAME;
```

```
-- Provides for names for each of the name set classes.
-- See 5.14.4.

-- NAMES

package NAMES renames
   PHIGS_NAME_SET_FACILITY.NAMES;

-- Provides for lists of name set classes. See 5.14.4.

-- NAME_SET_FILTER

subtype NAME_SET_FILTER is PHIGS_NAME_SET_FACILITY.NAME_SET_FILTER;

-- Used to define name set pairs used as filters.
-- See 5.14.4.

-- NAME_SET_FILTERS

package NAME_SET_FILTERS is
            new PHIGS_LIST_UTILITIES (NAME_SET_FILTER,
                                      MAX_NAME_SET_FILTER_LIST_LENGTH_SUPPORTED);

-- Used to provide lists of name set filters.
-- See 5.14.4.

-- NAME_SET_MEMBERSHIP

subtype NAME_SET_MEMBERSHIP is
   PHIGS_NAME_SET_FACILITY.NAME_SET_MEMBERSHIP;

-- Provides for indicating whether a class is contained in a
-- name set. See 5.14.4.

-- OPEN_STRUCTURE_STATUS

type OPEN_STRUCTURE_STATUS is (NONE,
                               OPEN);

-- Returns the status of the open structure.

-- OPERATING_MODE

type OPERATING_MODE is (REQUEST_MODE,
                        SAMPLE_MODE,
                        EVENT_MODE);

-- Defines the operating modes of an input device.

-- PATH_DEPTH

subtype PATH_DEPTH is
            PHIGS_NATURAL range 0..MAX_PATH_DEPTH_SUPPORTED;


-- Defines a type for specifying the depth of a traversal path.

-- PATH_ORDER
```

```
type PATH_ORDER is (TOPFIRST,
                    BOTTOMFIRST);

-- Provides for specifying the order of the pick and
-- reference paths.

-- PATTERN_INDEX

subtype PATTERN_INDEX is STYLE_INDEX range 1..STYLE_INDEX'LAST;

-- Defines the range of pattern table indices.

-- PATTERN_INDICES

package PATTERN_INDICES is
   new PHIGS_LIST_UTILITIES (PATTERN_INDEX,
                             MAX_PATTERN_INDICES_SUPPORTED);

-- Provides for lists of pattern table indices.

-- PICK_DEVICE_NUMBER

type PICK_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for pick device identifiers.

-- PICK_ID

type PICK_ID is new PHIGS_INTEGER;

-- Defines the range of pick identifiers available on an
-- implementation.

-- PICK_IDS

package PICK_IDS is
        new PHIGS_LIST_UTILITIES (PICK_ID,
                                  MAX_PICK_IDS_SUPPORTED);

-- Provides for a list of pick identifiers.

-- PICK_PROMPT_ECHO_TYPE

type PICK_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the pick prompt and echo type.

-- PICK_PROMPT_ECHO_TYPES

package PICK_PROMPT_ECHO_TYPES is
   new PHIGS_LIST_UTILITIES (PICK_PROMPT_ECHO_TYPE,
        MAX_PICK_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of pick prompt and echo types.

-- PICK_REQUEST_STATUS

type PICK_REQUEST_STATUS is (OK,
                             NOPICK,
```

```
                                        NONE);

-- Defines the status of a pick input operation for the
-- request function.

-- PICK_STATUS

subtype PICK_STATUS is PICK_REQUEST_STATUS range OK..NOPICK;

-- Defines the status of pick input operation for the sample,
-- get, and inquiry functions.

-- STRUCTURE_ID

type STRUCTURE_ID is new PHIGS_NATURAL;

-- Defines a type for structure identifiers.

-- STRUCTURE_IDS

package STRUCTURE_IDS is
   new PHIGS_LIST_UTILITIES (STRUCTURE_ID,
                             MAX_STRUCTURE_IDS_SUPPORTED);

-- Provides for lists of structure identifiers.

-- PICK_PATH_ENTRY

type PICK_PATH_ENTRY is
  record
    STRUCTURE_IDENTIFIER : STRUCTURE_ID;
    PICK_IDENTIFIER      : PICK_ID;
    ELEMENT_NUMBER       : ELEMENT_POSITION;
  end record;

-- Defines the entries of a pick path.

-- PICK_PATH_ARRAY

type PICK_PATH_ARRAY is array (PATH_DEPTH range <>) of
                             PICK_PATH_ENTRY;

--  Provides for the specification of an unconstrained array of pick
--  path entries.

-- PICK_PATH

type PICK_PATH (DEPTH : PATH_DEPTH := 0) is
      record
        PATH : PICK_PATH_ARRAY (0..DEPTH);
      end record;

--  Provides for the specification of a pick path.

-- POLYLINE_FILL_AREA_CONTROL_FLAG

type POLYLINE_FILL_AREA_CONTROL_FLAG is (POLYLINE,
                                         FILL_AREA,
                                         FILL_AREA_SET);
```

```
-- Provides for indicating the type of primitive used
-- to produce locator prompt and echo type 5.

-- PROJECTION_TYPE

type PROJECTION_TYPE is (PARALLEL,
                         PERSPECTIVE);

-- Defines the type of projections supported by PHIGS.

-- RASTER_UNITS

type RASTER_UNITS is new PHIGS_POSITIVE;

-- Defines the range of raster units.

-- RASTER_UNIT_SIZE_3

type RASTER_UNIT_SIZE_3 is
  record
    X : RASTER_UNITS;
    Y : RASTER_UNITS;
    Z : RASTER_UNITS;
  end record;

-- Defines the size of an object in raster units on a raster
-- device.

-- RASTER_UNIT_SIZE_2

type RASTER_UNIT_SIZE_2 is
  record
    X : RASTER_UNITS;
    Y : RASTER_UNITS;
  end record;

-- Defines the size of an object in raster units on a raster
-- device.


-- REFERENCE_HANDLING_FLAG

type REFERENCE_HANDLING_FLAG is (DELETE,
                                 KEEP);

-- Provides for specification of the handling of deleted
-- structure networks.

-- REFERENCE_PATH_ENTRY

type REFERENCE_PATH_ENTRY is
  record
    STRUCTURE_IDENTIFIER : STRUCTURE_ID;
    ELEMENT_NUMBER       : ELEMENT_POSITION;
  end record;

-- Defines the entries of a reference path.

-- REFERENCE_PATH_ARRAY
```

```
type REFERENCE_PATH_ARRAY is array (PATH_DEPTH range <>) of
  REFERENCE_PATH_ENTRY;

-- Provides for the specification of a basic reference path.

-- REFERENCE_PATH

type REFERENCE_PATH (DEPTH : PATH_DEPTH := 0) is
  record
    PATH : REFERENCE_PATH_ARRAY (0..DEPTH);
  end record;

-- Provides for the specification of a variable size
-- reference path.

-- REFERENCE_PATHS

package REFERENCE_PATHS is
   new PHIGS_LIST_UTILITIES (REFERENCE_PATH,
                             MAX_REFERENCE_PATHS_SUPPORTED);

-- Provides for the specification of lists of reference
-- paths.

-- RELATIVE_PRIORITY

type RELATIVE_PRIORITY is (HIGHER,
                           LOWER);

-- Indicates the relative input priority between two views.

-- RETURN_VALUE_TYPE

type RETURN_VALUE_TYPE is (SET,
                           REALIZED);

-- Indicates whether the returned values should be as they
-- were set by the program or as they were actually realized
-- on the device.

-- SEARCH_DIRECTION

type SEARCH_DIRECTION is (BACKWARD,
                          FORWARD);

-- This type provides for indicating search direction.

-- SEARCH_STATUS_INDICATOR

type SEARCH_STATUS_INDICATOR is (FAILURE,
                                 SUCCESS);

-- This type provides for indicating the result of a search.

-- STRING_DEVICE_NUMBER

type STRING_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for string device identifiers.
```

```
-- STRING_PROMPT_ECHO_TYPE

type STRING_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the string prompt and echo types.

-- STRING_PROMPT_ECHO_TYPES

package STRING_PROMPT_ECHO_TYPES is
   new PHIGS_LIST_UTILITIES (STRING_PROMPT_ECHO_TYPE,
                             MAX_STRING_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of string prompt and echo types.

-- STROKE DEVICE NUMBER

type STROKE_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for stroke device identifiers.

-- STROKE_PROMPT_ECHO_TYPE

type STROKE_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the stroke prompt and echo types.

-- STROKE_PROMPT_ECHO_TYPES

package STROKE_PROMPT_ECHO_TYPES is
   new PHIGS_LIST_UTILITIES (STROKE_PROMPT_ECHO_TYPE,
                             MAX_STROKE_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of stroke prompt and echo types

-- STRUCTURE_ELEMENT_TYPE

subtype STRUCTURE_ELEMENT_TYPE is
        ELEMENT_TYPE range NIL..ELEMENT_TYPE'LAST;

--  Provides for identifying the types of individual
--  structure elements.

-- STRUCTURE_NETWORK_SOURCE

type STRUCTURE_NETWORK_SOURCE is (CSS,
                                  ARCHIVE);

--  Provides for indicating source for check of conflicting
--  structure identifiers.

-- DISPLAY_PRIORITY

type DISPLAY_PRIORITY is digits PHIGS_PRECISION range 0.0..1.0;

-- Defines type for specifying structure display priority.


-- STRUCTURE_STATE
```

```
type STRUCTURE_STATE is (STCL,
                         STOP);

-- The type used to return the structure state value.

-- STRUCTURE_STATUS

type STRUCTURE_STATUS is (NON_EXISTENT,
                          EMPTY,
                          NOTEMPTY);

--  The type used to return the structure status.

-- POSTED_STRUCTURE

type POSTED_STRUCTURE is
  record
    STRUCTURE_IDENTIFIER : STRUCTURE_ID;
    PRIORITY             : DISPLAY_PRIORITY;
  end record;

--  Defines a data type for describing posted structure networks.

-- POSTED_STRUCTURES

package POSTED_STRUCTURES is
        new PHIGS_LIST_UTILITIES (POSTED_STRUCTURE,
                                  MAX_POSTED_STRUCTURES_SUPPORTED);

--  Provides for the specification of lists of posted structure
--  networks.


-- SUBPROGRAM_NAME

subtype SUBPROGRAM_NAME is PHIGS_STRING;

-- Defines the name of a PHIGS function detecting an error.

-- SYSTEM_STATE

type SYSTEM_STATE is (PHCL,
                      PHOP);

-- The type used to return the system state value.

-- VERTICAL_ALIGNMENT

type VERTICAL_ALIGNMENT is (NORMAL,
                            TOP,
                            CAP,
                            HALF,
                            BASE,
                            BOTTOM);

-- The alignment of the text extent parallelogram with respect
-- to the vertical positioning of the text.

-- TEXT_ALIGNMENT
```

```
type TEXT_ALIGNMENT is
  record
    HORIZONTAL : HORIZONTAL_ALIGNMENT;
    VERTICAL   : VERTICAL_ALIGNMENT;
  end record;

-- The type of the attribute controlling the positioning of
-- the text extent parallelogram in relation to the text
-- position, having horizontal and vertical components as
-- defined above.


-- TEXT_FONT

type TEXT_FONT is new PHIGS_INTEGER;

-- Defines the types of fonts provided by the implementation.

-- TEXT_PRECISION

type TEXT_PRECISION is (STRING_PRECISION,
                        CHAR_PRECISION,
                        STROKE_PRECISION);

-- The precision with which text appears.

-- TEXT_FONT_PRECISION

type TEXT_FONT_PRECISION is
  record
    FONT      : TEXT_FONT;
    PRECISION : TEXT_PRECISION;
  end record;

-- This type defines a record describing the text font and
-- precision supported.

-- TEXT_FONT_PRECISIONS

package TEXT_FONT_PRECISIONS is
   new PHIGS_LIST_UTILITIES (TEXT_FONT_PRECISION,
                             MAX_TEXT_FONT_PRECISION_PAIRS_SUPPORTED);

-- Provides for lists of text font and precision pairs.

-- TEXT_INDEX

type TEXT_INDEX is new PHIGS_POSITIVE;

-- Defines the range of text bundle table indices.

-- TEXT_INDICES

package TEXT_INDICES is
          new PHIGS_LIST_UTILITIES (TEXT_INDEX,
                                    MAX_TEXT_INDICES_SUPPORTED);

-- Provides for lists of text indices.
```

```
-- TEXT_PATH

type TEXT_PATH is (RIGHT,
                   LEFT,
                   UP,
                   DOWN);

-- The direction taken by a text string.

-- TRANSFORMATION_MATRIX_2

type TRANSFORMATION_MATRIX_2 is
        array (1..3, 1..3) of MC_TYPE;

-- For 2D modeling and view transformation matrices.

-- TRANSFORMATION_MATRIX_3

type TRANSFORMATION_MATRIX_3 is
        array (1..4, 1..4) of MC_TYPE;

-- For 3D modeling and view transformation matrices.

-- UPDATE_REGENERATION_FLAG

type UPDATE_REGENERATION_FLAG is (PERFORM,
                                  POSTPONE);

-- Flag indicating regeneration action on display.

-- UPDATE_STATE

type UPDATE_STATE is (NOTPENDING,
                      PENDING);

-- Indicates whether or not a workstation transformation
-- change has been requested and not yet provided.

-- VALUATOR_DEVICE_NUMBER

type VALUATOR_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for valuator device identifiers.

-- VALUATOR_PROMPT_ECHO_TYPE

type VALUATOR_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the possible range of valuator prompt and
-- echo types.

-- VALUATOR_PROMPT_ECHO_TYPES

package VALUATOR_PROMPT_ECHO_TYPES is
   new PHIGS_LIST_UTILITIES (VALUATOR_PROMPT_ECHO_TYPE,
                             MAX_VALUATOR_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of valuator prompt and echo types.
```

```
-- VALUATOR_VALUE

type VALUATOR_VALUE is digits PHIGS_PRECISION;

-- Defines the range of accuracy of valuator input values on
-- an implementation.


-- EVENT_DEVICE_NUMBER

type EVENT_DEVICE_NUMBER (CLASS : INPUT_CLASS := NONE) is
  record
    case CLASS is

      when NONE = > null;

      when LOCATOR_INPUT
        = > LOCATOR_EVENT_DEVICE : LOCATOR_DEVICE_NUMBER;

      when STROKE_INPUT
        = > STROKE_EVENT_DEVICE : STROKE_DEVICE_NUMBER;

      when VALUATOR_INPUT
        = > VALUATOR_EVENT_DEVICE : VALUATOR_DEVICE_NUMBER;

      when CHOICE_INPUT
        = > CHOICE_EVENT_DEVICE : CHOICE_DEVICE_NUMBER;

      when PICK_INPUT
        = > PICK_EVENT_DEVICE : PICK_DEVICE_NUMBER;

      when STRING_INPUT
        = > STRING_EVENT_DEVICE : STRING_DEVICE_NUMBER;

    end case;
  end record;

-- Provides for returning any class of device number from the
-- event queue.

-- EVENT_QUEUE_DEVICE_NUMBER

type EVENT_QUEUE_DEVICE_NUMBER
  (CLASS : INPUT_QUEUE_CLASS := LOCATOR_INPUT) is
      record
        case CLASS is

          when LOCATOR_INPUT
            = > LOCATOR_EVENT_DEVICE : LOCATOR_DEVICE_NUMBER;

          when STROKE_INPUT
            = > STROKE_EVENT_DEVICE : STROKE_DEVICE_NUMBER;

          when VALUATOR_INPUT
            = > VALUATOR_EVENT_DEVICE : VALUATOR_DEVICE_NUMBER;

          when CHOICE_INPUT
            = > CHOICE_EVENT_DEVICE : CHOICE_DEVICE_NUMBER;
```

```
            when PICK_INPUT
               => PICK_EVENT_DEVICE : PICK_DEVICE_NUMBER;

            when STRING_INPUT
               => STRING_EVENT_DEVICE : STRING_DEVICE_NUMBER;
         end case;
      end record;

-- Allows EVENT_DEVICE_NUMBERs which cannot have value NONE.  This is
-- used when returning only real input classes as causes of event
-- queue overflow and for flushing the queue.

-- ACCESS_COLOUR_MATRIX

type ACCESS_COLOUR_MATRIX is access COLOUR_MATRIX;

-- Provides for returning variable sized matrices containing colour
-- indices corresponding to a cell array or pattern array.

-- VARIABLE_CONNECTION_ID

type VARIABLE_CONNECTION_ID (LENGTH : STRING_SMALL_NATURAL := 0) is
  record
    CONNECT : CONNECTION_ID (1..LENGTH);
  end record;

-- Defines a variable length connection identifier for
-- INQ_WS_CONNECTION_AND_TYPE.


-- ACCESS_STRING

type ACCESS_STRING is access PHIGS_STRING;

--  Defines a variable length string needed to store strings in
--  structure element records.


-- VIEW_INDEX

type VIEW_INDEX is new PHIGS_NATURAL;

-- Defines a type for specifying view indices.

-- VIEW_INDICES

package VIEW_INDICES is
        new PHIGS_LIST_UTILITIES (VIEW_INDEX,
                                  MAX_VIEW_INDICES_SUPPORTED);


--  Provides for lists of view indices.

-- VISUAL REPRESENTATION STATE

type VISUAL_REPRESENTATION_STATE is (CORRECT,
                                     DEFERRED,
                                     SIMULATED);
```

```
-- Specifies state of image on display surface.

-- WS_CATEGORY

type WS_CATEGORY is (OUTPUT,
                     INPUT,
                     OUTIN,
                     MO,
                     MI);

-- Type for PHIGS workstation categories.

-- WS_DEPENDENCY

type WS_DEPENDENCY is (WS_INDEPENDENT,
                       WS_DEPENDENT);

--  Type for indications of workstation dependency.

-- WS_DEPENDENCIES

package WS_DEPENDENCIES is
   new PHIGS_LIST_UTILITIES (WS_DEPENDENCY,
                             MAX_GSE_IDS_SUPPORTED);

--  Provides for lists of workstation dependencies.

-- WS_ID

type WS_ID is new PHIGS_POSITIVE;

-- Defines the range of workstation identifiers.

-- WS_IDS

package WS_IDS is new PHIGS_LIST_UTILITIES (WS_ID,
                                            MAX_OPEN_WS_SUPPORTED);

-- Provides for lists of workstation identifiers.

-- WS_STATE

type WS_STATE is (WSCL,
                  WSOP);

-- The type used to return the workstation state value.

-- WS_TYPE

type WS_TYPE is new PHIGS_POSITIVE;

-- Range of values corresponding to valid workstation types.
-- Constants specifying names for the various types of
-- workstations should be provided by an implementation.

-- WS_TYPES

package WS_TYPES is new PHIGS_LIST_UTILITIES (WS_TYPE,
                                              MAX_WS_TYPES_SUPPORTED);
```

```
-- Provides for lists of workstation types.

-- STRUCTURE_ELEMENT_RECORD

type STRUCTURE_ELEMENT_RECORD
  (ELEMENT_TYPE : STRUCTURE_ELEMENT_TYPE := NIL) is
      record
        case ELEMENT_TYPE is

          -- The empty element

          when NIL = >
             null;

          -- Primitive elements

          when POLYLINE_3 = >
             POLYLINE_3_POINTS : MC.ACCESS_POINT_LIST_3;

          when POLYLINE = >
             POLYLINE_POINTS : MC.ACCESS_POINT_LIST_2;

          when POLYMARKER_3 = >
             POLYMARKER_3_POINTS : MC.ACCESS_POINT_LIST_3;

          when POLYMARKER = >
             POLYMARKER_POINTS : MC.ACCESS_POINT_LIST_2;

          when TEXT_3 = >
             TEXT_3_POINT : MC.POINT_3;
             TEXT_DIRECTION_VECTORS : MC.VECTOR_PAIR_3;
             TEXT_3_CHAR_STRING : ACCESS_STRING;

          when TEXT = >
             TEXT_POINT : MC.POINT_2;
             TEXT_CHAR_STRING : ACCESS_STRING;

          when ANNOTATION_TEXT_RELATIVE_3 = >
             ANNOTATION_TEXT_RELATIVE_3_REF_POINT : MC.POINT_3;
             ANNOTATION_TEXT_RELATIVE_3_OFFSET : NPC.POINT_3;
             ANNOTATION_TEXT_RELATIVE_3_CHAR_STRING : ACCESS_STRING;

          when ANNOTATION_TEXT_RELATIVE = >
             ANNOTATION_TEXT_RELATIVE_REF_POINT : MC.POINT_2;
             ANNOTATION_TEXT_RELATIVE_OFFSET : NPC.POINT_2;
             ANNOTATION_TEXT_RELATIVE_CHAR_STRING : ACCESS_STRING;

          when FILL_AREA_3 = >
             FILL_AREA_3_POINTS : MC.ACCESS_POINT_LIST_3;

          when FILL_AREA = >
             FILL_AREA_POINTS : MC.ACCESS_POINT_LIST_2;

          when FILL_AREA_SET_3 = >
             FILL_AREA_SET_3_POINT_LISTS : MC.ACCESS_LIST_OF_POINT_LIST_3;

          when FILL_AREA_SET = >
             FILL_AREA_SET_POINT_LISTS : MC.ACCESS_LIST_OF_POINT_LIST_2;
```

```
when CELL_ARRAY_3 = >
   CORNER_P_3 : MC.POINT_3;
   CORNER_Q_3 : MC.POINT_3;
   CORNER_R_3 : MC.POINT_3;
   CELL_ARRAY_3_CELLS : ACCESS_COLOUR_MATRIX;

when CELL_ARRAY = >
   CORNER_P : MC.POINT_2;
   CORNER_Q : MC.POINT_2;
   CELL_ARRAY_CELLS : ACCESS_COLOUR_MATRIX;

when GDP_3 = >
   GDP_3_POINTS : MC.ACCESS_POINT_LIST_3;
   GDP_3_DATA : GDP_3_RECORD;

when GDP = >
   GDP_POINTS : MC.ACCESS_POINT_LIST_2;
   GDP_DATA : GDP_RECORD;

-- Bundle index elements

when SET_POLYLINE_INDEX = >
   POLYLINE_IND : POLYLINE_INDEX;

when SET_POLYMARKER_INDEX = >
   POLYMARKER_IND : POLYMARKER_INDEX;

when SET_TEXT_INDEX = >
   TEXT_IND : TEXT_INDEX;

when SET_INTERIOR_INDEX = >
   INTERIOR_IND : INTERIOR_INDEX;

when SET_EDGE_INDEX = >
   EDGE_IND : EDGE_INDEX;

-- Individual aspect elements

when SET_LINETYPE = >
   TYPE_OF_LINE : LINETYPE;

when SET_LINEWIDTH_SCALE_FACTOR = >
   LINEWIDTH_SF : LINEWIDTH;

when SET_POLYLINE_COLOUR_INDEX = >
   LINE_COLOUR : COLOUR_INDEX;

when SET_MARKER_TYPE = >
   TYPE_OF_MARKER : MARKER_TYPE;

when SET_MARKER_SIZE_SCALE_FACTOR = >
   SIZE : MARKER_SIZE;

when SET_POLYMARKER_COLOUR_INDEX = >
   MARKER_COLOUR : COLOUR_INDEX;

when SET_TEXT_FONT = >
   FONT : TEXT_FONT;
```

```
when SET_TEXT_PRECISION = >
   PRECISION : TEXT_PRECISION;

when SET_CHAR_EXPANSION_FACTOR = >
   EXPANSION : CHAR_EXPANSION;

when SET_CHAR_SPACING = >
   SPACING : CHAR_SPACING;

when SET_TEXT_COLOUR_INDEX = >
   TEXT_COLOUR : COLOUR_INDEX;

when SET_CHAR_HEIGHT = >
   HEIGHT : MC.MAGNITUDE;

when SET_CHAR_UP_VECTOR = >
   CHAR_UP_VECTOR : MC.VECTOR_2;

when SET_TEXT_PATH = >
   PATH : TEXT_PATH;

when SET_TEXT_ALIGNMENT = >
   ALIGNMENT : TEXT_ALIGNMENT;

when SET_ANNOTATION_TEXT_CHAR_HEIGHT = >
   ANNOTATION_HEIGHT : NPC.MAGNITUDE;

when SET_ANNOTATION_TEXT_CHAR_UP_VECTOR = >
   ANNOTATION_CHAR_UP_VECTOR : NPC.VECTOR_2;

when SET_ANNOTATION_TEXT_PATH = >
   ANNOTATION_PATH : TEXT_PATH;

when SET_ANNOTATION_TEXT_ALIGNMENT = >
   ANNOTATION_ALIGNMENT : TEXT_ALIGNMENT;

when SET_ANNOTATION_STYLE = >
   STYLE_OF_ANNOTATION : ANNOTATION_STYLE;

when SET_INTERIOR_STYLE = >
   STYLE_OF_INTERIOR : INTERIOR_STYLE;

when SET_INTERIOR_STYLE_INDEX = >
   STYLE_IND : STYLE_INDEX;

when SET_INTERIOR_COLOUR_INDEX = >
   INTERIOR_COLOUR : COLOUR_INDEX;

when SET_EDGE_FLAG = >
   FLAG : EDGE_FLAG;

when SET_EDGETYPE = >
   TYPE_OF_EDGE : EDGETYPE;

when SET_EDGEWIDTH_SCALE_FACTOR = >
   EDGEWIDTH_SF : EDGEWIDTH;

when SET_EDGE_COLOUR_INDEX = >
   EDGE_COLOUR : COLOUR_INDEX;
```

```
-- Pattern attribute elements

when SET_PATTERN_SIZE = >
   PATTERN_SIZE : MC.SIZE_2;

when SET_PATTERN_REFERENCE_POINT_AND_VECTORS = >
   PATTERN_REFERENCE_POINT_3 : MC.POINT_3;
   PATTERN_REFERENCE_VECTORS : MC.VECTOR_PAIR_3;

when SET_PATTERN_REFERENCE_POINT = >
   PATTERN_REFERENCE_POINT : MC.POINT_2;

-- Name set elements

when ADD_NAMES_TO_SET = >
   NAMES_TO_ADD : NAME_SET;

when REMOVE_NAMES_FROM_SET = >
   NAMES_TO_REMOVE : NAME_SET;

-- ASF elements

when SET_INDIVIDUAL_ASF = >
   ATTRIBUTE_ID : ASPECT;
   SOURCE_FLAG : ASF;

-- HLHSR elements

when SET_HLHSR_IDENTIFIER = >
   HLHSR_IDENTIFIER : HLHSR_ID;

-- Transformation elements

when SET_LOCAL_TRANSFORMATION_3 = >
   LOCAL_MATRIX_3 : TRANSFORMATION_MATRIX_3;
   HOW_APPLIED_3 : COMPOSITION_TYPE;

when SET_LOCAL_TRANSFORMATION = >
   LOCAL_MATRIX : TRANSFORMATION_MATRIX_2;
   HOW_APPLIED : COMPOSITION_TYPE;

when SET_GLOBAL_TRANSFORMATION_3 = >
   GLOBAL_MATRIX_3 : TRANSFORMATION_MATRIX_3;

when SET_GLOBAL_TRANSFORMATION = >
   GLOBAL_MATRIX : TRANSFORMATION_MATRIX_2;

when SET_MODELLING_CLIPPING_VOLUME_3 = >
   MODELLING_CLIPPING_OPERATOR_3 : MODELLING_CLIP_OPERATION_TYPE;
   MODELLING_CLIPPING_LIMITS_3 : MC.ACCESS_HALF_SPACE_LIST_3;

when SET_MODELLING_CLIPPING_VOLUME = >
   MODELLING_CLIPPING_OPERATOR : MODELLING_CLIP_OPERATION_TYPE;
   MODELLING_CLIPPING_LIMITS : MC.ACCESS_HALF_SPACE_LIST_2;

when SET_MODELLING_CLIPPING_INDICATOR = >
   MODELLING_CLIPPING_INDICATOR : CLIPPING_INDICATOR;

when RESTORE_MODELLING_CLIPPING_VOLUME = >
```

```
            null;

         when SET_VIEW_INDEX = >
            VIEW_IND : VIEW_INDEX;

         -- Invocation elements

         when EXECUTE_STRUCTURE = >
            STRUCTURE_IDENTIFIER : STRUCTURE_ID;

         -- Structure content identification elements

         when LABEL = >
            LABEL_IDENTIFIER : LABEL_ID;

         when APPLICATION_DATA = >
            DATA : APPLICATION_DATA_RECORD;

         when GSE = >
            GSE_DATA : GSE_RECORD;

         when SET_PICK_IDENTIFIER = >
            PICK_IDENTIFIER : PICK_ID;

      end case;
   end record;

-- This type defines the format of structure contents and is used to
-- return structure elements during inquiry.

PHIGS_ERROR : exception;

-- Exception to notify the application program of an error in a
-- PHIGS procedure should the version of ERROR_HANDLING which raises
-- an exception be used.

-- This section contains the constants that provide the PHIGS standard
-- values defined for some PHIGS/Ada types.  The constants for colour
-- models are defined earlier.

-- The following constants define the PHIGS standard line types:

SOLID_LINE                  : constant LINETYPE := 1;
DASHED_LINE                 : constant LINETYPE := 2;
DOTTED_LINE                 : constant LINETYPE := 3;
DASHED_DOTTED_LINE          : constant LINETYPE := 4;

-- The following constants define the PHIGS standard marker types:

DOT_MARKER                  : constant MARKER_TYPE := 1;
PLUS_MARKER                 : constant MARKER_TYPE := 2;
STAR_MARKER                 : constant MARKER_TYPE := 3;
ZERO_MARKER                 : constant MARKER_TYPE := 4;
X_MARKER                    : constant MARKER_TYPE := 5;

-- The following constants define the PHIGS standard edgetypes:

SOLID_EDGE                  : constant EDGETYPE := 1;
DASHED_EDGE                 : constant EDGETYPE := 2;
```

```
   DOTTED_EDGE                    : constant EDGETYPE := 3;
   DASHED_DOTTED_EDGE             : constant EDGETYPE := 4;

   -- The following constants define the PHIGS standard annotation styles:

   UNCONNECTED_ANNOTATION         : constant ANNOTATION_STYLE := 1;
   LEAD_LINE_ANNOTATION           : constant ANNOTATION_STYLE := 2;

   -- The following constants are used for defining the modelling clipping
   -- operators specified by PHIGS:

   REPLACE_VOLUME                 : constant MODELLING_CLIP_OPERATION_TYPE := 1;
   INTERSECT_VOLUME               : constant MODELLING_CLIP_OPERATION_TYPE := 2;

   -- The following constants define the prompt and echo types supported
   -- by PHIGS:

   DEFAULT_LOCATOR                : constant LOCATOR_PROMPT_ECHO_TYPE := 1;
   CROSS_HAIR_LOCATOR             : constant LOCATOR_PROMPT_ECHO_TYPE := 2;
   TRACKING_CROSS_LOCATOR         : constant LOCATOR_PROMPT_ECHO_TYPE := 3;
   RUBBER_BAND_LINE_LOCATOR       : constant LOCATOR_PROMPT_ECHO_TYPE := 4;
   RECTANGLE_LOCATOR              : constant LOCATOR_PROMPT_ECHO_TYPE := 5;
   DIGITAL_LOCATOR                : constant LOCATOR_PROMPT_ECHO_TYPE := 6;

   DEFAULT_STROKE                 : constant STROKE_PROMPT_ECHO_TYPE := 1;
   DIGITAL_STROKE                 : constant STROKE_PROMPT_ECHO_TYPE := 2;
   MARKER_STROKE                  : constant STROKE_PROMPT_ECHO_TYPE := 3;
   LINE_STROKE                    : constant STROKE_PROMPT_ECHO_TYPE := 4;

   DEFAULT_VALUATOR               : constant VALUATOR_PROMPT_ECHO_TYPE := 1;
   GRAPHICAL_VALUATOR             : constant VALUATOR_PROMPT_ECHO_TYPE := 2;
   DIGITAL_VALUATOR               : constant VALUATOR_PROMPT_ECHO_TYPE := 3;

   DEFAULT_CHOICE                 : constant CHOICE_PROMPT_ECHO_TYPE := 1;
   PROMPT_ECHO_CHOICE             : constant CHOICE_PROMPT_ECHO_TYPE := 2;
   STRING_PROMPT_CHOICE           : constant CHOICE_PROMPT_ECHO_TYPE := 3;
   STRING_INPUT_CHOICE            : constant CHOICE_PROMPT_ECHO_TYPE := 4;
   STRUCTURE_CHOICE               : constant CHOICE_PROMPT_ECHO_TYPE := 5;

   DEFAULT_PICK                   : constant PICK_PROMPT_ECHO_TYPE := 1;
   GROUP_HIGHLIGHT_PICK           : constant PICK_PROMPT_ECHO_TYPE := 2;
   STRUCTURE_HIGHLIGHT_PICK       : constant PICK_PROMPT_ECHO_TYPE := 3;

end PHIGS_TYPES;
```

```
-- The PHIGS Package

with PHIGS_CONFIGURATION, PHIGS_TYPES, PHIGS_NAME_SET_FACILITY;
use PHIGS_CONFIGURATION, PHIGS_TYPES;

package PHIGS is

  -- The package PHIGS contains all of the procedures that are required
  -- to implement PHIGS.

  -- Private Type Declarations

  -- The following data types are the PHIGS private types and are
  -- included in package PHIGS for ease of manipulation.

  -- METAFILE_DATA_RECORD

  type METAFILE_DATA_RECORD (TYPE_OF_ITEM : METAFILE_ITEM_TYPE := 0) is private;

  --  A data record for metafiles.

  -- CHOICE_DATA_RECORD

  type CHOICE_DATA_RECORD
    (PROMPT_ECHO_TYPE : CHOICE_PROMPT_ECHO_TYPE := 1) is private;

  -- Defines a choice input data record.

  -- LOCATOR_DATA_RECORD

  type LOCATOR_DATA_RECORD
    (PROMPT_ECHO_TYPE : LOCATOR_PROMPT_ECHO_TYPE := 1) is private;

  -- Defines a locator input data record.

  -- PICK_DATA_RECORD

  type PICK_DATA_RECORD
    (PROMPT_ECHO_TYPE : PICK_PROMPT_ECHO_TYPE := 1) is private;

  -- Defines a pick input data record.

  -- STRING_DATA_RECORD

  type STRING_DATA_RECORD
    (PROMPT_ECHO_TYPE : STRING_PROMPT_ECHO_TYPE := 1) is private;

  -- Defines a string input data record.

  -- STROKE_DATA_RECORD

  type STROKE_DATA_RECORD
    (PROMPT_ECHO_TYPE : STROKE_PROMPT_ECHO_TYPE := 1) is private;

  -- Defines a stroke input data record.

  -- VALUATOR_DATA_RECORD
```

```
type VALUATOR_DATA_RECORD
  (PROMPT_ECHO_TYPE : VALUATOR_PROMPT_ECHO_TYPE := 1) is private;

-- Defines a valuator data record.

-- PHIGS Procedures

-- CONTROL FUNCTIONS

procedure OPEN_PHIGS
  (ERROR_FILE           : in FILE_ID := PHIGS_STRING(DEFAULT_ERROR_FILE);
   AMOUNT_OF_MEMORY     : in PHIGS_NATURAL := DEFAULT_MEMORY_UNITS);

procedure CLOSE_PHIGS;

procedure OPEN_WS
  (WS                   : in WS_ID;
   CONNECTION           : in CONNECTION_ID;
   TYPE_OF_WS           : in WS_TYPE);

procedure CLOSE_WS
  (WS : in WS_ID);

procedure REDRAW_ALL_STRUCTURES
  (WS       : in WS_ID;
   FLAG     : in CONTROL_FLAG);

procedure UPDATE_WS
  (WS                   : in WS_ID;
   REGENERATION         : in UPDATE_REGENERATION_FLAG);

procedure SET_DISPLAY_UPDATE_STATE
  (WS                   : in WS_ID;
   DEFERRAL             : in DEFERRAL_MODE;
   MODIFICATION         : in MODIFICATION_MODE);

procedure MESSAGE
  (WS                   : in WS_ID;
   CONTENTS             : in PHIGS_STRING);

-- OUTPUT FUNCTIONS

procedure POLYLINE
  (POINTS : in MC.POINT_LIST_3);

procedure POLYLINE
  (POINTS : in MC.POINT_LIST_2);

procedure POLYMARKER
  (POINTS : in MC.POINT_LIST_3);

procedure POLYMARKER
  (POINTS : in MC.POINT_LIST_2);

procedure TEXT
  (POSITION             : in MC.POINT_3;
   DIRECTION_VECTORS    : in MC.VECTOR_PAIR_3;
   CHAR_STRING          : in PHIGS_STRING);
```

```
procedure TEXT
  (POSITION       : in MC.POINT_2;
   CHAR_STRING    : in PHIGS_STRING);


procedure ANNOTATION_TEXT_RELATIVE
  (REFERENCE_POINT    : in MC.POINT_3;
   ANNOTATION_OFFSET  : in NPC.POINT_3;
   CHAR_STRING        : in PHIGS_STRING);

procedure ANNOTATION_TEXT_RELATIVE
  (REFERENCE_POINT    : in MC.POINT_2;
   ANNOTATION_OFFSET  : in NPC.POINT_2;
   CHAR_STRING        : in PHIGS_STRING);

procedure FILL_AREA
  (POINTS : in MC.POINT_LIST_3);

procedure FILL_AREA
  (POINTS : in MC.POINT_LIST_2);

procedure FILL_AREA_SET
  (POINT_LISTS : in MC.LIST_OF_POINT_LIST_3);

procedure FILL_AREA_SET
  (POINT_LISTS : in MC.LIST_OF_POINT_LIST_2);

procedure CELL_ARRAY
  (CORNER_P  : in MC.POINT_3;
   CORNER_Q  : in MC.POINT_3;
   CORNER_R  : in MC.POINT_3;
   CELLS     : in COLOUR_MATRIX);

procedure CELL_ARRAY
  (CORNER_P  : in MC.POINT_2;
   CORNER_Q  : in MC.POINT_2;
   CELLS     : in COLOUR_MATRIX);

-- OUTPUT ATTRIBUTE FUNCTIONS

procedure SET_POLYLINE_INDEX
  (POLYLINE_IND : in POLYLINE_INDEX);

procedure SET_POLYMARKER_INDEX
  (POLYMARKER_IND : in POLYMARKER_INDEX);

procedure SET_TEXT_INDEX
  (TEXT_IND : in TEXT_INDEX);

procedure SET_INTERIOR_INDEX
  (INTERIOR_IND : in INTERIOR_INDEX);

procedure SET_EDGE_INDEX
  (EDGE_IND : in EDGE_INDEX);

procedure SET_LINETYPE
  (TYPE_OF_LINE : in LINETYPE);

procedure SET_LINEWIDTH_SCALE_FACTOR
```

```
    (LINEWIDTH_SF : in LINEWIDTH);

  procedure SET_POLYLINE_COLOUR_INDEX
    (LINE_COLOUR : in COLOUR_INDEX);

  procedure SET_MARKER_TYPE
    (TYPE_OF_MARKER : in MARKER_TYPE);

  procedure SET_MARKER_SIZE_SCALE_FACTOR
    (SIZE : in MARKER_SIZE);

  procedure SET_POLYMARKER_COLOUR_INDEX
    (MARKER_COLOUR : in COLOUR_INDEX);

  procedure SET_TEXT_FONT
    (FONT : in TEXT_FONT);

  procedure SET_TEXT_PRECISION
    (PRECISION : in TEXT_PRECISION);

  procedure SET_CHAR_EXPANSION_FACTOR
    (EXPANSION : in CHAR_EXPANSION);

  procedure SET_CHAR_SPACING
    (SPACING : in CHAR_SPACING);

  procedure SET_TEXT_COLOUR_INDEX
    (TEXT_COLOUR : in COLOUR_INDEX);

  procedure SET_CHAR_HEIGHT
    (HEIGHT : in MC.MAGNITUDE);

  procedure SET_CHAR_UP_VECTOR
    (CHAR_UP_VECTOR : in MC.VECTOR_2);

  procedure SET_TEXT_PATH
    (PATH : in TEXT_PATH);

  procedure SET_TEXT_ALIGNMENT
    (ALIGNMENT : in TEXT_ALIGNMENT);

  procedure SET_ANNOTATION_TEXT_CHAR_HEIGHT
    (ANNOTATION_HEIGHT : in NPC.MAGNITUDE);

  procedure SET_ANNOTATION_TEXT_CHAR_UP_VECTOR
    (ANNOTATION_CHAR_UP_VECTOR : in NPC.VECTOR_2);

  procedure SET_ANNOTATION_TEXT_PATH
    (ANNOTATION_PATH : in TEXT_PATH);

  procedure SET_ANNOTATION_TEXT_ALIGNMENT
    (ANNOTATION_ALIGNMENT : in TEXT_ALIGNMENT);

  procedure SET_ANNOTATION_STYLE
    (STYLE_OF_ANNOTATION : in ANNOTATION_STYLE);

  procedure SET_INTERIOR_STYLE
    (STYLE_OF_INTERIOR : in INTERIOR_STYLE);
```

```
procedure SET_INTERIOR_STYLE_INDEX
  (STYLE_IND : in STYLE_INDEX);

procedure SET_INTERIOR_COLOUR_INDEX
  (INTERIOR_COLOUR : in COLOUR_INDEX);

procedure SET_EDGE_FLAG
  (FLAG : in EDGE_FLAG);

procedure SET_EDGETYPE
  (TYPE_OF_EDGE : in EDGETYPE);

procedure SET_EDGEWIDTH_SCALE_FACTOR
  (EDGEWIDTH_SF : in EDGEWIDTH);

procedure SET_EDGE_COLOUR_INDEX
  (EDGE_COLOUR : in COLOUR_INDEX);

procedure SET_PATTERN_SIZE
  (SIZE : in MC.SIZE_2);

procedure SET_PATTERN_REFERENCE_POINT_AND_VECTORS
  (REFERENCE_POINT      : in MC.POINT_3;
   REFERENCE_VECTORS    : in MC.VECTOR_PAIR_3);

procedure SET_PATTERN_REFERENCE_POINT
  (REFERENCE_POINT : MC.POINT_2);

procedure ADD_NAMES_TO_SET
  (NAMES_TO_ADD : in NAME_SET);

procedure REMOVE_NAMES_FROM_SET
  (NAMES_TO_REMOVE : in NAME_SET);

procedure SET_INDIVIDUAL_ASF
  (ASPECT_ID       : in ASPECT;
   SOURCE_FLAG     : in ASF);

procedure SET_POLYLINE_REPRESENTATION
  (WS               : in WS_ID;
   POLYLINE_IND     : in POLYLINE_INDEX;
   TYPE_OF_LINE     : in LINETYPE;
   LINEWIDTH_SF     : in LINEWIDTH;
   LINE_COLOUR      : in COLOUR_INDEX);

procedure SET_POLYMARKER_REPRESENTATION
  (WS               : in WS_ID;
   POLYMARKER_IND   : in POLYMARKER_INDEX;
   TYPE_OF_MARKER   : in MARKER_TYPE;
   SIZE             : in MARKER_SIZE;
   MARKER_COLOUR    : in COLOUR_INDEX);

procedure SET_TEXT_REPRESENTATION
  (WS               : in WS_ID;
   TEXT_IND         : in TEXT_INDEX;
   FONT             : in TEXT_FONT;
   PRECISION        : in TEXT_PRECISION;
   EXPANSION        : in CHAR_EXPANSION;
   SPACING          : in CHAR_SPACING;
```

```
    TEXT_COLOUR  : in COLOUR_INDEX);

procedure SET_INTERIOR_REPRESENTATION
   (WS                  : in WS_ID;
    INTERIOR_IND        : in INTERIOR_INDEX;
    STYLE_OF_INTERIOR   : in INTERIOR_STYLE;
    STYLE_IND           : in STYLE_INDEX;
    INTERIOR_COLOUR     : in COLOUR_INDEX);

procedure SET_EDGE_REPRESENTATION
   (WS                  : in WS_ID;
    EDGE_IND            : in EDGE_INDEX;
    FLAG                : in EDGE_FLAG;
    TYPE_OF_EDGE        : in EDGETYPE;
    EDGEWIDTH_SF        : in EDGEWIDTH;
    EDGE_COLOUR         : in COLOUR_INDEX);

procedure SET_PATTERN_REPRESENTATION
   (WS                  : in WS_ID;
    PATTERN_IND         : in PATTERN_INDEX;
    PATTERN             : in COLOUR_MATRIX);

procedure SET_COLOUR_REPRESENTATION
   (WS                  : in WS_ID;
    COLOUR_IND          : in COLOUR_INDEX;
    COLOUR              : in COLOUR_REPRESENTATION);

procedure SET_HIGHLIGHTING_FILTER
   (WS                  : in WS_ID;
    HIGHLIGHTING        : in NAME_SET_FILTER);

procedure SET_INVISIBILITY_FILTER
   (WS                  : in WS_ID;
    INVISIBILITY        : in NAME_SET_FILTER);

procedure SET_COLOUR_MODEL
   (WS          : in WS_ID;
    MODEL       : in COLOUR_MODEL);

procedure SET_HLHSR_IDENTIFIER
   (HLHSR_IDENTIFIER : in HLHSR_ID);

procedure SET_HLHSR_MODE
   (WS          : in WS_ID;
    MODE        : in HLHSR_MODE);

-- TRANSFORMATION FUNCTIONS

procedure SET_LOCAL_TRANSFORMATION
   (MATRIX              : in TRANSFORMATION_MATRIX_3;
    HOW_APPLIED         : in COMPOSITION_TYPE);

procedure SET_LOCAL_TRANSFORMATION
   (MATRIX              : in TRANSFORMATION_MATRIX_2;
    HOW_APPLIED         : in COMPOSITION_TYPE);

procedure SET_GLOBAL_TRANSFORMATION
   (MATRIX : in TRANSFORMATION_MATRIX_3);
```

```
procedure SET_GLOBAL_TRANSFORMATION
  (MATRIX : in TRANSFORMATION_MATRIX_2);

procedure SET_MODELLING_CLIPPING_VOLUME
  (OPERATOR     : in MODELLING_CLIP_OPERATION_TYPE;
   HALF_SPACES  : in MC.HALF_SPACE_LIST_3);

procedure SET_MODELLING_CLIPPING_VOLUME
  (OPERATOR     : in MODELLING_CLIP_OPERATION_TYPE;
   HALF_SPACES  : in MC.HALF_SPACE_LIST_2);

procedure SET_MODELLING_CLIPPING_INDICATOR
  (MODELLING_CLIPPING : in CLIPPING_INDICATOR);

procedure RESTORE_MODELLING_CLIPPING_VOLUME;

procedure SET_VIEW_INDEX
  (VIEW_IND : in VIEW_INDEX);

procedure SET_VIEW_REPRESENTATION
  (WS                  : in WS_ID;
   VIEW_IND            : in VIEW_INDEX;
   ORIENTATION_MATRIX  : in TRANSFORMATION_MATRIX_3;
   MAPPING_MATRIX      : in TRANSFORMATION_MATRIX_3;
   CLIPPING_LIMITS     : in NPC.RECTANGULAR_REGION_3;
   XY_CLIPPING         : in CLIPPING_INDICATOR;
   BACK_CLIPPING       : in CLIPPING_INDICATOR;
   FRONT_CLIPPING      : in CLIPPING_INDICATOR);

procedure SET_VIEW_REPRESENTATION
  (WS                  : in WS_ID;
   VIEW_IND            : in VIEW_INDEX;
   ORIENTATION_MATRIX  : in TRANSFORMATION_MATRIX_2;
   MAPPING_MATRIX      : in TRANSFORMATION_MATRIX_2;
   CLIPPING_LIMITS     : in NPC.RECTANGULAR_REGION_2;
   XY_CLIPPING         : in CLIPPING_INDICATOR);

procedure SET_VIEW_TRANSFORMATION_INPUT_PRIORITY
  (WS                  : in WS_ID;
   VIEW_IND            : in VIEW_INDEX;
   REFERENCE_INDEX     : in VIEW_INDEX;
   PRIORITY            : in RELATIVE_PRIORITY);

procedure SET_WS_WINDOW
  (WS                  : in WS_ID;
   WINDOW_LIMITS       : in NPC.RECTANGULAR_REGION_3);

procedure SET_WS_WINDOW
  (WS                  : in WS_ID;
   WINDOW_LIMITS       : in NPC.RECTANGULAR_REGION_2);

procedure SET_WS_VIEWPORT
  (WS                  : in WS_ID;
   VIEWPORT_LIMITS     : in DC.RECTANGULAR_REGION_3);

procedure SET_WS_VIEWPORT
  (WS                  : in WS_ID;
   VIEWPORT_LIMITS     : in DC.RECTANGULAR_REGION_2);
```

```
-- TRANSFORMATION UTILITY FUNCTIONS

procedure TRANSLATE
  (VECTOR              : in  MC.VECTOR_3;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MATRIX              : out TRANSFORMATION_MATRIX_3);

procedure TRANSLATE
  (VECTOR              : in  MC.VECTOR_2;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MATRIX              : out TRANSFORMATION_MATRIX_2);

procedure SCALE
  (FACTOR              : in  MC.VECTOR_3;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MATRIX              : out TRANSFORMATION_MATRIX_3);

procedure SCALE
  (FACTOR              : in  MC.VECTOR_2;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MATRIX              : out TRANSFORMATION_MATRIX_2);

procedure ROTATE_X
  (ANGLE_X             : in  ANGLE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MATRIX              : out TRANSFORMATION_MATRIX_3);

procedure ROTATE_Y
  (ANGLE_Y             : in  ANGLE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MATRIX              : out TRANSFORMATION_MATRIX_3);

procedure ROTATE_Z
  (ANGLE_Z             : in  ANGLE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MATRIX              : out TRANSFORMATION_MATRIX_3);

procedure ROTATE
  (ANGLE_Z             : in  ANGLE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   MATRIX              : out TRANSFORMATION_MATRIX_2);

procedure COMPOSE_MATRIX
  (MATRIX_A            : in  TRANSFORMATION_MATRIX_3;
   MATRIX_B            : in  TRANSFORMATION_MATRIX_3;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   COMPOSED_MATRIX     : out TRANSFORMATION_MATRIX_3);

procedure COMPOSE_MATRIX
  (MATRIX_A            : in  TRANSFORMATION_MATRIX_2;
   MATRIX_B            : in  TRANSFORMATION_MATRIX_2;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   COMPOSED_MATRIX     : out TRANSFORMATION_MATRIX_2);

procedure TRANSFORM_POINT
  (POINT               : in  MC.POINT_3;
   MATRIX              : in  TRANSFORMATION_MATRIX_3;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   TRANSFORMED_POINT   : out MC.POINT_3);
```

```
procedure TRANSFORM_POINT
  (POINT                  : in  MC.POINT_2;
   MATRIX                 : in  TRANSFORMATION_MATRIX_2;
   ERROR_INDICATOR        : out ERROR_NUMBER;
   TRANSFORMED_POINT      : out MC.POINT_2);

procedure BUILD_TRANSFORMATION_MATRIX
  (FIXED_POINT            : in  MC.POINT_3;
   SHIFT_VECTOR           : in  MC.VECTOR_3;
   ANGLE_X                : in  ANGLE;
   ANGLE_Y                : in  ANGLE;
   ANGLE_Z                : in  ANGLE;
   SCALE_FACTORS          : in  MC.VECTOR_3;
   ERROR_INDICATOR        : out ERROR_NUMBER;
   MATRIX                 : out TRANSFORMATION_MATRIX_3);

procedure BUILD_TRANSFORMATION_MATRIX
  (FIXED_POINT            : in  MC.POINT_2;
   SHIFT_VECTOR           : in  MC.VECTOR_2;
   ANGLE_Z                : in  ANGLE;
   SCALE_FACTORS          : in  MC.VECTOR_2;
   ERROR_INDICATOR        : out ERROR_NUMBER;
   MATRIX                 : out TRANSFORMATION_MATRIX_2);

procedure COMPOSE_TRANSFORMATION_MATRIX
  (MATRIX                 : in  TRANSFORMATION_MATRIX_3;
   FIXED_POINT            : in  MC.POINT_3;
   SHIFT_VECTOR           : in  MC.VECTOR_3;
   ANGLE_X                : in  ANGLE;
   ANGLE_Y                : in  ANGLE;
   ANGLE_Z                : in  ANGLE;
   SCALE_FACTORS          : in  MC.VECTOR_3;
   ERROR_INDICATOR        : out ERROR_NUMBER;
   COMPOSED_MATRIX        : out TRANSFORMATION_MATRIX_3);

procedure COMPOSE_TRANSFORMATION_MATRIX
  (MATRIX                 : in  TRANSFORMATION_MATRIX_2;
   FIXED_POINT            : in  MC.POINT_2;
   SHIFT_VECTOR           : in  MC.VECTOR_2;
   ANGLE_Z                : in  ANGLE;
   SCALE_FACTORS          : in  MC.VECTOR_2;
   ERROR_INDICATOR        : out ERROR_NUMBER;
   COMPOSED_MATRIX        : out TRANSFORMATION_MATRIX_2);

procedure EVALUATE_VIEW_ORIENTATION_MATRIX
  (REFERENCE_POINT        : in  WC.POINT_3;
   NORMAL_VECTOR          : in  WC.VECTOR_3;
   UP_VECTOR              : in  WC.VECTOR_3;
   ERROR_INDICATOR        : out ERROR_NUMBER;
   ORIENTATION_MATRIX     : out TRANSFORMATION_MATRIX_3);

procedure EVALUATE_VIEW_ORIENTATION_MATRIX
  (REFERENCE_POINT        : in  WC.POINT_2;
   UP_VECTOR              : in  WC.VECTOR_2;
   ERROR_INDICATOR        : out ERROR_NUMBER;
   ORIENTATION_MATRIX     : out TRANSFORMATION_MATRIX_2);

procedure EVALUATE_VIEW_MAPPING_MATRIX
```

```
    (WINDOW_LIMITS          : in  VRC.RECTANGULAR_REGION_2;
     VIEWPORT_LIMITS        : in  NPC.RECTANGULAR_REGION_3;
     TYPE_OF_PROJECTION     : in  PROJECTION_TYPE;
     REFERENCE_POINT        : in  VRC.POINT_3;
     VIEW_DISTANCE          : in  VRC_TYPE;
     BACK_DISTANCE          : in  VRC_TYPE;
     FRONT_DISTANCE         : in  VRC_TYPE;
     ERROR_INDICATOR        : out ERROR_NUMBER;
     MAPPING_MATRIX         : out TRANSFORMATION_MATRIX_3);

  procedure EVALUATE_VIEW_MAPPING_MATRIX
    (WINDOW_LIMITS      : in  VRC.RECTANGULAR_REGION_2;
     VIEWPORT_LIMITS    : in  NPC.RECTANGULAR_REGION_2;
     ERROR_INDICATOR    : out ERROR_NUMBER;
     MAPPING_MATRIX     : out TRANSFORMATION_MATRIX_2);

  -- STRUCTURE MANIPULATION FUNCTIONS

  procedure OPEN_STRUCTURE
    (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);

  procedure CLOSE_STRUCTURE;

  procedure EXECUTE_STRUCTURE
    (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);


  procedure LABEL
    (LABEL_IDENTIFIER : in LABEL_ID);

  procedure APPLICATION_DATA
    (DATA : in APPLICATION_DATA_RECORD);

  procedure SET_EDIT_MODE
    (MODE : in EDIT_MODE);

  procedure COPY_ALL_ELEMENTS_FROM_STRUCTURE
    (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);

  procedure SET_ELEMENT_POINTER
    (POSITION : in ELEMENT_POSITION);

  procedure OFFSET_ELEMENT_POINTER
    (OFFSET : in ELEMENT_POSITION);

  procedure SET_ELEMENT_POINTER_AT_LABEL
    (LABEL_IDENTIFIER : in LABEL_ID);

  procedure DELETE_ELEMENT;

  procedure DELETE_ELEMENT_RANGE
    (POSITION_1  : in ELEMENT_POSITION;
     POSITION_2  : in ELEMENT_POSITION);

  procedure DELETE_ELEMENTS_BETWEEN_LABELS
    (LABEL_IDENTIFIER_1  : in LABEL_ID;
     LABEL_IDENTIFIER_2  : in LABEL_ID);

  procedure EMPTY_STRUCTURE
```

```
   (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);

procedure DELETE_STRUCTURE
   (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);

procedure DELETE_STRUCTURE_NETWORK
   (STRUCTURE_IDENTIFIER : in STRUCTURE_ID;
    HANDLING_FLAG        : in REFERENCE_HANDLING_FLAG);

procedure DELETE_ALL_STRUCTURES;

procedure CHANGE_STRUCTURE_IDENTIFIER
   (ORIGINAL_IDENTIFIER   : in STRUCTURE_ID;
    RESULTING_IDENTIFIER  : in STRUCTURE_ID);

procedure CHANGE_STRUCTURE_REFERENCES
   (ORIGINAL_IDENTIFIER   : in STRUCTURE_ID;
    RESULTING_IDENTIFIER  : in STRUCTURE_ID);

procedure CHANGE_STRUCTURE_IDENTIFIER_AND_REFERENCES
   (ORIGINAL_IDENTIFIER   : in STRUCTURE_ID;
    RESULTING_IDENTIFIER  : in STRUCTURE_ID);

procedure POST_STRUCTURE
   (WS                    : in WS_ID;
    STRUCTURE_IDENTIFIER  : in STRUCTURE_ID;
    PRIORITY              : in DISPLAY_PRIORITY);

procedure UNPOST_STRUCTURE
   (WS                    : in WS_ID;
    STRUCTURE_IDENTIFIER  : in STRUCTURE_ID);

procedure UNPOST_ALL_STRUCTURES
   (WS : in WS_ID);

-- STRUCTURE ARCHIVING FUNCTIONS

procedure OPEN_ARCHIVE_FILE
   (ARCHIVE_IDENTIFIER : in ARCHIVE_ID;
    ARCHIVE_FILE       : in FILE_ID);

procedure CLOSE_ARCHIVE_FILE
   (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);

procedure ARCHIVE_STRUCTURES
   (ARCHIVE_IDENTIFIER      : in ARCHIVE_ID;
    STRUCTURE_IDENTIFIERS   : in STRUCTURE_IDS.LIST_OF);

procedure ARCHIVE_STRUCTURE_NETWORKS
   (ARCHIVE_IDENTIFIER      : in ARCHIVE_ID;
    STRUCTURE_IDENTIFIERS   : in STRUCTURE_IDS.LIST_OF);

procedure ARCHIVE_ALL_STRUCTURES
   (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);

procedure SET_CONFLICT_RESOLUTION
   (ARCHIVAL_CONFLICT    : in CONFLICT_RESOLUTION;
    RETRIEVAL_CONFLICT   : in CONFLICT_RESOLUTION);
```

```
procedure RETRIEVE_STRUCTURE_IDENTIFIERS
  (ARCHIVE_IDENTIFIER      : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS   : out STRUCTURE_IDS.LIST_OF);

procedure RETRIEVE_STRUCTURES
  (ARCHIVE_IDENTIFIER      : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS   : in STRUCTURE_IDS.LIST_OF);

procedure RETRIEVE_STRUCTURE_NETWORKS
  (ARCHIVE_IDENTIFIER      : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS   : in STRUCTURE_IDS.LIST_OF);

procedure RETRIEVE_ALL_STRUCTURES
  (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);

procedure RETRIEVE_PATHS_TO_ANCESTORS
  (ARCHIVE_IDENTIFIER      : in  ARCHIVE_ID;
   STRUCTURE_IDENTIFIER    : in  STRUCTURE_ID;
   ORDER                   : in  PATH_ORDER;
   DEPTH                   : in  PATH_DEPTH;
   LIST_OF_PATHS           : out REFERENCE_PATHS.LIST_OF);

procedure RETRIEVE_PATHS_TO_DESCENDANTS
  (ARCHIVE_IDENTIFIER      : in  ARCHIVE_ID;
   STRUCTURE_IDENTIFIER    : in  STRUCTURE_ID;
   ORDER                   : in  PATH_ORDER;
   DEPTH                   : in  PATH_DEPTH;
   LIST_OF_PATHS           : out REFERENCE_PATHS.LIST_OF);

procedure DELETE_STRUCTURES_FROM_ARCHIVE
  (ARCHIVE_IDENTIFIER      : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS   : in STRUCTURE_IDS.LIST_OF);

procedure DELETE_STRUCTURE_NETWORKS_FROM_ARCHIVE
  (ARCHIVE_IDENTIFIER      : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS   : in STRUCTURE_IDS.LIST_OF);

procedure DELETE_ALL_STRUCTURES_FROM_ARCHIVE
  (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);

-- INPUT FUNCTIONS

procedure SET_PICK_IDENTIFIER
  (PICK_IDENTIFIER : in PICK_ID);

procedure SET_PICK_FILTER
  (WS              : in WS_ID;
   PICK_DEVICE     : in PICK_DEVICE_NUMBER;
   PICKABILITY     : in NAME_SET_FILTER);

procedure INITIALIZE_LOCATOR
  (WS                  : in WS_ID;
   DEVICE              : in LOCATOR_DEVICE_NUMBER;
   INITIAL_VIEW_IND    : in VIEW_INDEX;
   INITIAL_POSITION    : in WC.POINT_3;
   ECHO_VOLUME         : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD         : in LOCATOR_DATA_RECORD);

procedure INITIALIZE_LOCATOR
```

```
  (WS                  : in WS_ID;
   DEVICE              : in LOCATOR_DEVICE_NUMBER;
   INITIAL_VIEW_IND    : in VIEW_INDEX;
   INITIAL_POSITION    : in WC.POINT_2;
   ECHO_AREA           : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD         : in LOCATOR_DATA_RECORD);

procedure INITIALIZE_STROKE
  (WS                  : in WS_ID;
   DEVICE              : in STROKE_DEVICE_NUMBER;
   INITIAL_VIEW_IND    : in VIEW_INDEX;
   INITIAL_STROKE      : in WC.POINT_LIST_3;
   ECHO_VOLUME         : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD         : in STROKE_DATA_RECORD);

procedure INITIALIZE_STROKE
  (WS                  : in WS_ID;
   DEVICE              : in STROKE_DEVICE_NUMBER;
   INITIAL_VIEW_IND    : in VIEW_INDEX;
   INITIAL_STROKE      : in WC.POINT_LIST_2;
   ECHO_AREA           : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD         : in STROKE_DATA_RECORD);

procedure INITIALIZE_VALUATOR
  (WS                  : in WS_ID;
   DEVICE              : in VALUATOR_DEVICE_NUMBER;
   INITIAL_VALUE       : in VALUATOR_VALUE;
   ECHO_VOLUME         : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD         : in VALUATOR_DATA_RECORD);

procedure INITIALIZE_VALUATOR
  (WS                  : in WS_ID;
   DEVICE              : in VALUATOR_DEVICE_NUMBER;
   INITIAL_VALUE       : in VALUATOR_VALUE;
   ECHO_AREA           : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD         : in VALUATOR_DATA_RECORD);

procedure INITIALIZE_CHOICE
  (WS                  : in WS_ID;
   DEVICE              : in CHOICE_DEVICE_NUMBER;
   INITIAL_STATUS      : in CHOICE_STATUS;
   INITIAL_CHOICE      : in CHOICE_NUMBER;
   ECHO_VOLUME         : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD         : in CHOICE_DATA_RECORD);

procedure INITIALIZE_CHOICE
  (WS                  : in WS_ID;
   DEVICE              : in CHOICE_DEVICE_NUMBER;
   INITIAL_STATUS      : in CHOICE_STATUS;
   INITIAL_CHOICE      : in CHOICE_NUMBER;
   ECHO_AREA           : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD         : in CHOICE_DATA_RECORD);

procedure INITIALIZE_PICK
  (WS                  : in WS_ID;
   DEVICE              : in PICK_DEVICE_NUMBER;
   INITIAL_STATUS      : in PICK_STATUS;
   INITIAL_PATH        : in PICK_PATH;
   ECHO_VOLUME         : in DC.RECTANGULAR_REGION_3;
```

```
    DATA_RECORD   : in PICK_DATA_RECORD;
    ORDER         : in PATH_ORDER);

procedure INITIALIZE_PICK
   (WS            : in WS_ID;
    DEVICE        : in PICK_DEVICE_NUMBER;
    INITIAL_STATUS : in PICK_STATUS;
    INITIAL_PATH  : in PICK_PATH;
    ECHO_AREA     : in DC.RECTANGULAR_REGION_2;
    DATA_RECORD   : in PICK_DATA_RECORD;
    ORDER         : in PATH_ORDER);

procedure INITIALIZE_STRING
   (WS            : in WS_ID;
    DEVICE        : in STRING_DEVICE_NUMBER;
    INITIAL_STRING : in PHIGS_STRING;
    ECHO_VOLUME   : in DC.RECTANGULAR_REGION_3;
    DATA_RECORD   : in STRING_DATA_RECORD);

procedure INITIALIZE_STRING
   (WS            : in WS_ID;
    DEVICE        : in STRING_DEVICE_NUMBER;
    INITIAL_STRING : in PHIGS_STRING;
    ECHO_AREA     : in DC.RECTANGULAR_REGION_2;
    DATA_RECORD   : in STRING_DATA_RECORD);

procedure SET_LOCATOR_MODE
   (WS      : in WS_ID;
    DEVICE  : in LOCATOR_DEVICE_NUMBER;
    MODE    : in OPERATING_MODE;
    SWITCH  : in ECHO_SWITCH);

procedure SET_STROKE_MODE
   (WS      : in WS_ID;
    DEVICE  : in STROKE_DEVICE_NUMBER;
    MODE    : in OPERATING_MODE;
    SWITCH  : in ECHO_SWITCH);

procedure SET_VALUATOR_MODE
   (WS      : in WS_ID;
    DEVICE  : in VALUATOR_DEVICE_NUMBER;
    MODE    : in OPERATING_MODE;
    SWITCH  : in ECHO_SWITCH);

procedure SET_CHOICE_MODE
   (WS      : in WS_ID;
    DEVICE  : in CHOICE_DEVICE_NUMBER;
    MODE    : in OPERATING_MODE;
    SWITCH  : in ECHO_SWITCH);

procedure SET_PICK_MODE
   (WS      : in WS_ID;
    DEVICE  : in PICK_DEVICE_NUMBER;
    MODE    : in OPERATING_MODE;
    SWITCH  : in ECHO_SWITCH);

procedure SET_STRING_MODE
   (WS      : in WS_ID;
    DEVICE  : in STRING_DEVICE_NUMBER;
```

```
    MODE    : in OPERATING_MODE;
    SWTICH  : in ECHO_SWITCH);

procedure REQUEST_LOCATOR
  (WS          : in  WS_ID;
   DEVICE      : in  LOCATOR_DEVICE_NUMBER;
   STATUS      : out INPUT_STATUS;
   VIEW_IND    : out VIEW_INDEX;
   POSITION    : out WC.POINT_3);

procedure REQUEST_LOCATOR
  (WS          : in  WS_ID;
   DEVICE      : in  LOCATOR_DEVICE_NUMBER;
   STATUS      : out INPUT_STATUS;
   VIEW_IND    : out VIEW_INDEX;
   POSITION    : out WC.POINT_2);

procedure REQUEST_STROKE
  (WS                : in  WS_ID;
   DEVICE            : in  STROKE_DEVICE_NUMBER;
   STATUS            : out INPUT_STATUS;
   VIEW_IND          : out VIEW_INDEX;
   STROKE_POINTS     : out WC.ACCESS_POINT_LIST_3);

procedure REQUEST_STROKE
  (WS                : in  WS_ID;
   DEVICE            : in  STROKE_DEVICE_NUMBER;
   STATUS            : out INPUT_STATUS;
   VIEW_IND          : out VIEW_INDEX;
   STROKE_POINTS     : out WC.ACCESS_POINT_LIST_2);

procedure REQUEST_VALUATOR
  (WS       : in  WS_ID;
   DEVICE   : in  VALUATOR_DEVICE_NUMBER;
   STATUS   : out INPUT_STATUS;
   VALUE    : out VALUATOR_VALUE);

procedure REQUEST_CHOICE
  (WS       : in  WS_ID;
   DEVICE   : in  CHOICE_DEVICE_NUMBER;
   STATUS   : out CHOICE_REQUEST_STATUS;
   CHOICE   : out CHOICE_NUMBER);

procedure REQUEST_PICK
  (WS                  : in  WS_ID;
   DEVICE              : in  PICK_DEVICE_NUMBER;
   DEPTH_TO_RETURN     : in  PATH_DEPTH;
   STATUS              : out PICK_REQUEST_STATUS;
   PATH                : out PICK_PATH);

procedure REQUEST_STRING
  (WS              : in  WS_ID;
   DEVICE          : in  STRING_DEVICE_NUMBER;
   STATUS          : out INPUT_STATUS;
   CHAR_STRING     : out INPUT_STRING);

procedure SAMPLE_LOCATOR
  (WS          : in  WS_ID;
   DEVICE      : in  LOCATOR_DEVICE_NUMBER;
```

```
    VIEW_IND  : out VIEW_INDEX;
    POSITION  : out WC.POINT_3);

procedure SAMPLE_LOCATOR
   (WS        : in  WS_ID;
    DEVICE    : in  LOCATOR_DEVICE_NUMBER;
    VIEW_IND  : out VIEW_INDEX;
    POSITION  : out WC.POINT_2);

procedure SAMPLE_STROKE
   (WS                : in  WS_ID;
    DEVICE            : in  STROKE_DEVICE_NUMBER;
    VIEW_IND          : out VIEW_INDEX;
    STROKE_POINTS     : out WC.ACCESS_POINT_LIST_3);

procedure SAMPLE_STROKE
   (WS                : in  WS_ID;
    DEVICE            : in  STROKE_DEVICE_NUMBER;
    VIEW_IND          : out VIEW_INDEX;
    STROKE_POINTS     : out WC.ACCESS_POINT_LIST_2);

procedure SAMPLE_VALUATOR
   (WS       : in  WS_ID;
    DEVICE   : in  VALUATOR_DEVICE_NUMBER;
    VALUE    : out VALUATOR_VALUE);

procedure SAMPLE_CHOICE
   (WS       : in  WS_ID;
    DEVICE   : in  CHOICE_DEVICE_NUMBER;
    STATUS   : out CHOICE_STATUS;
    CHOICE   : out CHOICE_NUMBER);

procedure SAMPLE_PICK
   (WS                : in  WS_ID;
    DEVICE            : in  PICK_DEVICE_NUMBER;
    DEPTH_TO_RETURN   : in  PATH_DEPTH;
    STATUS            : out PICK_STATUS;
    PATH              : out PICK_PATH);

procedure SAMPLE_STRING
   (WS          : in  WS_ID;
    DEVICE      : in  STRING_DEVICE_NUMBER;
    CHAR_STRING : out INPUT_STRING);

procedure AWAIT_EVENT
   (TIMEOUT : in  DURATION;
    WS      : out WS_ID;
    DEVICE  : out EVENT_DEVICE_NUMBER);

procedure FLUSH_DEVICE_EVENTS
   (WS       : in WS_ID;
    DEVICE   : in EVENT_QUEUE_DEVICE_NUMBER);

procedure GET_LOCATOR
   (VIEW_IND : out VIEW_INDEX;
    POSITION : out WC.POINT_3);

procedure GET_LOCATOR
   (VIEW_IND : out VIEW_INDEX;
```

```
    POSITION   : out WC.POINT_2);

procedure GET_STROKE
   (VIEW_IND        : out VIEW_INDEX;
    STROKE_POINTS   : out WC.ACCESS_POINT_LIST_3);

procedure GET_STROKE
   (VIEW_IND        : out VIEW_INDEX;
    STROKE_POINTS   : out WC.ACCESS_POINT_LIST_2);

procedure GET_VALUATOR
   (VALUE : out VALUATOR_VALUE);

procedure GET_CHOICE
   (STATUS   : out CHOICE_STATUS;
    CHOICE   : out CHOICE_NUMBER);

procedure GET_PICK
   (DEPTH_TO_RETURN : in   PATH_DEPTH;
    STATUS          : out PICK_STATUS;
    PATH            : out PICK_PATH);

procedure GET_STRING
   (CHAR_STRING : out INPUT_STRING);

-- METAFILE FUNCTIONS

procedure WRITE_ITEM_TO_METAFILE
   (WS     : in WS_ID;
    ITEM   : in METAFILE_DATA_RECORD);

procedure GET_ITEM_TYPE_FROM_METAFILE
   (WS               : in  WS_ID;
    TYPE_OF_ITEM     : out METAFILE_ITEM_TYPE;
    LENGTH           : out METAFILE_ITEM_LENGTH);

procedure READ_ITEM_FROM_METAFILE
   (WS               : in  WS_ID;
    MAX_LENGTH       : in  METAFILE_ITEM_LENGTH;
    ITEM             : out METAFILE_DATA_RECORD);

procedure INTERPRET_ITEM
   (ITEM : in METAFILE_DATA_RECORD);

-- INQUIRY FUNCTIONS

procedure INQ_SYSTEM_STATE_VALUE
   (STATE_VALUE : out SYSTEM_STATE);

procedure INQ_WS_STATE_VALUE
   (STATE_VALUE : out WS_STATE);

procedure INQ_STRUCTURE_STATE_VALUE
   (STATE_VALUE : out STRUCTURE_STATE);

procedure INQ_ARCHIVE_STATE_VALUE
   (STATE_VALUE : out ARCHIVE_STATE);

procedure INQ_LIST_OF_AVAILABLE_WS_TYPES
```

```
   (ERROR_INDICATOR : out ERROR_NUMBER;
    LIST_OF_TYPES   : out WS_TYPES.LIST_OF);

procedure INQ_PHIGS_FACILITIES
   (ERROR_INDICATOR                : out ERROR_NUMBER;
    MAX_SIMUL_OPEN_WS              : out PHIGS_POSITIVE;
    MAX_SIMUL_OPEN_ARCHIVES        : out PHIGS_POSITIVE;
    NUMBER_AVAIL_NAMES             : out PHIGS_POSITIVE;
    AVAIL_CHAR_SETS                : out CHAR_SETS.LIST_OF;
    MAX_ISS_NORMAL_FILTER_LIST     : out PHIGS_POSITIVE;
    MAX_ISS_INVERTED_FILTER_LIST   : out PHIGS_POSITIVE);

procedure INQ_GSE_FACILITIES
   (ERROR_INDICATOR          : out ERROR_NUMBER;
    LIST_AVAIL_GSES          : out GSE_IDS.LIST_OF;
    LIST_WS_DEPENDENCIES     : out WS_DEPENDENCIES.LIST_OF);

procedure INQ_MODELLING_CLIPPING_FACILITIES
   (ERROR_INDICATOR          : out ERROR_NUMBER;
    NUMBER_DISTINCT_PLANES   : out PHIGS_POSITIVE;
    LIST_OF_OPERATIONS       : out MODELLING_CLIP_OPERATION_TYPES.LIST_OF);

procedure INQ_EDIT_MODE
   (ERROR_INDICATOR : out ERROR_NUMBER;
    MODE            : out EDIT_MODE);

procedure INQ_SET_OF_OPEN_WS
   (ERROR_INDICATOR : out ERROR_NUMBER;
    OPEN_WS         : out WS_IDS.LIST_OF);

procedure INQ_STRUCTURE_IDENTIFIERS
   (ERROR_INDICATOR     : out ERROR_NUMBER;
    LIST_OF_STRUCTURES  : out STRUCTURE_IDS.LIST_OF);

procedure INQ_ARCHIVE_FILES
   (ERROR_INDICATOR          : out ERROR_NUMBER;
    LIST_ARCHIVE_IDENTIFIERS : out ARCHIVE_IDS.LIST_OF;
    LIST_ARCHIVE_FILES       : out VARIABLE_FILE_IDS.LIST_OF);

procedure INQ_CONFLICT_RESOLUTION
   (ERROR_INDICATOR     : out ERROR_NUMBER;
    ARCHIVAL_CONFLICT   : out CONFLICT_RESOLUTION;
    RETRIEVAL_CONFLICT  : out CONFLICT_RESOLUTION);

procedure INQ_ALL_CONFLICTING_STRUCTURES
   (ARCHIVE_IDENTIFIER  : in  ARCHIVE_ID;
    ERROR_INDICATOR     : out ERROR_NUMBER;
    LIST_OF_STRUCTURES  : out STRUCTURE_IDS.LIST_OF);

procedure INQ_CONFLICTING_STRUCTURES_IN_NETWORK
   (ARCHIVE_IDENTIFIER      : in  ARCHIVE_ID;
    STRUCTURE_IDENTIFIER    : in  STRUCTURE_ID;
    SOURCE                  : in  STRUCTURE_NETWORK_SOURCE;
    ERROR_INDICATOR         : out ERROR_NUMBER;
    LIST_OF_STRUCTURES      : out STRUCTURE_IDS.LIST_OF);

procedure INQ_MORE_SIMULTANEOUS_EVENTS
   (ERROR_INDICATOR : out ERROR_NUMBER;
    EVENTS          : out MORE_EVENTS);
```

```
procedure INQ_WS_CONNECTION_AND_TYPE
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   CONNECTION          : out VARIABLE_CONNECTION_ID;
   TYPE_OF_WS          : out WS_TYPE);

procedure INQ_LIST_OF_VIEW_INDICES
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   DEFINED_VIEW_LIST   : out VIEW_INDICES.LIST_OF);

procedure INQ_VIEW_REPRESENTATION
  (WS                                  : in  WS_ID;
   VIEW_IND                            : in  VIEW_INDEX;
   ERROR_INDICATOR                     : out ERROR_NUMBER;
   UPDATE                              : out UPDATE_STATE;
   REQUESTED_ORIENTATION_MATRIX        : out TRANSFORMATION_MATRIX_3;
   CURRENT_ORIENTATION_MATRIX          : out TRANSFORMATION_MATRIX_3;
   REQUESTED_MAPPING_MATRIX            : out TRANSFORMATION_MATRIX_3;
   CURRENT_MAPPING_MATRIX              : out TRANSFORMATION_MATRIX_3;
   REQUESTED_CLIPPING_LIMITS           : out NPC.RECTANGULAR_REGION_3;
   CURRENT_CLIPPING_LIMITS             : out NPC.RECTANGULAR_REGION_3;
   REQUESTED_XY_CLIPPING               : out CLIPPING_INDICATOR;
   CURRENT_XY_CLIPPING                 : out CLIPPING_INDICATOR;
   REQUESTED_BACK_CLIPPING             : out CLIPPING_INDICATOR;
   CURRENT_BACK_CLIPPING               : out CLIPPING_INDICATOR;
   REQUESTED_FRONT_CLIPPING            : out CLIPPING_INDICATOR;
   CURRENT_FRONT_CLIPPING              : out CLIPPING_INDICATOR);


procedure INQ_HLHSR_MODE
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   UPDATE              : out UPDATE_STATE;
   CURRENT_MODE        : out HLHSR_MODE;
   REQUESTED_MODE      : out HLHSR_MODE);

procedure INQ_POSTED_STRUCTURES
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   LIST_OF_STRUCTURES  : out POSTED_STRUCTURES.LIST_OF);

procedure INQ_DISPLAY_UPDATE_STATE
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   DEFERRAL            : out DEFERRAL_MODE;
   MODIFICATION        : out MODIFICATION_MODE;
   EMPTINESS           : out DISPLAY_SURFACE_EMPTY;
   VISUAL_STATE        : out VISUAL_REPRESENTATION_STATE);

procedure INQ_LIST_OF_POLYLINE_INDICES
  (WS                  : in  WS_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   INDICES             : out POLYLINE_INDICES.LIST_OF);

procedure INQ_POLYLINE_REPRESENTATION
  (WS                  : in  WS_ID;
   POLYLINE_IND        : in  POLYLINE_INDEX;
```

```
    RETURNED_VALUES  : in  RETURN_VALUE_TYPE;
    ERROR_INDICATOR  : out ERROR_NUMBER;
    TYPE_OF_LINE     : out LINETYPE;
    LINEWIDTH_SF     : out LINEWIDTH;
    LINE_COLOUR      : out COLOUR_INDEX);

procedure INQ_LIST_OF_POLYMARKER_INDICES
  (WS                : in  WS_ID;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   INDICES           : out POLYMARKER_INDICES.LIST_OF);

procedure INQ_POLYMARKER_REPRESENTATION
  (WS                : in  WS_ID;
   POLYMARKER_IND    : in  POLYMARKER_INDEX;
   RETURNED_VALUES   : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   TYPE_OF_MARKER    : out MARKER_TYPE;
   SIZE              : out MARKER_SIZE;
   MARKER_COLOUR     : out COLOUR_INDEX);

procedure INQ_LIST_OF_TEXT_INDICES
  (WS                : in  WS_ID;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   INDICES           : out TEXT_INDICES.LIST_OF);

procedure INQ_TEXT_REPRESENTATION
  (WS                : in  WS_ID;
   TEXT_IND          : in  TEXT_INDEX;
   RETURNED_VALUES   : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   FONT              : out TEXT_FONT;
   PRECISION         : out TEXT_PRECISION;
   EXPANSION         : out CHAR_EXPANSION;
   SPACING           : out CHAR_SPACING;
   TEXT_COLOUR       : out COLOUR_INDEX);

procedure INQ_LIST_OF_INTERIOR_INDICES
  (WS                : in  WS_ID;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   INDICES           : out INTERIOR_INDICES.LIST_OF);

procedure INQ_INTERIOR_REPRESENTATION
  (WS                : in  WS_ID;
   INTERIOR_IND      : in  INTERIOR_INDEX;
   RETURNED_VALUES   : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   STYLE_OF_INTERIOR : out INTERIOR_STYLE;
   STYLE_IND         : out STYLE_INDEX;
   INTERIOR_COLOUR   : out COLOUR_INDEX);

procedure INQ_LIST_OF_EDGE_INDICES
  (WS                : in  WS_ID;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   INDICES           : out EDGE_INDICES.LIST_OF);

procedure INQ_EDGE_REPRESENTATION
  (WS                : in  WS_ID;
   EDGE_IND          : in  EDGE_INDEX;
   RETURNED_VALUES   : in  RETURN_VALUE_TYPE;
```

```
    ERROR_INDICATOR   : out ERROR_NUMBER;
    FLAG              : out EDGE_FLAG;
    TYPE_OF_EDGE      : out EDGETYPE;
    EDGEWIDTH_SF      : out EDGEWIDTH;
    EDGE_COLOUR       : out COLOUR_INDEX);

procedure INQ_LIST_OF_PATTERN_INDICES
  (WS                 : in  WS_ID;
    ERROR_INDICATOR   : out ERROR_NUMBER;
    INDICES           : out PATTERN_INDICES.LIST_OF);

procedure INQ_PATTERN_REPRESENTATION
  (WS                 : in  WS_ID;
    PATTERN_IND       : in  PATTERN_INDEX;
    RETURNED_VALUES   : in  RETURN_VALUE_TYPE;
    ERROR_INDICATOR   : out ERROR_NUMBER;
    PATTERN           : out ACCESS_COLOUR_MATRIX);


procedure INQ_COLOUR_MODEL
  (WS                 : in  WS_ID;
    ERROR_INDICATOR   : out ERROR_NUMBER;
    MODEL             : out COLOUR_MODEL);

procedure INQ_LIST_OF_COLOUR_INDICES
  (WS                 : in  WS_ID;
    ERROR_INDICATOR   : out ERROR_NUMBER;
    INDICES           : out COLOUR_INDICES.LIST_OF);

procedure INQ_COLOUR_REPRESENTATION
  (WS                 : in  WS_ID;
    COLOUR_IND        : in  COLOUR_INDEX;
    RETURNED_VALUES   : in  RETURN_VALUE_TYPE;
    ERROR_INDICATOR   : out ERROR_NUMBER;
    COLOUR            : out COLOUR_REPRESENTATION);

procedure INQ_HIGHLIGHTING_FILTER
  (WS                 : in  WS_ID;
    ERROR_INDICATOR   : out ERROR_NUMBER;
    HIGHLIGHTING      : out NAME_SET_FILTER);

procedure INQ_INVISIBILITY_FILTER
  (WS                 : in  WS_ID;
    ERROR_INDICATOR   : out ERROR_NUMBER;
    INVISIBILITY      : out NAME_SET_FILTER);

procedure INQ_WS_TRANSFORMATION
  (WS                           : in  WS_ID;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    UPDATE                      : out UPDATE_STATE;
    REQUESTED_WINDOW_LIMITS     : out NPC.RECTANGULAR_REGION_3;
    CURRENT_WINDOW_LIMITS       : out NPC.RECTANGULAR_REGION_3;
    REQUESTED_VIEWPORT_LIMITS   : out DC.RECTANGULAR_REGION_3;
    CURRENT_VIEWPORT_LIMITS     : out DC.RECTANGULAR_REGION_3);

procedure INQ_WS_TRANSFORMATION
  (WS                           : in  WS_ID;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    UPDATE                      : out UPDATE_STATE;
```

```
    REQUESTED_WINDOW_LIMITS     : out NPC.RECTANGULAR_REGION_2;
    CURRENT_WINDOW_LIMITS       : out NPC.RECTANGULAR_REGION_2;
    REQUESTED_VIEWPORT_LIMITS   : out DC.RECTANGULAR_REGION_2;
    CURRENT_VIEWPORT_LIMITS     : out DC.RECTANGULAR_REGION_2);

procedure INQ_LOCATOR_DEVICE_STATE
  (WS                           : in  WS_ID;
   DEVICE                       : in  LOCATOR_DEVICE_NUMBER;
   RETURNED_VALUES              : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR              : out ERROR_NUMBER;
   MODE                         : out OPERATING_MODE;
   SWITCH                       : out ECHO_SWITCH;
   INITIAL_VIEW_IND             : out VIEW_INDEX;
   INITIAL_POSITION             : out WC.POINT_3;
   ECHO_VOLUME                  : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD                  : out LOCATOR_DATA_RECORD);

procedure INQ_LOCATOR_DEVICE_STATE
  (WS                           : in  WS_ID;
   DEVICE                       : in  LOCATOR_DEVICE_NUMBER;
   RETURNED_VALUES              : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR              : out ERROR_NUMBER;
   MODE                         : out OPERATING_MODE;
   SWITCH                       : out ECHO_SWITCH;
   INITIAL_VIEW_IND             : out VIEW_INDEX;
   INITIAL_POSITION             : out WC.POINT_2;
   ECHO_AREA                    : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD                  : out LOCATOR_DATA_RECORD);

procedure INQ_STROKE_DEVICE_STATE
  (WS                           : in  WS_ID;
   DEVICE                       : in  STROKE_DEVICE_NUMBER;
   RETURNED_VALUES              : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR              : out ERROR_NUMBER;
   MODE                         : out OPERATING_MODE;
   SWITCH                       : out ECHO_SWITCH;
   INITIAL_VIEW_IND             : out VIEW_INDEX;
   INITIAL_STROKE_POINTS        : out WC.ACCESS_POINT_LIST_3;
   ECHO_VOLUME                  : out DC.RECTANGULAR_REGION_3;
   DATA_RECORD                  : out STROKE_DATA_RECORD);

procedure INQ_STROKE_DEVICE_STATE
  (WS                           : in  WS_ID;
   DEVICE                       : in  STROKE_DEVICE_NUMBER;
   RETURNED_VALUES              : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR              : out ERROR_NUMBER;
   MODE                         : out OPERATING_MODE;
   SWITCH                       : out ECHO_SWITCH;
   INITIAL_VIEW_IND             : out VIEW_INDEX;
   INITIAL_STROKE_POINTS        : out WC.ACCESS_POINT_LIST_2;
   ECHO_AREA                    : out DC.RECTANGULAR_REGION_2;
   DATA_RECORD                  : out STROKE_DATA_RECORD);

procedure INQ_VALUATOR_DEVICE_STATE
  (WS                           : in  WS_ID;
   DEVICE                       : in  VALUATOR_DEVICE_NUMBER;
   ERROR_INDICATOR              : out ERROR_NUMBER;
   MODE                         : out OPERATING_MODE;
   SWITCH                       : out ECHO_SWITCH;
```

```
    INITIAL_VALUE       : out VALUATOR_VALUE;
    ECHO_VOLUME         : out DC.RECTANGULAR_REGION_3;
    DATA_RECORD         : out VALUATOR_DATA_RECORD);

procedure INQ_VALUATOR_DEVICE_STATE
   (WS                  : in  WS_ID;
    DEVICE              : in  VALUATOR_DEVICE_NUMBER;
    ERROR_INDICATOR     : out ERROR_NUMBER;
    MODE                : out OPERATING_MODE;
    SWITCH              : out ECHO_SWITCH;
    INITIAL_VALUE       : out VALUATOR_VALUE;
    ECHO_AREA           : out DC.RECTANGULAR_REGION_2;
    DATA_RECORD         : out VALUATOR_DATA_RECORD);

procedure INQ_CHOICE_DEVICE_STATE
   (WS                  : in  WS_ID;
    DEVICE              : in  CHOICE_DEVICE_NUMBER;
    ERROR_INDICATOR     : out ERROR_NUMBER;
    MODE                : out OPERATING_MODE;
    SWITCH              : out ECHO_SWITCH;
    INITIAL_STATUS      : out CHOICE_STATUS;
    INITIAL_CHOICE      : out CHOICE_NUMBER;
    ECHO_VOLUME         : out DC.RECTANGULAR_REGION_3;
    DATA_RECORD         : out CHOICE_DATA_RECORD);

procedure INQ_CHOICE_DEVICE_STATE
   (WS                  : in  WS_ID;
    DEVICE              : in  CHOICE_DEVICE_NUMBER;
    ERROR_INDICATOR     : out ERROR_NUMBER;
    MODE                : out OPERATING_MODE;
    SWITCH              : out ECHO_SWITCH;
    INITIAL_STATUS      : out CHOICE_STATUS;
    INITIAL_CHOICE      : out CHOICE_NUMBER;
    ECHO_AREA           : out DC.RECTANGULAR_REGION_2;
    DATA_RECORD         : out CHOICE_DATA_RECORD);

procedure INQ_PICK_DEVICE_STATE
   (WS                  : in  WS_ID;
    DEVICE              : in  PICK_DEVICE_NUMBER;
    RETURNED_VALUES     : in  RETURN_VALUE_TYPE;
    ERROR_INDICATOR     : out ERROR_NUMBER;
    MODE                : out OPERATING_MODE;
    SWITCH              : out ECHO_SWITCH;
    PICKABILITY         : out NAME_SET_FILTER;
    INITIAL_STATUS      : out PICK_STATUS;
    INITIAL_PATH        : out PICK_PATH;
    ECHO_VOLUME         : out DC.RECTANGULAR_REGION_3;
    DATA_RECORD         : out PICK_DATA_RECORD;
    ORDER               : out PATH_ORDER);

procedure INQ_PICK_DEVICE_STATE
   (WS                  : in  WS_ID;
    DEVICE              : in  PICK_DEVICE_NUMBER;
    RETURNED_VALUES     : in  RETURN_VALUE_TYPE;
    ERROR_INDICATOR     : out ERROR_NUMBER;
    MODE                : out OPERATING_MODE;
    SWITCH              : out ECHO_SWITCH;
    PICKABILITY         : out NAME_SET_FILTER;
    INITIAL_STATUS      : out PICK_STATUS;
```

```
    INITIAL_PATH          : out PICK_PATH;
    ECHO_AREA             : out DC.RECTANGULAR_REGION_2;
    DATA_RECORD           : out PICK_DATA_RECORD;
    ORDER                 : out PATH_ORDER);

procedure INQ_STRING_DEVICE_STATE
   (WS                    : in  WS_ID;
    DEVICE                : in  STRING_DEVICE_NUMBER;
    ERROR_INDICATOR       : out ERROR_NUMBER;
    MODE                  : out OPERATING_MODE;
    SWITCH                : out ECHO_SWITCH;
    INITIAL_CHAR_STRING   : out INPUT_STRING;
    ECHO_VOLUME           : out DC.RECTANGULAR_REGION_3;
    DATA_RECORD           : out STRING_DATA_RECORD);

procedure INQ_STRING_DEVICE_STATE
   (WS                    : in  WS_ID;
    DEVICE                : in  STRING_DEVICE_NUMBER;
    ERROR_INDICATOR       : out ERROR_NUMBER;
    MODE                  : out OPERATING_MODE;
    SWITCH                : out ECHO_SWITCH;
    INITIAL_CHAR_STRING   : out INPUT_STRING;
    ECHO_AREA             : out DC.RECTANGULAR_REGION_2;
    DATA_RECORD           : out STRING_DATA_RECORD);

procedure INQ_WS_CATEGORY
   (TYPE_OF_WS            : in  WS_TYPE;
    ERROR_INDICATOR       : out ERROR_NUMBER;
    CATEGORY              : out WS_CATEGORY);

procedure INQ_DISPLAY_SPACE_SIZE
   (TYPE_OF_WS            : in  WS_TYPE;
    ERROR_INDICATOR       : out ERROR_NUMBER;
    UNITS                 : out DC_UNITS;
    MAX_DC_UNIT_SIZE      : out DC.SIZE_3;
    MAX_RASTER_UNIT_SIZE  : out RASTER_UNIT_SIZE_3);

procedure INQ_DISPLAY_SPACE_SIZE
   (TYPE_OF_WS            : in  WS_TYPE;
    ERROR_INDICATOR       : out ERROR_NUMBER;
    UNITS                 : out DC_UNITS;
    MAX_DC_UNIT_SIZE      : out DC.SIZE_2;
    MAX_RASTER_UNIT_SIZE  : out RASTER_UNIT_SIZE_2);

procedure INQ_HLHSR_FACILITIES
   (TYPE_OF_WS            : in  WS_TYPE;
    ERROR_INDICATOR       : out ERROR_NUMBER;
    LIST_OF_IDENTIFIERS   : out HLHSR_IDS.LIST_OF;
    LIST_OF_MODES         : out HLHSR_MODES.LIST_OF);

procedure INQ_VIEW_FACILITIES
   (TYPE_OF_WS            : in  WS_TYPE;
    ERROR_INDICATOR       : out ERROR_NUMBER;
    NUMBER_OF_INDICES     : out PHIGS_POSITIVE);

procedure INQ_PREDEFINED_VIEW_REPRESENTATION
   (TYPE_OF_WS            : in  WS_TYPE;
    VIEW_IND              : in  VIEW_INDEX;
    ERROR_INDICATOR       : out ERROR_NUMBER;
```

```
    ORIENTATION_MATRIX  : out TRANSFORMATION_MATRIX_3;
    MAPPING_MATRIX      : out TRANSFORMATION_MATRIX_3;
    CLIPPING_LIMITS     : out NPC.RECTANGULAR_REGION_3;
    XY_CLIPPING         : out CLIPPING_INDICATOR;
    BACK_CLIPPING       : out CLIPPING_INDICATOR;
    FRONT_CLIPPING      : out CLIPPING_INDICATOR);

procedure INQ_WS_CLASSIFICATION
  (TYPE_OF_WS           : in  WS_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   CLASS                : out DISPLAY_CLASS);

procedure INQ_DYNAMICS_OF_WS_ATTRIBUTES
  (TYPE_OF_WS                       : in  WS_TYPE;
   ERROR_INDICATOR                  : out ERROR_NUMBER;
   POLYLINE_REPRESENTATION          : out DYNAMIC_MODIFICATION;
   POLYMARKER_REPRESENTATION        : out DYNAMIC_MODIFICATION;
   TEXT_REPRESENTATION              : out DYNAMIC_MODIFICATION;
   INTERIOR_REPRESENTATION          : out DYNAMIC_MODIFICATION;
   EDGE_REPRESENTATION              : out DYNAMIC_MODIFICATION;
   PATTERN_REPRESENTATION           : out DYNAMIC_MODIFICATION;
   COLOUR_REPRESENTATION            : out DYNAMIC_MODIFICATION;
   VIEW_REPRESENTATION              : out DYNAMIC_MODIFICATION;
   WS_TRANSFORMATION                : out DYNAMIC_MODIFICATION;
   HIGHLIGHTING_FILTER              : out DYNAMIC_MODIFICATION;
   INVISIBILITY_FILTER              : out DYNAMIC_MODIFICATION;
   HLHSR_MODE_CHANGE                : out DYNAMIC_MODIFICATION);

procedure INQ_DEFAULT_DISPLAY_UPDATE_STATE
  (TYPE_OF_WS           : in  WS_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   DEFERRAL             : out DEFERRAL_MODE;
   MODIFICATION         : out MODIFICATION_MODE);

procedure INQ_POLYLINE_FACILITIES
  (TYPE_OF_WS           : in  WS_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   NUMBER_OF_TYPES      : out PHIGS_INTEGER;
   LIST_OF_TYPES        : out LINETYPES.LIST_OF;
   NUMBER_OF_WIDTHS     : out PHIGS_NATURAL;
   NOMINAL_WIDTH        : out DC.MAGNITUDE;
   RANGE_OF_WIDTHS      : out DC.RANGE_OF_MAGNITUDES;
   NUMBER_OF_INDICES    : out PHIGS_NATURAL);

procedure INQ_PREDEFINED_POLYLINE_REPRESENTATION
  (TYPE_OF_WS           : in  WS_TYPE;
   POLYLINE_IND         : in  POLYLINE_INDEX;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   TYPE_OF_LINE         : out LINETYPE;
   LINEWIDTH_SF         : out LINEWIDTH;
   LINE_COLOUR          : out COLOUR_INDEX);

procedure INQ_POLYMARKER_FACILITIES
  (TYPE_OF_WS           : in  WS_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   NUMBER_OF_TYPES      : out PHIGS_INTEGER;
   LIST_OF_TYPES        : out MARKER_TYPES.LIST_OF;
   NUMBER_OF_SIZES      : out PHIGS_NATURAL;
   NOMINAL_SIZE         : out DC.MAGNITUDE;
```

```
    RANGE_OF_SIZES       : out DC.RANGE_OF_MAGNITUDES;
    NUMBER_OF_INDICES    : out PHIGS_NATURAL);

procedure INQ_PREDEFINED_POLYMARKER_REPRESENTATION
  (TYPE_OF_WS            : in  WS_TYPE;
   POLYMARKER_IND        : in  POLYMARKER_INDEX;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   TYPE_OF_MARKER        : out MARKER_TYPE;
   SIZE                  : out MARKER_SIZE;
   MARKER_COLOUR         : out COLOUR_INDEX);

procedure INQ_TEXT_FACILITIES
  (TYPE_OF_WS                      : in  WS_TYPE;
   ERROR_INDICATOR                 : out ERROR_NUMBER;
   LIST_OF_FONT_PRECISION_PAIRS    : out TEXT_FONT_PRECISIONS.LIST_OF;
   NUMBER_OF_HEIGHTS               : out PHIGS_NATURAL;
   RANGE_OF_HEIGHTS                : out DC.RANGE_OF_MAGNITUDES;
   NUMBER_OF_EXPANSIONS            : out PHIGS_NATURAL;
   RANGE_OF_EXPANSIONS             : out DC.RANGE_OF_MAGNITUDES;
   NUMBER_OF_INDICES               : out PHIGS_NATURAL);

procedure INQ_PREDEFINED_TEXT_REPRESENTATION
  (TYPE_OF_WS            : in  WS_TYPE;
   TEXT_IND              : in  TEXT_INDEX;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   FONT                  : out TEXT_FONT;
   PRECISION             : out TEXT_PRECISION;
   EXPANSION             : out CHAR_EXPANSION;
   SPACING               : out CHAR_SPACING;
   TEXT_COLOUR           : out COLOUR_INDEX);

procedure INQ_ANNOTATION_FACILITIES
  (TYPE_OF_WS                      : in  WS_TYPE;
   ERROR_INDICATOR                 : out ERROR_NUMBER;
   LIST_OF_ANNOTATION_STYLES       : out ANNOTATION_STYLES.LIST_OF;
   NUMBER_OF_CHAR_HEIGHTS          : out PHIGS_NATURAL;
   RANGE_OF_CHAR_HEIGHTS           : out DC.RANGE_OF_MAGNITUDES);

procedure INQ_TEXT_EXTENT
  (TYPE_OF_WS            : in  WS_ID;
   FONT                  : in  TEXT_FONT;
   EXPANSION             : in  CHAR_EXPANSION;
   SPACING               : in  CHAR_SPACING;
   HEIGHT                : in  MC.MAGNITUDE;
   PATH                  : in  TEXT_PATH;
   ALIGNMENT             : in  TEXT_ALIGNMENT;
   CHAR_STRING           : in  PHIGS_STRING;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   EXTENT_RECTANGLE      : out MC.RECTANGULAR_REGION_2;
   CONCATENATION_OFFSET  : out MC.VECTOR_2);

-- This procedure using MC coordinates is for use with text primitives.


procedure INQ_TEXT_EXTENT
  (TYPE_OF_WS            : in  WS_ID;
   FONT                  : in  TEXT_FONT;
   EXPANSION             : in  CHAR_EXPANSION;
   SPACING               : in  CHAR_SPACING;
```