INTERNATIONAL STANDARD

ISO/IEC 9638-3

First edition
1994-12-15

Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Language bindings —

Part 3:
Ada

Technologies de l'information — Infographie — Techniques interfaciales pour dialogues avec dispositifs graphiques (CGI) — Liants de langage —

Partie 3: Ada

Reference number
ISO/IEC 9638-3:1994(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. 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 voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 9638-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee 24, *Computer graphics and image processing*.

ISO/IEC 9638 consists of the following parts, under the general title *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Language bindings*:

— *Part 1: FORTRAN*

— *Part 2: PASCAL*

— *Part 3: Ada*

Annexes A, B, C, D and E of this part of ISO/IEC 9638 are for information only.

Introduction

The Computer Graphics Interface (CGI) (ISO/IEC 9636) 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 document is to define a standard binding of CGI to the Ada computer programming language.

This page intentionally left blank

Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Language bindings —

Part 3:
Ada

1 Scope

The Computer Graphics Interface (CGI) (ISO/IEC 9636), specifies a language independent standard interface between device-independent and device-dependent parts of a graphics system. For integration into a programming language, CGI is embedded in a language dependent layer obeying the particular conventions of that language. This part of ISO/IEC 9638 specifies such a language dependent layer for the Ada programming language.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9638. 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 9638 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.*

ISO/IEC 9636-1:1991, *Information technology - Computer graphics - Interfacing techniques for dialogues with graphics devices (CGI) - Functional Specification - Part 1: Overview, profiles, and conformance.*

ISO/IEC 9636-2:1991, *Information technology - Computer graphics - Interfacing techniques for dialogues with graphics devices (CGI) - Functional Specification - Part 2: Control.*

ISO/IEC 9636-3:1991, *Information technology - Computer graphics - Interfacing techniques for dialogues with graphics devices (CGI) - Functional Specification - Part 3: Output.*

ISO/IEC 9636-4:1991, *Information technology - Computer graphics - Interfacing techniques for dialogues with graphics devices (CGI) - Functional Specification - Part 4: Segments.*

ISO/IEC 9636-5:1991, *Information technology - Computer graphics - Interfacing techniques for dialogues with graphics devices (CGI) - Functional Specification - Part 5: Input and echoing.*

ISO/IEC 9636-6:1991, *Information technology - Computer graphics - Interfacing techniques for dialogues with graphics devices (CGI) - Functional Specification - Part 6: Raster.*

ISO/IEC 9637-1:1992, *Information technology - Computer graphics - Interfacing techniques for dialogues with graphics devices(CGI) - Data stream binding - Part 1: Character encoding.*

ISO/IEC 9637-2:1992, *Information technology - Computer graphics - Interfacing techniques for dialogues with graphics devices(CGI) - Data stream binding - Part 2: Binary encoding.*

3 Principles

This binding supports the implementation independent aspects of the Ada standard except as discussed under multi-tasking. This binding does not assume that the compiler supports any Ada language features which are implementation dependent, but implies that the compiler must be able to support the declarations contained in this CGI/Ada binding.

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

3.1 Conformance

This binding incorporates the rules of conformance defined in the ISO/IEC 9636 for CGI implementations with these additional requirements specifically defined for Ada implementations of CGI.

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 the CGI standard as modified or extended for Ada as stated in this binding document.

- The package(s) corresponding to CGI shall be an available Ada library unit, with all names as specified by this document or as modified for one or more CGI profiles.

3.2 Implications of the language

3.2.1 Functional mapping

The functions which constitute the ISO/IEC 9636 are each mapped to Ada procedures within this language binding. This mapping utilizes a one-to-one correlation between the CGI functions embodied in the CGI standard and the Ada procedures herein, with the exception of the generic functional definitions within the CGI standard. In the case of these generic functional definitions, multiple Ada procedures have been utilized to attain the functional mapping (binding). The following list denotes all such functions and their Ada binding complements:

ISO/IEC 9636 Function : Put Current <input class> Measure

is bound to - Put Current Locator Measure
Put Current Stroke Measure
Put Current Valuator Measure
Put Current Choice Measure
Put Current Pick Measure
Put Current String Measure
Put Current Raster Measure
Put Current General Measure

ISO/IEC 9636 Function : <input class> Device Data

is bound to - Set Locator Device Data
Set Stroke Device Data
Set Valuator Device Data
Set Choice Device Data
Set Pick Device Data
Set String Device Data
Set Raster Device Data
Set General Device Data

ISO/IEC 9636 Function : Request <input class>

is bound to - Request Locator
Request Stroke
Request Valuator
Request Choice
Request Pick
Request String
Request Raster
Request General

ISO/IEC 9636 Function : Sample <input class>

is bound to - Sample Locator
Sample Stroke
Sample Valuator
Sample Choice
Sample Pick
Sample String
Sample Raster
Sample General

ISO/IEC 9636 Function : Echo Request <input class>

is bound to - Echo Request Locator
Echo Request Stroke
Echo Request Valuator
Echo Request Choice
Echo Request Pick
Echo Request String
Echo Request Raster
Echo Request General

ISO/IEC 9636 Function : Dequeue <input class> Event

is bound to - Dequeue Locator Event
Dequeue Stroke Event
Dequeue Valuator Event
Dequeue Choice Event
Dequeue Pick Event
Dequeue String Event

Dequeue Raster Event
Dequeue General Event

ISO/IEC 9636 Function : Update <input class> Echo Output

is bound to - Update Locator Echo Output
Update Stroke Echo Output
Update Valuator Echo Output
Update Choice Echo Output
Update Pick Echo Output
Update String Echo Output
Update Raster Echo Output
Update General Echo Output

3.2.2 Implementation and host dependencies

There are a number of implementation and host dependent issues which will be associated with an Ada compiler and its run-time environment. These issues will affect the portability of application (client/generator, driver, target, ...) programs utilizing this binding of CGI. The client programmer should follow accepted practices for ensuring portability of Ada programs to avoid introducing problems when rehosting an application of CGI to another system. This binding attempts to avoid dependencies on compiler specific Ada types which may vary from machine to machine.

Since CGI provides for variable precisions which may be specified by the client, the situation could exist in which an 8-bit, 16-bit, 24-bit, or 32-bit machine will meet all of the required needs for a particular CGI client as long as the client stays within the ranges provided by the host machine. Wherever possible, universal integer type definitions have been applied in this part of ISO/IEC 9638 in order to support the variable precisions required by the CGI client. These universal types are specified via minimum and maximum values which are contained in the CGI configuration package. Therefore a conforming implementation/application of CGI may change these value range limits to meet its particular needs. Some additional range limits were added in the CGI configuration package to handle the mapping of the Fixed Integer data type. Since the ISO/IEC 9636-1 provides a fixed integer (IF) data type which can either be a fixed 8, 16, or 32 bit integer, it was decided to map each subtype of the fixed integer with its own range value limits. The fixed integer definitions which were mapped in this way were the Device Coordinate, Error Report, Intrinsic Name, and Input Surface Coordinate types. Through this mapping, the fixed integer type definitions may be changed easily by the client at compile time of the CGI. For an implementation of CGI which generates a data stream, it is possible that the precision specified at the language binding layer may differ from the precision specified for data stream transmission. If a particular implementation or client application of CGI can not support all of the fixed integer type definitions, the non-supported fixed integer definitions should be redefined or removed. If such a situation occurs and a precision is selected by a client which the implementation can not support, the CGI 3:204 error, specified precision requirement not achievable shall be logged to the error queue.

3.2.3 Error handling

CGI provides an error queue implementation which can be inquired about by a client. Through this mechanism, CGI errors can be dequeued by the client and error handling controls can be put in effect by the client. In certain configurations of CGI, this error handling mechanism will not be able to account for errors which occur through the improper execution of this language binding. Cases like this include, but are not limited to Generator/Interpreter implementations or any implementation where the language binding does not co-exist with the graphics interpreter. For these situations an implementation of CGI may choose to define an

error handler in order to detect, record, and report errors which occur as a result of Ada language binding violations. It is the intent of this language binding that all of the Ada predefined exceptions (NUMERIC_ERROR, PROGRAM_ERROR, STORAGE_ERROR, CONSTRAINT_ERROR, and TASKING_ERROR) be handled without causing interruption to client performance. This is in accordance with the ISO/IEC 9636 error philosophy which states that the CGI does not automatically report errors to the client program. Detection of any of these predefined Ada exceptions during the implementation's execution of a function, will result in the generation of the appropriate binding specific error (if allowed by the error controls in effect).

This part of ISO/IEC 9638 defines an error handling scheme for the language binding which is implemented in much the same fashion as the interpreter error handling mechanism defined in the ISO/IEC 9636. A language binding error queue and a set of error handling controls which are modifiable via the CGI function, SET_ERROR_HANDLING_CONTROLS may be provided in order to give the client greater visibility into the performance of the language binding implementation. This is knowledge which the client may find beneficial either in integrating to or in determining the limitations of a CGI implementation or system. Any language binding error functionality shall always default to the default states defined in the ISO/IEC 9636 upon system startup or upon the receipt of an INITIALIZE_SESSION command. While in the disabled state no binding errors will be queued. All error reporting shall remain under client control. If such an error implementation is provided by a language binding, it shall be explicitly documented. An implementation of CGI may choose whether or not to provide additional error detection facilities, but in either case shall provide the inquire error handling support function in the error handling utilities package. Subclause 7.6 will contain more information on the specific binding and implementation of the error functions.

It is also necessary for the Ada language binding to include a file of defined error constants for shared use between the client and the language binding. This file will contain all of the CGI errors defined in the ISO/IEC 9636 (for client interpretation) as well as any implementation dependent errors which have been defined by the client or the implementation (all negative). All implementation defined errors should be explicitly documented by the CGI implementation. The language binding error constants file shall be called CGI_ERROR_CONST and shall be encoded as described in Annex A. The language binding error constants file will also contain the predefined language binding errors defined in this standard. All predefined Ada language binding errors will be of the appropriate CGI error class (according to the guidelines in the ISO/IEC 9636-1). The Ada language binding error identifiers defined within this part of ISO/IEC 9638 will be numbered starting with error identifier 2500. The language binding specific errors listed in order of error class are defined as follows:

Error Identifier: 3:2500
Cause: Content of Packed Data Array Invalid
Reaction: Function ignored.

Error Identifier: 5:2500
Cause: Data Continuation Expected, Not Received
Reaction: Function ignored.

Error Identifier: 6:2500
Cause: Ada Constraint Error
Reaction: Function ignored.

Error Identifier: 6:2501
Cause: Ada Numeric Error
Reaction: Function ignored.

Error Identifier: 6:2502
Cause: Ada Program Error
Reaction: Function ignored.

Error Identifier: 6:2503
Cause: Ada Storage Error
Reaction: Function ignored.

Error Identifier: 6:2504
Cause: Ada Tasking Error
Reaction: Function ignored.

Error Identifier: 6:2505
Cause: Client Out of Memory
Reaction: Function ignored.

3.2.4 Continuation of functions

Certain CGI functions have been bound such that the data associated with the single, conceptual CGI function can be partitioned. In general these CGI functions handle data which may be of indeterminate length and may be very large. The intent is to allow clients of the procedural binding to pass large quantities of data to a CGI implementation without having to buffer all of the data at one time in the client's memory.

The functions which have been bound in this manner are:

Cell Array
Generalized Drawing Primitive
Pixel Array
Polygon
Polyline

The continuation of these functions has been provided for through the addition of a continuation flag. The continuation flag will appear as the last input parameter in the function's parameter list. The continuation parameter is a FINAL_FLAG_TYPE enumerated value and will indicate to an implementation the state of the input data. If the flag is set to FINAL, the input data passed in with the function is the last or only data partition. If the flag is set to NOT_FINAL, it is expected that the client will continue to call this function providing input data partitions which are to be appended in secession until a call of the function is processed with the continuation flag set to FINAL. After the first partition of a continuation function is received, each successive call to the function shall only contain valid information for the portion of the function which is being extended as follows:

Cell Array	Cell Colour Specifiers
Generalized Drawing Primitive	Point List
Pixel Array	Colour Specifiers
Polygon	Point List
Polyline	Point List

In this binding, the continuation flag shall be implemented as a defaulted parameter with a default value of FINAL. With this parameter defaulted to FINAL, a client may execute any of the continuation functions without specifying the value of the continuation flag and achieve the effect of the function exactly as it is described in

the ISO/IEC 9636. When a function is processed by an implementation with a value of NOT_FINAL, failure to invoke the same function with further continuations or a final data partition will result in the class 5 binding specific error, Data Continuation Expected, Not Received. The reaction to this error will be to ignore the function causing the error. If the execution of a function causes an error while the continuation flag is set to a value of NOT FINAL, the implementation must honour the state of the continuation flag and discontinue the execution of the function. This means accepting data partitions for the function until a partition is received with a continuation flag value of FINAL (even though further data partitions will have no effect). This is done in order to preserve the client's state of execution and to reduce side affects.

3.2.5 Packed data formats

The following CGI functions have been bound such that the data associated with the single, conceptual CGI function can be packed into a concise format.

Cell Array
Pixel Array
Colour Array

The reason that these functions have been provided for in this manner is due to the fact that CGI defines local colour precisions which govern the transfer of colour data between a client and a graphics device. Since graphical devices vary in the precision with which colour data can be specified, this local colour precision may be inquired of a device and then used in the encoding of the colour data within a data stream. It may also be the case that a client will only need a subset of index values to describe a particular image, but overall will desire some greater precision for colour data. In this case, it is to the client's advantage to pack this colour data not only for transmission across a data stream but for image storage in client resident memory as well. The packed versions of these functions are defined in subclause 7.6. The packing methods defined by this language binding coincide with the packing methods supported by the various CGI data stream encodings. There is also an implementation private encoding which is for use with either an Application Programmer Interface (API) implementation of CGI or for a data stream generator which can pack the data in a unique format. An implementation may choose to support none, all, or a selected subset of packing methods but shall support the inquire available packing methods function regardless. This function will return to the client the number of methods supported and a list of those methods. If the returned number of methods supported is 0, then no packing support is available from the implementation. The implementation is also responsible for storing information about the packed data and the packing method used to pack the packed data. Under this methodology, a client may pass any packed array to an implementation and expect the implementation to handle the data without any prior or further client intervention. Invoking a packing utility with data which can not be deciphered by the implementation will result in a class 3 binding specific error, Content Of Packed Data Array Invalid. The reaction will be to ignore the function causing the error.

3.2.6 Events and event report lists

The CGI event type and the CGI event report list are defined in the ISO/IEC 9636 as data record types. Both of these types are unique to the CGI event queue transfer function. The event queue transfer function will return to a client all of the event information stored in the event queue by way of an event report list. The client has no way of knowing the size of this list or the number of events contained in the event queue of the graphics device. This language binding has mapped the event report list type to a pointer (Ada access type) to a data record. This is due to the variable size of the event queue. With this implementation, the generator will size the input data and pass the data record pointer for the client's use. If not enough memory is available to the implementation for the allocation of the event report list, a class 6 binding specific error (Client Out of Memory) will be generated (if allowed by the error controls in effect). Once the client has received the event report list, the data

record utility package may be called to extract the individual events. When the client has extracted all of the information from the event report list, the list may be deallocated via the deallocate event report list function in the CGI data record utilities package.

The CGI event type in the ISO/IEC 9636 is defined as a data record of returned input data for which the content is dependent upon the event's input class as follows:

```
Input Class          E
LID Index            IX
Trigger              IX
Timestamp            R
Measure Validity     E
Measure Value:
  if input class = LOCATOR:
    position(P)
  if input class = STROKE:
    list of points (in stroke)(nP)
  if input class = VALUATOR:
    value (R)
  if input class = CHOICE:
    choice number (I)
  if input class = PICK:
    list of pick values (nPV)
   if input class = STRING:
    string (S)
  if input class = RASTER:
    xcount, ycount (2I)
    list of input colour values (nICO)
  if input class = GENERAL:
    data record (D)
```

Since the internal structure of an event is known, this language binding has defined an event type as an Ada variant record which varies upon the input class. Since large amounts of data may be returned in an event for stroke, pick, string, raster, and general measures, these variant portions will return pointers (Ada accesses) to the measure data. In this way, the client may declare event types without having to worry about memory sizing constraints. An event type will be returned to the client via a call to the remove event function in the CGI data record utilities package. If not enough memory is available to allocate the event, a class 6 binding specific error (Client Out of Memory) will be generated (if allowed by the error controls in effect). When the client has extracted all of the information from the event, the event may be deallocated via the deallocate event function in the CGI data record utilities package.

3.2.7 Data mapping

The basic and abstract data types of CGI 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 CGI data types and the Ada binding data types is summarized below:

The CGI attribute set name type (ASN) is mapped to the Ada integer type.

The CGI bitmap identifier type (BN) is mapped to the Ada integer type.

The CGI direct colour value type (CD) is mapped to an Ada record type.

The CGI colour index type (CI) is mapped to an Ada integer type.

The CGI colour specifier type (CO) is mapped to an Ada variant record type.

The CGI character set type (CS) is mapped to an Ada record type.

The CGI client specified name type (CSN) is mapped to an Ada integer type.

The CGI data record type (D) is mapped to an Ada private type.

The CGI device coordinate type (DC) is mapped to an Ada integer type.

The CGI device point type (DP) is mapped to an Ada record type.

The CGI Enumerated data types (E) are mapped to Ada enumerated data types.

The CGI error identifier type (EI) is mapped to an Ada record type.

The CGI error report type (ER) is mapped to an Ada variant record type.

The CGI event type (EV) is mapped to an Ada variant record type.

The CGI event reports list type (EVL) is mapped to an Ada access type.

The CGI function identifier type (FN) is mapped to an Ada integer type.

The CGI integer type (I) is mapped to an Ada integer type.

The CGI input colour specifier type (ICO) is mapped to an Ada variant record type.

The CGI fixed 8-bit integer type (IF8) is mapped to an Ada integer type.

The CGI fixed 16-bit integer type (IF16) is mapped to an Ada integer type.

The CGI fixed 32-bit integer type (IF32) is mapped to an Ada integer type.

The CGI fixed integer type (IF) has not been mapped directly by this binding document. Instead, all occurrences of the IF data type have been substituted with more exact integer type definitions which are constrained where applicable within the CGI configuration package.

The CGI intrinsic name type (IN) is mapped to an Ada integer type.

The CGI input surface coordinate type (ISC) is mapped to an Ada integer type.

The CGI input surface point type (ISP) is mapped to an Ada record type.

The CGI index type (IX) is mapped to an Ada integer type.

The CGI VDC point type (P) is mapped to an Ada record type.

The CGI pick identifier type (PN) is mapped to an Ada integer type.

The CGI profile identifier type (PRN) is mapped to an Ada integer type.

The CGI pick value type (PV) is mapped to an Ada record type.

The CGI real type (R) is mapped to an Ada variant record type.

The CGI string type (S) is mapped to an Ada discriminated record type.

The CGI fixed string type (SF) is mapped to an Ada discriminated record type.

The CGI segment identifier type (SN) is mapped to an Ada integer type.

The CGI size specification type (SS) is mapped to an Ada variant record type.

The CGI viewport coordinate type (VC) is mapped to an Ada variant record type.

The CGI virtual device coordinate type (VDC) is mapped to an Ada variant record type.

The CGI viewport point type (VP) is mapped to an Ada record type.

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

Annex E contains guidelines for implementors who want to support multi-tasking client programs.

3.2.9 Packaging

In binding the CGI to the Ada programming language it is most important to provide a packaging structure which will ensure reusability, portability, and maintainability. It is the goal of this part of ISO/IEC 9638 to satisfy these requirements with the best possible packaging structure while still paying attention to the needs of the client programmer.

The functions provided within the ISO/IEC 9636 may be subdivided into various profiles as denoted in the ISO/IEC 9636-1. Although profiles are intended to define the level of support provided by a CGI graphics device, it is possible that various generator implementations or applications may be developed which support one or more profiles. In the case that all functions within the CGI are not provided for in an implementation, it is imperative that the following criteria be met in accordance with the ISO/IEC 9636-1:

a) The Ada procedure mappings for supported non-soliciting functions shall be available to the client.

b) The CGI functions LOOKUP FUNCTION SUPPORT and LOOKUP PROFILE SUPPORT shall be available to the client. If an unsupported soliciting function is present in an implementation, it shall respond by setting the response validity output parameter of the called function to invalid.

c) The CGI functions which are supported by the implementation shall match at least one of the defined profiles and be explicitly documented for the client's inspection.

Throughout this part of ISO/IEC 9638, the following conventions have been adhered to with respect to the naming of Ada packages.

d) The identifier (CGI_) precedes all CGI related Ada packages.

e) The identifier (<PACKAGE_NAME>_CONST) is used to identify an Ada package specification which strictly contains constant definitions.

The following packages specify the mapping of the CGI to the Ada programming language. A package description has been included in order to convey the purpose and intent of each of the defined packages. An effort has also been made, where applicable, to reference supporting clauses and subclauses of this part of ISO/IEC 9638 which provide more detailed information about the content of these package definitions. These packages are described in their entirety in Annex A.

Package: CGI_CONFIG
Description: This package provides a client tailorable interface to the CGI implementation. The objects contained within this package will specify various constraints as well as the level of implementation support desired by a CGI client. This package is defined in clause 5.

Package: CGI_TYPES
Description: This package accounts for all of the CGI types defined within the ISO/IEC 9636 with the exception of the Data Record dependent types. This package is defined in the clause 6.

Package: CGI
Description: This package accounts for all of the CGI procedures defined in the ISO/IEC 9636. This package is defined in clause 7.

Package: CGI_ERROR_CONST
Description: This package defines all of the CGI error values as defined throughout the ISO/IEC 9636 as well as any language binding specific errors defined within this part of ISO/IEC 9638. This package is defined in Annex A.

Package: CGI_FUNCTION_ID_CONST
Description: This package defines all of the CGI function identifiers as specified in Annex A of the ISO/IEC 9636-1. This package is defined in Annex A.

Package: CGI_PROFILE_ID_CONST
Description: This package defines all of the CGI profile identifiers as specified in Annex A of the ISO/IEC 9636-1. This package is defined in Annex A.

Package: CGI_REGISTRATION_CONST
Description: This package defines all of the CGI standardized values which have been specified throughout the ISO/IEC 9636. This package is defined in Annex A of this standard.

Package: CGI_DATA_RECORD_UTILS
Description: This package defines all of the data record types specified in the ISO/IEC 9636 as well as the data record utilities necessary to build and interpret CGI data records. This package is defined in the Binding Defined Utility Functions clause of this standard.

Package: CGI_ERROR_HANDLING_UTILS
Description: This package provides a set of error handing utilities and controls which may be used to specify a language binding or generator error handling mechanism. This package is defined in the Binding Defined Utility Functions clause of this standard.

Package: CGI_PACKING_UTILS
Description: This package provides a set of data packing utilities used to pack large arrays of colour and pixel data for data stream transmission or client storage. This package is defined in the Binding Defined Utility Functions clause of this standard.

Package: CGI_STRING_UTILS
Description: This package defines a set of string conversions necessary to convert between CGI and Ada strings. This package is defined in the Binding Defined Utility Functions clause of this standard.

3.2.10 Client program environment

A client program utilizing an Ada implementation of CGI will need to be aware of the environment in which both CGI and the client program(s) reside.

One aspect of the environment is the Ada program library. The Ada language requires that the client program have access to the program library in which the CGI 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 CGI implementation should specify where the CGI library exists in the system, and how access to the library is acquired.

3.2.11 Registration

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

4 Tables

4.1 Abbreviations used in the Ada language binding

The following abbreviations appear throughout the declarations of types and procedures for this Ada language binding.

ACK	Acknowledgement
ASF	Aspect Source Flag
ASN	Attribute Set Name
AVAIL	Available
BITBLT	Bit Block Transfer
CAP *	Capability
CHAR	Character
CONST	Constant
COORD	Coordinate
CSN	Client Specified Name
DSCRECT *	Drawing Surface Clip Rectangle
DYN	Dynamic
ESC	Escape
EXPAN	Expansion
GDP	Generalized Drawing Primitive
HOR	Horizontal
ID	Identifier
IMPL	Implicit
INDIV	Individual
INQ	Inquire
LIST	<Null>
LIST OF	<Null>
MAX	Maximum
MIN	Minimum
NUM	Number
PREC	Precision
REGEN	Regeneration
REP	Representation
SPECIF	Specification
UTILS	Utilities
VERT	Vertical

* This abbreviation is only used within the CGI type definitions where this identifier appears. This is necessary due to the descriptive types used throughout CGI. In this case only, it has been decided not to abbreviate any CGI function names containing this identifier in order to avoid deprivation of clarity.

4.2 Abbreviation policy in construction of identifiers

In the construction of the data types, function names, etc., the following policy is applied:

The conventions employed by this language binding as they relate to the mapping of CGI function names to Ada

procedures are as follows:

a) All procedure names appear in upper case.

b) Underscores have been substituted for spaces in all procedure names.

c) All CGI functions which set information within state lists have been prefaced with the identifier SET_ in accordance with subclause C.3.3 of ISO/IEC 9636-1.

d) Only approved abbreviations for bindings have been introduced into this standard.

e) The use of abbreviations has been restricted to those CGI functions which are excessive in length. In order to avoid confusion for implementors, abbreviations have been employed in a consistent manner throughout all of the Ada procedures.

f) The identifiers LIST and LIST OF have been omitted from the Ada procedure names which map to the CGI inquiry functions in which they are found.

g) Where abbreviations or the omission of identifiers has been applied within this Ada language binding, there has been a strict observance to maintain the intent and descriptive meaning of any affected functions.

4.3 CGI function names

4.3.1 Alphabetical by bound name

The following table alphabetically lists the Ada procedural binding names as they correspond to each of the CGI functions as specified in the ISO/IEC 9636.

APPEND_TEXT	append text
ASSOCIATE_TRIGGERS	associate triggers
AWAIT_EVENT	await event
BEGIN_FIGURE	begin figure
CELL_ARRAY	cell array
CIRCLE	circle
CIRCULAR_ARC_3_POINT	circular arc 3 point
CIRCULAR_ARC_3_POINT_CLOSE	circular arc 3 point close
CIRCULAR_ARC_CENTRE	circular arc centre
CIRCULAR_ARC_CENTRE_CLOSE	circular arc centre close
CIRCULAR_ARC_CENTRE_REVERSED	circular arc centre reversed
CLOSE_SEGMENT	close segment
CONNECTING_EDGE	connecting edge
COPY_SEGMENT	copy segment
CREATE_BITMAP	create bitmap
CREATE_SEGMENT	create segment
DELETE_ALL_SEGMENTS	delete all segments
DELETE_BITMAP	delete bitmap
DELETE_BUNDLE_REP	delete bundle representation
DELETE_PATTERN	delete pattern
DELETE_PRIMITIVE_ATTRIBUTE_SAVE_SET	delete primitive attribute save set
DELETE_SEGMENT	delete segment
DEQUEUE_CHOICE_EVENT	dequeue choice event
DEQUEUE_ERROR_REPORTS	dequeue error reports
DEQUEUE_GENERAL_EVENT	dequeue general event

DEQUEUE_LOCATOR_EVENT	dequeue locator event
DEQUEUE_PICK_EVENT	dequeue pick event
DEQUEUE_RASTER_EVENT	dequeue raster event
DEQUEUE_STRING_EVENT	dequeue string event
DEQUEUE_STROKE_EVENT	dequeue stroke event
DEQUEUE_VALUATOR_EVENT	dequeue valuator event
DISABLE_EVENTS	disable events
DISJOINT_POLYLINE	disjoint polyline
DISPLAY_BITMAP	display bitmap
DRAW_ALL_SEGMENTS	draw all segments
ECHO_REQUEST_CHOICE	echo request choice
ECHO_REQUEST_GENERAL	echo request general
ECHO_REQUEST_LOCATOR	echo request locator
ECHO_REQUEST_PICK	echo request pick
ECHO_REQUEST_RASTER	echo request raster
ECHO_REQUEST_STRING	echo request string
ECHO_REQUEST_STROKE	echo request stroke
ECHO_REQUEST_VALUATOR	echo request valuator
ELLIPSE	ellipse
ELLIPTICAL_ARC	elliptical arc
ELLIPTICAL_ARC_CLOSE	elliptical arc close
ENABLE_EVENTS	enable events
END_FIGURE	end figure
END_PAGE	end page
ESCAPE	escape
EVENT_QUEUE_BLOCK_CONTROL	event queue block control
EVENT_QUEUE_TRANSFER	event queue transfer
EXECUTE_DEFERRED_ACTIONS	execute deferred actions
FLUSH_DEVICE_EVENTS	flush device events
FLUSH_EVENTS	flush events
GENERALIZED_DRAWING_PRIMITIVE	generalized drawing primitive
GET_ADDITIONAL_PICK_DATA	get additional pick data
GET_ADDITIONAL_RASTER_DATA	get additional raster data
GET_ADDITIONAL_STRING_DATA	get additional string data
GET_ADDITIONAL_STROKE_DATA	get additional stroke data
GET_ESCAPE	get escape
GET_NEW_BITMAP_ID	get new bitmap identifier
GET_NEW_SEGMENT_ID	get new segment identifier
GET_PIXEL_ARRAY	get pixel array
GET_PIXEL_ARRAY_DIMENSIONS	get pixel array dimensions
GET_TEXT_EXTENT	get text extent
INITIALIZE_ECHO_OUTPUT	initialize echo output
INITIALIZE_ECHO_REQUEST	initialize echo request
INITIALIZE_EVENT_QUEUE	initialize event queue
INITIALIZE_LOGICAL_INPUT_DEVICE	initialize logical input device
INITIALIZE_SESSION	initialize
INQ_ASSOCIABLE_TRIGGERS	inquire list of associable triggers
INQ_ASSOCIATED_TRIGGERS	inquire list of associated triggers
INQ_ATTRIBUTE_SET_NAMES_IN_USE	inquire list of attribute set names in use
INQ_AVAIL_CHAR_EXPAN_FACTORS	inquire list of available character expansion factors
INQ_AVAIL_CHAR_HEIGHTS	inquire list of available character heights
INQ_AVAIL_CHAR_ORIENTATIONS	inquire list of available character orientations
INQ_AVAIL_CHAR_SETS	inquire list of available character sets
INQ_AVAIL_CHAR_SPACINGS	inquire list of available character spacings
INQ_AVAIL_EDGE_TYPES	inquire list of available edge types
INQ_AVAIL_HATCH_STYLES	inquire list of available hatch styles
INQ_AVAIL_INPUT_CHAR_SETS	inquire list of available input character sets
INQ_AVAIL_INPUT_DEVICES	inquire list of available input devices
INQ_AVAIL_LINE_TYPES	inquire list of available line types
INQ_AVAIL_MARKER_TYPES	inquire list of available marker types

INQ_AVAIL_SCALED_EDGE_WIDTHS	inquire list of available scaled edge widths
INQ_AVAIL_SCALED_LINE_WIDTHS	inquire list of available scaled line widths
INQ_AVAIL_SCALED_MARKER_SIZES	inquire list of available scaled marker sizes
INQ_AVAIL_TEXT_FONTS	inquire list of available text fonts
INQ_BITMAP_STATE	inquire bitmap state
INQ_CHAR_SET_LIST	inquire character set list
INQ_CHOICE_CAPABILITIES	inquire choice capabilities
INQ_CIE_CHARACTERISTICS	inquire cie characteristics
INQ_CLIPPING_INHERITANCE	inquire clipping inheritance
INQ_COLOUR_CAPABILITY	inquire colour capability
INQ_COLOUR_STATE	inquire colour state
INQ_COLOUR_TABLE_ENTRIES	inquire list of colour table entries
INQ_COMMON_INPUT_DEVICE_PROPERTIES	inquire common input device properties
INQ_COMMON_LID_STATE	inquire common logical input device state
INQ_CONTROL_STATE	inquire control state
INQ_CURRENTLY_EXISTING_ECHO_ENTITIES	inquire list of currently existing echo entities
INQ_CURRENT_PREC_REQUIREMENTS	inquire current precision requirements
INQ_DEVICE_CONTROL_CAPABILITY	inquire device control capability
INQ_DEVICE_DESCRIPTION	inquire device description
INQ_DEVICE_ID	inquire device identification
INQ_DISPLAYABLE_BITMAP_IDS	inquire list of displayable bitmap identifiers
INQ_ECHO_DATA_RECORD	inquire echo data record
INQ_ECHO_ENTITY_STATE	inquire echo entity state
INQ_ECHO_OUTPUT_ACK_TYPES	inquire list of echo output acknowledgement types
INQ_ECHO_OUTPUT_CAPABILITIES	inquire echo output capabilities
INQ_ECHO_OUTPUT_DATA_RECORD	inquire echo output data record
INQ_ECHO_OUTPUT_ECHO_TYPES	inquire list of echo output echo types
INQ_ECHO_OUTPUT_PROMPT_TYPES	inquire list of echo output prompt types
INQ_EDGE_ATTRIBUTES	inquire edge attributes
INQ_EDGE_BUNDLE_INDICES	inquire list of edge bundle indices
INQ_EDGE_CAPABILITY	inquire edge capability
INQ_EDGE_REP	inquire edge representation
INQ_ERROR_HANDLING	inquire error handling
INQ_EVENT_INPUT_STATE	inquire event input state
INQ_FILL_ATTRIBUTES	inquire fill attributes
INQ_FILL_BUNDLE_INDICES	inquire list of fill bundle indices
INQ_FILL_CAPABILITY	inquire fill capability
INQ_FILL_REP	inquire fill representation
INQ_FONT_CAPABILITIES	inquire font capabilities
INQ_FONT_LIST	inquire font list
INQ_GDP_ATTRIBUTES	inquire gdp attributes
INQ_GENERAL_CAPABILITIES	inquire general capabilities
INQ_GENERAL_STATE	inquire general state
INQ_INDIV_SEGMENT_STATE	inquire individual segment state
INQ_INHERITANCE_FILTER_SETTINGS	inquire list of inheritance filter settings
INQ_INPUT_CAPABILITY	inquire input capability
INQ_INPUT_DEVICE_DATA_RECORD	inquire input device data record
INQ_LINE_ATTRIBUTES	inquire line attributes
INQ_LINE_BUNDLE_INDICES	inquire list of line bundle indices
INQ_LINE_CAPABILITY	inquire line capability
INQ_LINE_REP	inquire line representation
INQ_LOCATOR_CAPABILITIES	inquire locator capabilities
INQ_LOCATOR_STATE	inquire locator state
INQ_MARKER_ATTRIBUTES	inquire marker attributes
INQ_MARKER_BUNDLE_INDICES	inquire list of marker bundle indices
INQ_MARKER_CAPABILITY	inquire marker capability
INQ_MARKER_REP	inquire marker representation
INQ_MAX_SIMULTANEOUS_ATTRIBUTE_SETS	inquire maximum number of simultaneously saved attribute sets
INQ_MISCELLANEOUS_CONTROL_STATE	inquire miscellaneous control state
INQ_NON_DISPLAYABLE_BITMAP_IDS	inquire list of non displayable bitmap identifiers

INQ_OBJECT_CLIPPING	inquire object clipping
INQ_OUTPUT_STATE	inquire output state
INQ_PATTERN	inquire pattern
INQ_PATTERN_DIMENSIONS	inquire pattern dimensions
INQ_PATTERN_INDICES	inquire list of pattern indices
INQ_PERMITTED_RASTER_SPOT_CENTRE_SEPARATIONS	list of permitted raster spot centre separations
INQ_PICK_CAPABILITIES	inquire pick capabilities
INQ_PICK_STATE	inquire pick state
INQ_PRIMITIVE_SUPPORT_LEVELS	inquire primitive support levels
INQ_PROFILE_SUPPORT_INDICATORS	inquire list of profile support indicators
INQ_RASTER_CAPABILITY	inquire raster capability
INQ_RASTER_INPUT_CAPABILITIES	inquire raster input capabilities
INQ_RASTER_INPUT_STATE	inquire raster input state
INQ_RASTER_STATE	inquire raster state
INQ_SEGMENT_CAPABILITY	inquire segment capability
INQ_SEGMENT_IDS_IN_USE	inquire list of segment identifiers in use
INQ_SEGMENT_STATE	inquire segment state
INQ_STRING_CAPABILITIES	inquire string capabilities
INQ_STRING_STATE	inquire string state
INQ_STROKE_CAPABILITIES	inquire stroke capabilities
INQ_STROKE_STATE	inquire stroke state
INQ_SUPPORTED_ACK_TYPES	inquire list of supported acknowledgement types
INQ_SUPPORTED_CHAR_CODING_ANNOUNCERS	inquire array of supported character coding announcers
INQ_SUPPORTED_DRAWING_MODE_TRANSPARENCY_PAIRS	list of supported drawing mode transparency pairs
INQ_SUPPORTED_DRAWING_MODE_3_TRANSPARENCY_PAIRS	list of supported drawing mode 3 transparency pairs
INQ_SUPPORTED_ECHO_TYPES	inquire list of supported echo types
INQ_SUPPORTED_GENERAL_FORMAT_IDS	inquire list of supported general format identifiers
INQ_SUPPORTED_GENERAL_MEASURE_FORMATS	inquire list of supported general measure formats
INQ_SUPPORTED_PROMPT_TYPES	inquire list of supported prompt types
INQ_SUPPORTED_VDC_TYPES	inquire supported vdc types
INQ_TEXT_ATTRIBUTES	inquire text attributes
INQ_TEXT_BUNDLE_INDICES	inquire list of text bundle indices
INQ_TEXT_CAPABILITY	inquire text capability
INQ_TEXT_REP	inquire text representation
INQ_VALUATOR_STATE	inquire valuator state
INQ_VDC_TO_DEVICE_MAPPING	inquire vdc to device mapping
LOOKUP_ASPECT_SOURCE_FLAGS	lookup aspect source flags
LOOKUP_ESCAPE_SUPPORT	lookup escape support
LOOKUP_FUNCTION_SUPPORT	lookup function support
LOOKUP_GDP_SUPPORT	lookup gdp support
LOOKUP_GET_ESCAPE_SUPPORT	lookup get escape support
LOOKUP_PROFILE_SUPPORT	lookup profile support
MESSAGE	message
NEW_REGION	new region
PERFORM_ACK	perform acknowledgement
PIXEL_ARRAY	pixel array
POLYGON	polygon
POLYGON_SET	polygon set
POLYLINE	polyline
POLYMARKER	polymarker
PREPARE_DRAWING_SURFACE	prepare drawing surface
PUT_CURRENT_CHOICE_MEASURE	put current choice measure
PUT_CURRENT_GENERAL_MEASURE	put current general measure
PUT_CURRENT_LOCATOR_MEASURE	put current locator measure
PUT_CURRENT_PICK_MEASURE	put current pick measure
PUT_CURRENT_RASTER_MEASURE	put current raster measure
PUT_CURRENT_STRING_MEASURE	put current string measure
PUT_CURRENT_STROKE_MEASURE	put current stroke measure
PUT_CURRENT_VALUATOR_MEASURE	put current valuator measure
RECTANGLE	rectangle

RELEASE_ECHO_OUTPUT	release echo output
RELEASE_EVENT_QUEUE	release event queue
RELEASE_LOGICAL_INPUT_DEVICE	release logical input device
RENAME_SEGMENT	rename segment
REOPEN_SEGMENT	reopen segment
REQUEST_CHOICE	request choice
REQUEST_GENERAL	request general
REQUEST_LOCATOR	request locator
REQUEST_PICK	request pick
REQUEST_RASTER	request raster
REQUEST_STRING	request string
REQUEST_STROKE	request stroke
REQUEST_VALUATOR	request valuator
RESET_REGEN_PENDING	reset regeneration pending
RESTORE_PRIMITIVE_ATTRIBUTES	restore primitive attributes
RESTRICTED_TEXT	restricted text
SAMPLE_CHOICE	sample choice
SAMPLE_GENERAL	sample general
SAMPLE_LOCATOR	sample locator
SAMPLE_PICK	sample pick
SAMPLE_RASTER	sample raster
SAMPLE_STRING	sample string
SAMPLE_STROKE	sample stroke
SAMPLE_VALUATOR	sample valuator
SAVE_PRIMITIVE ATTRIBUTES	save primitive attributes
SET_ALTERNATE_CHAR_SET_INDEX	alternate character set index
SET_ASPECT_SOURCE_FLAGS	aspect source flags
SET_AUXILIARY_COLOUR	auxiliary colour
SET_BACKGROUND_COLOUR	background colour
SET_BITMAP_BACKGROUND_COLOUR	mapped bitmap background colour
SET_BITMAP_FOREGROUND_COLOUR	mapped bitmap foreground colour
SET_CHAR_CODING_ANNOUNCER	character coding announcer
SET_CHAR_EXPAN_FACTOR	character expansion factor
SET_CHAR_HEIGHT	character height
SET_CHAR_ORIENTATION	character orientation
SET_CHAR_SET_INDEX	character set index
SET_CHAR_SET_LIST	character set list
SET_CHAR_SPACING	character spacing
SET_CHOICE_DEVICE_DATA	choice device data
SET_CLIPPING_INHERITANCE	clipping inheritance
SET_CLIP_INDICATOR	clip indicator
SET_CLIP_RECTANGLE	clip rectangle
SET_COLOUR_INDEX_PREC_REQUIREMENT	colour index precision requirement
SET_COLOUR_PREC_REQUIREMENT	colour precision requirement
SET_COLOUR_SELECTION_MODE	colour selection mode
SET_COLOUR_TABLE	colour table
SET_COLOUR_VALUE_EXTENT	colour value extent
SET_CSN_PREC_REQUIREMENT	client specified name precision requirement
SET_DEFERRAL_MODE	set_deferral mode
SET_DEVICE_VIEWPORT	device viewport
SET_DEVICE_VIEWPORT_MAPPING	device viewport mapping
SET_DEVICE_VIEWPORT_SPECIF_MODE	device viewport specification mode
SET_DRAWING_BITMAP	drawing bitmap
SET_DRAWING_MODE	drawing mode
SET_DRAWING_SURFACE_CLIP_INDICATOR	drawing surface clip indicator
SET_DRAWING_SURFACE_CLIP_RECTANGLE	drawing surface clip rectangle
SET_ECHO_CONTROLS	echo controls
SET_ECHO_DATA	echo data
SET_ECHO_OUTPUT_CONTROLS	echo output controls
SET_ECHO_OUTPUT_DATA	echo output data

SET_EDGE_BUNDLE_INDEX	edge bundle index
SET_EDGE_CLIPPING_MODE	edge clipping mode
SET_EDGE_COLOUR	edge colour
SET_EDGE_REP	edge representation
SET_EDGE_TYPE	edge type
SET_EDGE_VISIBILITY	edge visibility
SET_EDGE_WIDTH	edge width
SET_EDGE_WIDTH_SPECIF_MODE	edge width specification mode
SET_ERROR_HANDLING_CONTROL	error handling control
SET_FILL_BITMAP	fill bitmap
SET_FILL_BUNDLE_INDEX	fill bundle index
SET_FILL_COLOUR	fill colour
SET_FILL_REFERENCE_POINT	fill reference point
SET_FILL_REP	fill representation
SET_FONT_LIST	font list
SET_GENERAL_DEVICE_DATA	general device data
SET_HATCH_INDEX	hatch index
SET_IMPL_SEGMENT_REGEN_MODE	implicit segment regeneration mode
SET_INDEX_PREC_REQUIREMENT	index precision requirement
SET_INHERITANCE_FILTER	inheritance filter
SET_INTEGER_PREC_REQUIREMENT	integer precision requirement
SET_INTERIOR_STYLE	interior style
SET_LINE_BUNDLE_INDEX	line bundle index
SET_LINE_CLIPPING_MODE	line clipping mode
SET_LINE_COLOUR	line colour
SET_LINE_REP	line representation
SET_LINE_TYPE	line type
SET_LINE_WIDTH	line width
SET_LINE_WIDTH_SPECIF_MODE	line width specification mode
SET_LOCATOR_DEVICE_DATA	locator device data
SET_MARKER_BUNDLE_INDEX	marker bundle index
SET_MARKER_CLIPPING_MODE	marker clipping mode
SET_MARKER_COLOUR	marker colour
SET_MARKER_REP	marker representation
SET_MARKER_SIZE	marker size
SET_MARKER_SIZE_SPECIF_MODE	marker size specification mode
SET_MARKER_TYPE	marker type
SET_PATTERN_INDEX	pattern index
SET_PATTERN_SIZE	pattern size
SET_PATTERN_TABLE	pattern table
SET_PICK_DEVICE_DATA	pick device data
SET_PICK_ID	pick identifier
SET_RASTER_DEVICE_DATA	raster device data
SET_REAL_PREC_REQUIREMENTS	real precision requirements
SET_SAMPLING_STATE	sampling state
SET_SEGMENT_DETECTABILITY	segment detectability
SET_SEGMENT_DISPLAY_PRIORITY	segment display priority
SET_SEGMENT_HIGHLIGHTING	segment highlighting
SET_SEGMENT_PICK_PRIORITY	segment pick priority
SET_SEGMENT_TRANSFORMATION	segment transformation
SET_SEGMENT_VISIBILITY	segment visibility
SET_STATE_LIST_INQUIRY_SOURCE	state list inquiry source
SET_STRING_DEVICE_DATA	string device data
SET_STROKE_DEVICE_DATA	stroke device data
SET_TEXT_ALIGNMENT	text alignment
SET_TEXT_BUNDLE_INDEX	text bundle index
SET_TEXT_COLOUR	text colour
SET_TEXT_FONT_INDEX	text font index
SET_TEXT_PATH	text path
SET_TEXT_PRECISION	text precision

SET_TEXT_REP	text representation
SET_TRANSPARENCY	transparency
SET_TRANSPARENT_COLOUR	transparent colour
SET_VALUATOR_DEVICE_DATA	valuator device data
SET_VDC_EXTENT	vdc extent
SET_VDC_INTEGER_PREC_REQUIREMENTS	vdc integer precision requirements
SET_VDC_REAL_PREC_REQUIREMENTS	vdc real precision requirements
SET_VDC_TYPE	vdc type
SIMULATE_PICK	simulate pick
SOURCE_DESTINATION_BITBLT	source destination bitblt
TERMINATE_SESSION	terminate
TEXT	text
TILE_3_OPERAND_BITBLT	tile three operand bitblt
UPDATE_CHOICE_ECHO_OUTPUT	update choice echo output
UPDATE_GENERAL_ECHO_OUTPUT	update general echo output
UPDATE_LOCATOR_ECHO_OUTPUT	update locator echo output
UPDATE_PICK_ECHO_OUTPUT	update pick echo output
UPDATE_RASTER_ECHO_OUTPUT	update raster echo output
UPDATE_STRING_ECHO_OUTPUT	update string echo output
UPDATE_STROKE_ECHO_OUTPUT	update stroke echo output
UPDATE_VALUATOR_ECHO_OUTPUT	update valuator echo output

4.3.2 Alphabetical by CGI function name

The following table alphabetically lists the CGI functions denoted in the ISO/IEC 9636 as they correspond to their Ada procedural binding names.

ALTERNATE CHARACTER SET INDEX	set_alternate_char_set_index
APPEND TEXT	append_text
ASPECT SOURCE FLAGS	set_aspect_source_flags
ASSOCIATE TRIGGERS	associate_triggers
AUXILIARY COLOUR	set_auxiliary_colour
AWAIT EVENT	await_event
BACKGROUND COLOUR	set_background_colour
BEGIN FIGURE	begin_figure
CELL ARRAY	cell_array
CHARACTER CODING ANNOUNCER	set_char_coding_announcer
CHARACTER EXPANSION FACTOR	set_char_expan_factor
CHARACTER HEIGHT	set_char_height
CHARACTER ORIENTATION	set_char_orientation
CHARACTER SET INDEX	set_char_set_index
CHARACTER SET LIST	set_char_set_list
CHARACTER SPACING	set_char_spacing
CHOICE DEVICE DATA	set_choice_device_data
CIRCLE	circle
CIRCULAR ARC 3 POINT	circular_arc_3_point
CIRCULAR ARC 3 POINT CLOSE	circular_arc_3_point_close
CIRCULAR ARC CENTRE	circular_arc_centre
CIRCULAR ARC CENTRE CLOSE	circular_arc_centre_close
CIRCULAR ARC CENTRE REVERSED	circular_arc_centre_reversed
CLIENT SPECIFIED NAME PRECISION REQUIREMENT	set_csn_prec_requirement
CLIP INDICATOR	set_clip_indicator
CLIP RECTANGLE	set_clip_rectangle
CLIPPING INHERITANCE	set_clipping_inheritance
CLOSE SEGMENT	close_segment
COLOUR INDEX PRECISION REQUIREMENT	set_colour_index_prec_requirement
COLOUR PRECISION REQUIREMENT	set_colour_prec_requirement
COLOUR SELECTION MODE	set_colour_selection_mode
COLOUR TABLE	set_colour_table

COLOUR VALUE EXTENT	set_colour_value_extent
CONNECTING EDGE	connecting_edge
COPY SEGMENT	copy_segment
CREATE BITMAP	create_bitmap
CREATE SEGMENT	create_segment
DEFERRAL MODE	set_deferral_mode
DELETE ALL SEGMENTS	delete_all_segments
DELETE BITMAP	delete_bitmap
DELETE BUNDLE REPRESENTATION	delete_bundle_rep
DELETE PATTERN	delete_pattern
DELETE PRIMITIVE ATTRIBUTE SAVE SET	delete_primitive_attribute_save_set
DELETE SEGMENT	delete_segment
DEQUEUE CHOICE EVENT	dequeue_choice_event
DEQUEUE ERROR REPORTS	dequeue_error_reports
DEQUEUE GENERAL EVENT	dequeue_general_event
DEQUEUE LOCATOR EVENT	dequeue_locator_event
DEQUEUE PICK EVENT	dequeue_pick_event
DEQUEUE RASTER EVENT	dequeue_raster_event
DEQUEUE STRING EVENT	dequeue_string_event
DEQUEUE STROKE EVENT	dequeue_stroke_event
DEQUEUE VALUATOR EVENT	dequeue_valuator_event
DEVICE VIEWPORT	set_device_viewport
DEVICE VIEWPORT MAPPING	set_device_viewport_mapping
DEVICE VIEWPORT SPECIFICATION MODE	set_device_viewport_specif_mode
DISABLE EVENTS	disable_events
DISJOINT POLYLINE	disjoint_polyline
DISPLAY BITMAP	display_bitmap
DRAW ALL SEGMENTS	draw_all_segments
DRAWING BITMAP	set_drawing_bitmap
DRAWING MODE	set_drawing_mode
DRAWING SURFACE CLIP INDICATOR	set_drawing_surface_clip_indicator
DRAWING SURFACE CLIP RECTANGLE	set_drawing_surface_clip_rectangle
ECHO CONTROLS	set_echo_controls
ECHO DATA	set_echo_data
ECHO OUTPUT CONTROLS	set_echo_output_controls
ECHO OUTPUT DATA	set_echo_output_data
ECHO REQUEST CHOICE	echo_request_choice
ECHO REQUEST GENERAL	echo_request_general
ECHO REQUEST LOCATOR	echo_request_locator
ECHO REQUEST PICK	echo_request_pick
ECHO REQUEST RASTER	echo_request_raster
ECHO REQUEST STRING	echo_request_string
ECHO REQUEST STROKE	echo_request_stroke
ECHO REQUEST VALUATOR	echo_request_valuator
EDGE BUNDLE INDEX	set_edge_bundle_index
EDGE CLIPPING MODE	set_edge_clipping_mode
EDGE COLOUR	set_edge_colour
EDGE REPRESENTATION	set_edge_rep
EDGE TYPE	set_edge_type
EDGE VISIBILITY	set_edge_visibility
EDGE WIDTH	set_edge_width
EDGE WIDTH SPECIFICATION MODE	set_edge_width_specif_mode
ELLIPSE	ellipse
ELLIPTICAL ARC	elliptical_arc
ELLIPTICAL ARC CLOSE	elliptical_arc_close
ENABLE EVENTS	enable_events
END FIGURE	end_figure
END PAGE	end_page
ERROR HANDLING CONTROL	set_error_handling_control
ESCAPE	escape

EVENT QUEUE BLOCK CONTROL	event_queue_block_control
EVENT QUEUE TRANSFER	event_queue_transfer
EXECUTE DEFERRED ACTIONS	execute_deferred_actions
FILL BITMAP	set_fill_bitmap
FILL BUNDLE INDEX	set_fill_bundle_index
FILL COLOUR	set_fill_colour
FILL REFERENCE POINT	set_fill_reference_point
FILL REPRESENTATION	set_fill_rep
FLUSH DEVICE EVENTS	flush_device_events
FLUSH EVENTS	flush_events
FONT LIST	set_font_list
GENERAL DEVICE DATA	set_general_device_data
GENERALIZED DRAWING PRIMITIVE	generalized_drawing_primitive
GET ADDITIONAL PICK DATA	get_additional_pick_data
GET ADDITIONAL RASTER DATA	get_additional_raster_data
GET ADDITIONAL STRING DATA	get_additional_string_data
GET ADDITIONAL STROKE DATA	get_additional_stroke_data
GET ESCAPE	get_escape
GET NEW BITMAP IDENTIFIER	get_new_bitmap_id
GET NEW SEGMENT IDENTIFIER	get_new_segment_id
GET PIXEL ARRAY	get_pixel_array
GET PIXEL ARRAY DIMENSIONS	get_pixel_array_dimensions
GET TEXT EXTENT	get_text_extent
HATCH INDEX	set_hatch_index
IMPLICIT SEGMENT REGENERATION MODE	set_impl_segment_regen_mode
INDEX PRECISION REQUIREMENT	set_index_prec_requirement
INHERITANCE FILTER	set_inheritance_filter
INITIALIZE	initialize_session
INITIALIZE ECHO OUTPUT	initialize_echo_output
INITIALIZE ECHO REQUEST	initialize_echo_request
INITIALIZE EVENT QUEUE	initialize_event_queue
INITIALIZE LOGICAL INPUT DEVICE	initialize_logical_input_device
INQUIRE ARRAY OF SUPPORTED CHARACTER CODING ANNOUNCERS	inq_supported_char_coding_announcers
INQUIRE BITMAP STATE	inq_bitmap_state
INQUIRE CHARACTER SET LIST	inq_char_set_list
INQUIRE CHOICE CAPABILITIES	inq_choice_capabilities
INQUIRE CIE CHARACTERISTICS	inq_cie_characteristics
INQUIRE CLIPPING INHERITANCE	inq_clipping_inheritance
INQUIRE COLOUR CAPABILITY	inq_colour_capability
INQUIRE COLOUR STATE	inq_colour_state
INQUIRE COMMON INPUT DEVICE PROPERTIES	inq_common_input_device_properties
INQUIRE COMMON LOGICAL INPUT DEVICE STATE	inq_common_lid_state
INQUIRE CONTROL STATE	inq_control_state
INQUIRE CURRENT PRECISION REQUIREMENTS	inq_current_prec_requirements
INQUIRE DEVICE CONTROL CAPABILITY	inq_device_control_capability
INQUIRE DEVICE DESCRIPTION	inq_device_description
INQUIRE DEVICE IDENTIFICATION	inq_device_id
INQUIRE ECHO DATA RECORD	inq_echo_data_record
INQUIRE ECHO ENTITY STATE	inq_echo_entity_state
INQUIRE ECHO OUTPUT CAPABILITIES	inq_echo_output_capabilities
INQUIRE ECHO OUTPUT DATA RECORD	inq_echo_output_data_record
INQUIRE EDGE ATTRIBUTES	inq_edge_attributes
INQUIRE EDGE CAPABILITY	inq_edge_capability
INQUIRE EDGE REPRESENTATION	inq_edge_rep
INQUIRE ERROR HANDLING	inq_error_handling
INQUIRE EVENT INPUT STATE	inq_event_input_state
INQUIRE FILL ATTRIBUTES	inq_fill_attributes
INQUIRE FILL CAPABILITY	inq_fill_capability
INQUIRE FILL REPRESENTATION	inq_fill_rep
INQUIRE FONT CAPABILITIES	inq_font_capabilities

INQUIRE FONT LIST	inq_font_list
INQUIRE GDP ATTRIBUTES	inq_gdp_attributes
INQUIRE GENERAL CAPABILITIES	inq_general_capabilities
INQUIRE GENERAL STATE	inq_general_state
INQUIRE INDIVIDUAL SEGMENT STATE	inq_indiv_segment_state
INQUIRE INPUT CAPABILITY	inq_input_capability
INQUIRE INPUT DEVICE DATA RECORD	inq_input_device_data_record
INQUIRE LINE ATTRIBUTES	inq_line_attributes
INQUIRE LINE CAPABILITY	inq_line_capability
INQUIRE LINE REPRESENTATION	inq_line_rep
INQUIRE LIST OF ASSOCIABLE TRIGGERS	inq_associable_triggers
INQUIRE LIST OF ASSOCIATED TRIGGERS	inq_associated_triggers
INQUIRE LIST OF ATTRIBUTE SET NAMES IN USE	inq_attribute_set_names_in_use
INQUIRE LIST OF AVAILABLE CHARACTER EXPANSION FACTORS	inq_avail_char_expan_factors
INQUIRE LIST OF AVAILABLE CHARACTER HEIGHTS	inq_avail_char_heights
INQUIRE LIST OF AVAILABLE CHARACTER ORIENTATIONS	inq_avail_char_orientations
INQUIRE LIST OF AVAILABLE CHARACTER SETS	inq_avail_char_sets
INQUIRE LIST OF AVAILABLE CHARACTER SPACINGS	inq_avail_char_spacings
INQUIRE LIST OF AVAILABLE EDGE TYPES	inq_avail_edge_types
INQUIRE LIST OF AVAILABLE HATCH STYLES	inq_avail_hatch_styles
INQUIRE LIST OF AVAILABLE INPUT CHARACTER SETS	inq_avail_input_char_sets
INQUIRE LIST OF AVAILABLE INPUT DEVICES	inq_avail_input_devices
INQUIRE LIST OF AVAILABLE LINE TYPES	inq_avail_line_types
INQUIRE LIST OF AVAILABLE MARKER TYPES	inq_avail_marker_types
INQUIRE LIST OF AVAILABLE SCALED EDGE WIDTHS	inq_avail_scaled_edge_widths
INQUIRE LIST OF AVAILABLE SCALED LINE WIDTHS	inq_avail_scaled_line_widths
INQUIRE LIST OF AVAILABLE SCALED MARKER SIZES	inq_avail_scaled_marker_sizes
INQUIRE LIST OF AVAILABLE TEXT FONTS	inq_avail_text_fonts
INQUIRE LIST OF COLOUR TABLE ENTRIES	inq_colour_table_entries
INQUIRE LIST OF CURRENTLY EXISTING ECHO ENTITIES	inq_currently_existing_echo_entities
INQUIRE LIST OF DISPLAYABLE BITMAP IDENTIFIERS	inq_displayable_bitmap_ids
INQUIRE LIST OF ECHO OUTPUT ACKNOWLEDGEMENT TYPES	inq_echo_output_ack_types
INQUIRE LIST OF ECHO OUTPUT ECHO TYPES	inq_echo_output_echo_types
INQUIRE LIST OF ECHO OUTPUT PROMPT TYPES	inq_echo_output_prompt_types
INQUIRE LIST OF EDGE BUNDLE INDICES	inq_edge_bundle_indices
INQUIRE LIST OF FILL BUNDLE INDICES	inq_fill_bundle_indices
INQUIRE LIST OF INHERITANCE FILTER SETTINGS	inq_inheritance_filter_settings
INQUIRE LIST OF LINE BUNDLE INDICES	inq_line_bundle_indices
INQUIRE LIST OF MARKER BUNDLE INDICES	inq_marker_bundle_indices
INQUIRE LIST OF NON DISPLAYABLE BITMAP IDENTIFIERS	inq_non_displayable_bitmap_ids
INQUIRE LIST OF PATTERN INDICES	inq_pattern_indices
INQUIRE LIST OF PERMITTED RASTER SPOT CENTRE SEPARATION	inq_permitted_raster_spot_centre_separations
INQUIRE LIST OF PROFILE SUPPORT INDICATORS	inq_profile_support_indicators
INQUIRE LIST OF SEGMENT IDENTIFIERS IN USE	inq_segment_ids_in_use
INQUIRE LIST OF SUPPORTED ACKNOWLEDGEMENT TYPES	inq_supported_ack_types
INQUIRE LIST OF SUPPORTED DRAWING MODE 3 TRANSPARENCY PAIRS	inq_supported_drawing_mode_3_transparency_pairs
INQUIRE LIST OF SUPPORTED DRAWING MODE TRANSPARENCY PAIRS	inq_supported_drawing_mode_transparency_pairs
INQUIRE LIST OF SUPPORTED ECHO TYPES	inq_supported_echo_types
INQUIRE LIST OF SUPPORTED GENERAL FORMAT IDENTIFIERS	inq_supported_general_format_ids
INQUIRE LIST OF SUPPORTED GENERAL MEASURE FORMATS	inq_supported_general_measure_formats
INQUIRE LIST OF SUPPORTED PROMPT TYPES	inq_supported_prompt_types
INQUIRE LIST OF TEXT BUNDLE INDICES	inq_text_bundle_indices
INQUIRE LOCATOR CAPABILITIES	inq_locator_capabilities
INQUIRE LOCATOR STATE	inq_locator_state
INQUIRE MARKER ATTRIBUTES	inq_marker_attributes
INQUIRE MARKER CAPABILITY	inq_marker_capability
INQUIRE MARKER REPRESENTATION	inq_marker_rep
INQUIRE MAXIMUM NUMBER OF SIMULTANEOUSLY SAVED ATTRIBUTE SETS	inq_max_simultaneous_attribute_sets
INQUIRE MISCELLANEOUS CONTROL STATE	inq_miscellaneous_control_state
INQUIRE OBJECT CLIPPING	inq_object_clipping

INQUIRE OUTPUT STATE	inq_output_state
INQUIRE PATTERN	inq_pattern
INQUIRE PATTERN DIMENSIONS	inq_pattern_dimensions
INQUIRE PICK CAPABILITIES	inq_pick_capabilities
INQUIRE PICK STATE	inq_pick_state
INQUIRE PRIMITIVE SUPPORT LEVELS	inq_primitive_support_levels
INQUIRE RASTER CAPABILITY	inq_raster_capability
INQUIRE RASTER INPUT CAPABILITIES	inq_raster_input_capabilities
INQUIRE RASTER INPUT STATE	inq_raster_input_state
INQUIRE RASTER STATE	inq_raster_state
INQUIRE SEGMENT CAPABILITY	inq_segment_capability
INQUIRE SEGMENT STATE	inq_segment_state
INQUIRE STRING CAPABILITIES	inq_string_capabilities
INQUIRE STRING STATE	inq_string_state
INQUIRE STROKE CAPABILITIES	inq_stroke_capabilities
INQUIRE STROKE STATE	inq_stroke_state
INQUIRE SUPPORTED VDC TYPES	inq_supported_vdc_types
INQUIRE TEXT ATTRIBUTES	inq_text_attributes
INQUIRE TEXT CAPABILITY	inq_text_capability
INQUIRE TEXT REPRESENTATION	inq_text_rep
INQUIRE VALUATOR STATE	inq_valuator_state
INQUIRE VDC TO DEVICE MAPPING	inq_vdc_to_device_mapping
INTEGER PRECISION REQUIREMENT	set_integer_prec_requirement
INTERIOR STYLE	set_interior_style
LINE BUNDLE INDEX	set_line_bundle_index
LINE CLIPPING MODE	set_line_clipping_mode
LINE COLOUR	set_line_colour
LINE REPRESENTATION	set_line_rep
LINE TYPE	set_line_type
LINE WIDTH	set_line_width
LINE WIDTH SPECIFICATION MODE	set_line_width_specif_mode
LOCATOR DEVICE DATA	set_locator_device_data
LOOKUP ASPECT SOURCE FLAGS	lookup_aspect_source_flags
LOOKUP ESCAPE SUPPORT	lookup_escape_support
LOOKUP FUNCTION SUPPORT	lookup_function_support
LOOKUP GDP SUPPORT	lookup_gdp_support
LOOKUP GET ESCAPE SUPPORT	lookup_get_escape_support
LOOKUP PROFILE SUPPORT	lookup_profile_support
MAPPED BITMAP BACKGROUND COLOUR	set_bitmap_background_colour
MAPPED BITMAP FOREGROUND COLOUR	set_bitmap_foreground_colour
MARKER BUNDLE INDEX	set_marker_bundle_index
MARKER CLIPPING MODE	set_marker_clipping_mode
MARKER COLOUR	set_marker_colour
MARKER REPRESENTATION	set_marker_rep
MARKER SIZE	set_marker_size
MARKER SIZE SPECIFICATION MODE	set_marker_size_specif_mode
MARKER TYPE	set_marker_type
MESSAGE	message
NEW REGION	new_region
PATTERN INDEX	set_pattern_index
PATTERN SIZE	set_pattern_size
PATTERN TABLE	set_pattern_table
PERFORM ACKNOWLEDGEMENT	perform_ack
PICK DEVICE DATA	set_pick_device_data
PICK IDENTIFIER	set_pick_id
PIXEL ARRAY	pixel_array
POLYGON	polygon
POLYGON SET	polygon_set
POLYLINE	polyline
POLYMARKER	polymarker

PREPARE DRAWING SURFACE	prepare_drawing_surface
PUT CURRENT CHOICE MEASURE	put_current_choice_measure
PUT CURRENT GENERAL MEASURE	put_current_general_measure
PUT CURRENT LOCATOR MEASURE	put_current_locator_measure
PUT CURRENT PICK MEASURE	put_current_pick_measure
PUT CURRENT RASTER MEASURE	put_current_raster_measure
PUT CURRENT STRING MEASURE	put_current_string_measure
PUT CURRENT STROKE MEASURE	put_current_stroke_measure
PUT CURRENT VALUATOR MEASURE	put_current_valuator_measure
RASTER DEVICE DATA	set_raster_device_data
REAL PRECISION REQUIREMENTS	set_real_prec_requirements
RECTANGLE	rectangle
RELEASE ECHO OUTPUT	release_echo_output
RELEASE EVENT QUEUE	release_event_queue
RELEASE LOGICAL INPUT DEVICE	release_logical_input_device
RENAME SEGMENT	rename_segment
REOPEN SEGMENT	reopen_segment
REQUEST CHOICE	request_choice
REQUEST GENERAL	request_general
REQUEST LOCATOR	request_locator
REQUEST PICK	request_pick
REQUEST RASTER	request_raster
REQUEST STRING	request_string
REQUEST STROKE	request_stroke
REQUEST VALUATOR	request_valuator
RESET REGENERATION PENDING	reset_regen_pending
RESTORE PRIMITIVE ATTRIBUTES	restore_primitive_attributes
RESTRICTED TEXT	restricted_text
SAMPLE CHOICE	sample_choice
SAMPLE GENERAL	sample_general
SAMPLE LOCATOR	sample_locator
SAMPLE PICK	sample_pick
SAMPLE RASTER	sample_raster
SAMPLE STRING	sample_string
SAMPLE STROKE	sample_stroke
SAMPLE VALUATOR	sample_valuator
SAMPLING STATE	set_sampling_state
SAVE PRIMITIVE ATTRIBUTES	save_primitive_attributes
SEGMENT DETECTABILITY	set_segment_detectability
SEGMENT DISPLAY PRIORITY	set_segment_display_priority
SEGMENT HIGHLIGHTING	set_segment_highlighting
SEGMENT PICK PRIORITY	set_segment_pick_priority
SEGMENT TRANSFORMATION	set_segment_transformation
SEGMENT VISIBILITY	set_segment_visibility
SIMULATE PICK	simulate_pick
SOURCE DESTINATION BITBLT	source_destination_bitblt
STATE LIST INQUIRY SOURCE	set_state_list_inquiry_source
STRING DEVICE DATA	set_string_device_data
STROKE DEVICE DATA	set_stroke_device_data
TERMINATE	terminate_session
TEXT	text
TEXT ALIGNMENT	set_text_alignment
TEXT BUNDLE INDEX	set_text_bundle_index
TEXT COLOUR	set_text_colour
TEXT FONT INDEX	set_text_font_index
TEXT PATH	set_text_path
TEXT PRECISION	set_text_precision
TEXT REPRESENTATION	set_text_rep
TILE THREE OPERAND BITBLT	tile_3_operand_bitblt
TRANSPARENCY	set_transparency

TRANSPARENT COLOUR	set_transparent_colour
UPDATE CHOICE ECHO OUTPUT	update_choice_echo_output
UPDATE GENERAL ECHO OUTPUT	update_general_echo_output
UPDATE LOCATOR ECHO OUTPUT	update_locator_echo_output
UPDATE PICK ECHO OUTPUT	update_pick_echo_output
UPDATE RASTER ECHO OUTPUT	update_raster_echo_output
UPDATE STRING ECHO OUTPUT	update_string_echo_output
UPDATE STROKE ECHO OUTPUT	update_stroke_echo_output
UPDATE VALUATOR ECHO OUTPUT	update_valuator_echo_output
VALUATOR DEVICE DATA	set_valuator_device_data
VDC EXTENT	set_vdc_extent
VDC INTEGER PRECISION REQUIREMENTS	set_vdc_integer_prec_requirements
VDC REAL PRECISION REQUIREMENTS	set_vdc_real_prec_requirements
VDC TYPE	set_vdc_type

5 CGI configuration values

The following set of constants define the client tailorable portion of CGI. These constants are declared in the CGI configuration package and are used to define and or constrain various CGI types. These configuration values 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 CGI configuration package may be used to compute these values and/or provide initial values. An implementation of CGI may permit the client programmer to replace the body of the CGI configuration package. In such an implementation, instructions for replacing the package body should be explicitly defined in the implementation documentation.

In general, these configuration values are provided in order to achieve one of the following two functionalities:

a) Notify the implementation of the client's constraints which are to be imposed on the declaration of various types and subtypes.

b) Notify the implementation of the client's desires in tailoring the language binding.

The values assigned to these constants are for illustration only and in no way impose any obligations which are required by either an implementation or an application of CGI.

It is also feasible that an implementation may want to enhance this configuration package for the following purposes:

c) In order to provide the client with increased capability.

d) In order to obtain pertinent information from the client in regard to the interpreter capabilities and configuration.

If configuration values are added under the precept of number two above, they should be added with the intention of obtaining information about the interpreter which is static in its nature (i.e. description table information). The intent is to provide the implementation with information which is necessary to increase or enhance performance and which is in no way restrictive to the client. Any additional objects which are added to this package by an implementation should be explicitly documented with attention to the object's definition, purpose, and units (if applicable).

REAL_DIGITS

Purpose - Specifies the number of digits required by the client in order to define a CGI floating point real value. This value will be used in the declaration of the CGI floating point real type.

VDC_REAL_DIGITS

Purpose - Specifies the number of digits required by the client in order to define a CGI VDC floating point real value. This value will be used in the declaration of the CGI VDC floating point real type.

MIN_FIXED_POINT_WHOLE_VALUE

Purpose - Specifies the minimum value for the whole part of a CGI fixed point real. This value will be used in the declaration of the CGI fixed point real type.

MAX_FIXED_POINT_WHOLE_VALUE

Purpose - Specifies the maximum value for the whole part of a CGI fixed point real. This value will be used in the declaration of the CGI fixed point real type.

MAX_FIXED_POINT_FRACTION_VALUE

Purpose - Specifies the maximum value for the fractional part of a CGI fixed point real. This value will be used in the declaration of the CGI fixed point real type.

VDC_MIN_FIXED_POINT_WHOLE_VALUE

Purpose - Specifies the minimum value for the whole part of a CGI fixed point VDC real. This value will be used in the declaration of the CGI VDC fixed point real type.

VDC_MAX_FIXED_POINT_WHOLE_VALUE

Purpose - Specifies the maximum value for the whole part of a CGI fixed point VDC real. This value will be used in the declaration of the CGI VDC fixed point real type.

VDC_MAX_FIXED_POINT_FRACTION_VALUE

Purpose - Specifies the maximum value for the fractional part of a CGI VDC fixed point real. This value will be used in the declaration of the CGI fixed point real type.

MIN_INTEGER

Purpose - Specifies the minimum value of a CGI signed integer. The CGI integer range will be used to constrain the basic integer data type. For an implementation which supports varying integer precision, this range should represent the largest integer precision achievable on the host machine.

MAX_INTEGER

Purpose - Specifies the maximum value of a CGI signed integer. The CGI integer range will be used to constrain the basic integer data type. For an implementation which supports varying integer precision, this range should represent the largest integer precision achievable on the host machine.

VDC_MAX_INTEGER

Purpose - Specifies the maximum value of a CGI signed VDC integer. The CGI VDC integer range will be used to constrain the basic VDC integer data type. For an implementation which supports varying VDC integer precision, this range should represent the largest VDC integer precision achievable on the host machine.

MIN_INDEX

Purpose - Specifies the minimum value of a CGI index. The CGI index range will be used to constrain the CGI basic index data type. For an implementation which supports varying index precision, this range should represent the largest index precision achievable on the host machine.

MAX_INDEX

Purpose - Specifies the maximum value of a CGI index. The CGI index range will be used to constrain the CGI basic index data type. For an implementation which supports varying index precision, this range should represent the largest index precision achievable on the host machine.

MIN_CSN

Purpose - Specifies the minimum value of a CGI client specified name. The CGI CSN range will be used to constrain the CGI basic client specified name data type. For an implementation which supports varying CSN precision, this range should represent the largest CSN precision achievable on the host machine.

MAX_CSN

Purpose - Specifies the maximum value of a CGI client specified name. The CGI CSN range will be used to constrain the CGI basic client specified name data type. For an implementation which supports varying CSN precision, this range should represent the largest CSN precision achievable on the host machine.

MAX_COLOUR_INDEX

Purpose - Specifies the maximum value of a CGI colour index. The CGI colour index range will be used to constrain the CGI basic colour index data type. For an implementation which supports varying colour index precision, this range should represent the largest colour index precision achievable on the host machine.

MIN_DIRECT_COLOUR

Purpose - Specifies the minimum value of a CGI direct colour specifier. The CGI direct colour value range will be used to constrain the CGI basic direct colour value data type. For an implementation which supports varying direct colour value precision, this range should represent the largest direct colour value precision achievable on the host machine.

MAX_DIRECT_COLOUR

Purpose - Specifies the maximum value of a CGI direct colour specifier. The CGI direct colour value range will be used to constrain the CGI basic direct colour value data type. For an implementation which supports varying direct colour value precision, this range should represent the largest direct colour value precision achievable on the host machine.

MAX_INPUT_COLOUR_INDEX

Purpose - Specifies the maximum value of a CGI input colour index. The CGI input colour index range will be used to constrain the CGI abstract input colour data type. For an implementation which supports raster input, this range should represent the maximum bits per colour in the Raster Class Specific Logical Input Device State List.

MIN_DIRECT_INPUT_COLOUR

Purpose - Specifies the minimum value of a CGI direct input colour value. The CGI direct input colour value range will be used to constrain the CGI abstract input colour data type. For an implementation which supports raster input, this range should represent the maximum bits per colour in the Raster Class Specific Logical Input Device State List.

MAX_DIRECT_INPUT_COLOUR

Purpose - Specifies the maximum value of a CGI direct input colour value. The CGI direct input colour value range will be used to constrain the CGI abstract input colour data type. For an implementation which supports raster input, this range should represent the maximum bits per colour in the Raster Class Specific Logical Input Device State List.

MIN_DEVICE_COORD

Purpose - Specifies the minimum value of the CGI abstract device coordinate data type. This value will be used in the declaration of the CGI device coordinate data type.

MAX_DEVICE_COORD

Purpose - Specifies the maximum value of the CGI abstract device coordinate data type. This value will be used in the declaration of the CGI device coordinate data type.

MIN_INTRINSIC_NAME

Purpose - Specifies the minimum value of the CGI abstract intrinsic name data type. This value will be used in the declaration of the CGI intrinsic name data type.

MAX_INTRINSIC_NAME

Purpose - Specifies the maximum value of the CGI abstract intrinsic name data type. This value will be used in the declaration of the CGI intrinsic name data type.

MIN_INPUT_SURFACE_COORD

Purpose - Specifies the minimum value of the CGI abstract input surface coordinate data type. This value will be used in the declaration of the CGI input surface coordinate data type.

MAX_INPUT_SURFACE_COORD

Purpose - Specifies the maximum value of the CGI abstract input surface coordinate data type. This value will be used in the declaration of the CGI input surface coordinate data type.

MAX_LOST_ERROR_COUNT

Purpose - Specifies the maximum value of the Lost Error Count part of the error report data type. This value will be used in the declaration of the CGI error report data type.

MAX_CHAR_SET_LENGTH

Purpose - Specifies the maximum number of octets within a CGI character set string. This value will be used in the declaration of the CGI character set type. This may be useful for applications of CGI (client programs) which know the maximum size of CGI character set strings or which want to limit the character set string size due to memory constraints.

MAX_FIXED_STRING_LENGTH

Purpose - Specifies the maximum number of octets within a CGI fixed string. This value will be used in the declaration of the CGI fixed string type. This may be useful for applications of CGI (client programs) which know the maximum size of CGI fixed strings or which want to limit the string size due to memory constraints.

MAX_STRING_LENGTH

Purpose - Specifies the maximum number of octets within a CGI string. This value will be used in the declaration of the CGI string type. This may be useful for applications of CGI (client programs) which know the maximum size of CGI strings or which want to limit the string size due to memory constraints.

CGI_ERROR_NOTIFICATION

Purpose - Specifies whether or not an application wishes to be notified by the implementation when errors occur. Normally CGI errors will be stored in the CGI error queue until the client performs a DEQUEUE ERROR REPORTS. This constant allows the client to obtain knowledge of errors (only the errors detected by the CGI generator/language binding) as they are detected. When an error is detected by the CGI generator/language binding and the error handling state is set to REPORTING ON for the associated error class, the implementation will raise the CGI ERROR defined in the CGI Exceptions clause of this standard.

SIZE_OF_GENERATOR_ERROR_QUEUE

Purpose - Specifies the maximum number of error reports which may be held in the CGI generator/language binding error queue before error reports are lost. This value will be used in the declaration of the CGI generator error queue type.

SIZE_OF_INTERPRETER_ERROR_QUEUE

Purpose - Specifies the maximum number of error reports which may be held in the CGI Interpreter error queue before error reports are lost. This value will be used in the declaration of the CGI error queue type.

LAST_ARRAY_INDEX

Purpose - Specifies the last index of the array index type. The array index type is used as a common array index range for CGI array declarations.

FIRST_ENTRY_IN_COLOUR_TABLE

Purpose - Specifies the index of the first entry in the colour table of the graphics device. This value will be used in the definition of the colour table type.

LAST_ENTRY_IN_COLOUR_TABLE

Purpose - Specifies the index of the last entry in the colour table of the graphics device. This value will be used in the definition of the colour table type.

---

6 Type definitions

This clause provides all of the basic and abstract data types which will be utilized throughout the CGI Ada language binding. This clause is organized as follows:

a) Array index ranging
Defines the index range for array declarations.

b) Representation of CGI basic data types
Defines the representation of the CGI basic data types for this CGI Ada Language Binding. These types are abstractly defined in the ISO/IEC 9636.

c) Representation of CGI strings
Defines the implementation of CGI strings for this CGI Ada Language Binding.

d) Representation of CGI data records
Defines the implementation of CGI data records for this CGI Ada Language Binding.

e) Representation of CGI abstract data types
Defines the representation of the CGI abstract data types for this CGI Ada Language Binding. These types are abstractly defined in the ISO/IEC 9636.

f) Representation of CGI enumerated data types
Defines the implementation of the CGI enumerated types for this CGI Ada Language Binding.

g) CGI Ada record types
Defines various record structures used throughout the CGI Ada Language Binding.

h) CGI Ada subtypes
Defines various Ada subtypes used throughout the CGI Ada Language Binding

i) CGI Ada array types
Defines various array types used throughout the CGI Ada Language Binding

j) CGI Ada access types
Defines the Ada access types used throughout the CGI Ada Language Binding.

k) CGI Ada exceptions
Defines the Ada exceptions included within this Ada Language Binding.

6.1 Array index ranging

Since the format of this type definitions clause is intended to provide a logical building and grouping of the types used throughout the CGI, it is relevant that the indexing of arrays be discussed here, prior to its usage. This language binding has defined a mechanism which allows both the client and the implementation to index into CGI array types in a common manner. This mechanism consists of an array index range type and an array index type. The array index range specifies all non-negative values of the desired array range including zero. The upper limit of the array index range type is specified in the CGI configuration package. The lower limit of zero provides for the client's declaration of null arrays.

---

type ARRAY_INDEX_RANGE_TYPE is range 0..LAST_ARRAY_INDEX ;

Subtype - Array Index Range Type
Purpose - Specifies an array index range which will be used to define discriminated record length types. This range will include the index value 0 for null arrays.

---

subtype ARRAY_INDEX_TYPE is ARRAY_INDEX_RANGE_TYPE
range 1..ARRAY_INDEX_RANGE_TYPE'LAST ;

Subtype - Array Index Type
Purpose - Used to specify CGI array ranges (positive arrays only).

The array index type and the array index range type are used throughout this language binding to declare various CGI record types which contain arrays. The array index range type is used to define an array range (length), the upper limit of which is normally specified by the client through the CGI configuration package. This length is then used in the definition of a discriminated record type to constrain an array. The array index type is a subtype of the array index range type and defines a positive array range. An example of how this technique has been applied in this language binding appears below:

```
type ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of DATA_TYPE ;

subtype LENGTH_TYPE is ARRAY_INDEX_RANGE_TYPE range 0..CLIENT_SPECIFIED_MAX ;

type RECORD_TYPE( LENGTH : LENGTH_TYPE := ARRAY_INDEX_TYPE'LAST ) is
  record
    DATA : ARRAY_TYPE( 1..LENGTH ) ;
  end record ;
```

6.2 Representation of CGI basic data types

This subclause contains the Ada mapping to the basic CGI data types. Some CGI types may vary with the current selection mode controls in effect. For example, the VDC type may be integer, fixed point real or floating point real. Types of this nature have been mapped to Ada variant records within this language binding. Since the valid parts of these variants may change throughout the CGI session, it is the intention of this Ada Language Binding that all variants be defaulted. The selected defaults for these variants will be determined according to the various specification modes listed within the ISO/IEC 9636, if applicable. The first part of this subclause will define the enumerations which will be used for the variant parts of these records.

In order to provide a complete definition of the basic data types in one section of this standard, any intermediate type definitions defined by this language binding, which are used to further define the CGI basic data types have been presented here. The intermediate type definitions most often appear as types which apply values specified in the CGI configuration package. An ISO/IEC 9636-1 reference has been provided for the types which map directly to the basic data types within the ISO/IEC 9636.

The precision controls in effect during a CGI session determine the precision with which the various data types are to be transferred across a data stream. An implementation of this language binding to a data stream generator will be responsible for extracting or inserting data transferred across the CGI data stream into its proper Ada representations.

These type definitions have been listed in alphabetical order.

CGI Basic Data Type Variant Declarations

```
type REAL_MODE_TYPE is ( FIXED_POINT, FLOATING_POINT ) ;
```

Type - Real Mode Type
Purpose - Determines whether CGI reals will be specified as fixed or floating point.

```
type VDC_MODE_TYPE is ( INTEGER_VDC, FIXED_POINT_VDC, FLOATING_POINT_VDC ) ;
```

Type - VDC Mode Type
Purpose - Determines whether Virtual Device Coordinates will be specified as integer or real values.

---

type COLOUR_INDEX_TYPE is range 0..MAX_COLOUR_INDEX ;

Type - Colour Index Type
Purpose - Basic data type, colour index. This unsigned index, subject to COLOUR INDEX PRECISION within a data stream encoding as specified by COLOUR INDEX PRECISION REQUIREMENT, is an index into the colour table.

Reference - ISO/IEC 9636-1 : CI data type

---

type CSN_TYPE is range MIN_CSN..MAX_CSN ;

Type - CSN Type
Purpose - Basic data type, client specified name. Subject to CLIENT SPECIFIED NAME PRECISION within a data stream encoding as specified by CLIENT SPECIFIED NAME REQUIREMENT.

Reference - ISO/IEC 9636-1 : CSN data type

---

type DIRECT_COLOUR_VALUE_RANGE_TYPE is range
MIN_DIRECT_COLOUR..MAX_DIRECT_COLOUR ;

Type - Direct Colour Value Range Type
Purpose - Specifies a range of direct colour value.

---

type DIRECT_COLOUR_VALUE_TYPE is
record
RED : DIRECT_COLOUR_VALUE_RANGE_TYPE ;
GREEN : DIRECT_COLOUR_VALUE_RANGE_TYPE ;
BLUE : DIRECT_COLOUR_VALUE_RANGE_TYPE ;
end record ;

Type - Direct Colour Value Type
Purpose - Basic data type, direct colour value. This is a red-green-blue triple subject to COLOUR PRECISION within a data stream encoding as specified by COLOUR PRECISION REQUIREMENT.

Reference - ISO/IEC 9636-1 : CD data type

---

type DIRECT_INPUT_COLOUR_VALUE_RANGE_TYPE is range
MIN_DIRECT_INPUT_COLOUR..MAX_DIRECT_INPUT_COLOUR ;

Type - Direct Input Colour Value Range Type

Purpose - Specifies a range of direct input colour value.

```
type DIRECT_INPUT_COLOUR_VALUE_TYPE is
  record
    RED   : DIRECT_INPUT_COLOUR_VALUE_RANGE_TYPE ;
    GREEN : DIRECT_INPUT_COLOUR_VALUE_RANGE_TYPE ;
    BLUE  : DIRECT_INPUT_COLOUR_VALUE_RANGE_TYPE ;
  end record ;
```

Type - Direct Input Colour Value Type
Purpose - Specifies an direct input colour value type which is used to achieve colour values for raster input devices.

```
type FIXED_INTEGER_8_TYPE is range -( 2 ** 7 )..( 2 ** 7 ) - 1 ;
```

Type - Fixed Integer 8 Type
Purpose - Basic data type, fixed integer 8. Of fixed precision i.e. not subject to either INTEGER or INDEX precision within a data stream encoding.

Reference - ISO/IEC 9636-1 : IF8 data type

```
type FIXED_INTEGER_16_TYPE is range -( 2 ** 15 )..( 2 ** 15 ) - 1 ;
```

Type - Fixed Integer 16 Type
Purpose - Basic data type, fixed integer 16. Of fixed precision i.e. not subject to either INTEGER or INDEX precision within a data stream encoding.

Reference - ISO/IEC 9636-1 : IF16 data type

```
type FIXED_INTEGER_32_TYPE is range -( 2 ** 31 )..( 2 ** 31 ) - 1 ;
```

Type - Fixed Integer 32 Type
Purpose - Basic data type, fixed integer 32. Of fixed precision i.e. not subject to either INTEGER or INDEX precision within a data stream encoding.

Reference - ISO/IEC 9636-1 : IF32 data type

```
type FIXED_POINT_FRACTION_PART_TYPE is range 0..MAX_FIXED_POINT_FRACTION_VALUE ;
```

Type - Fixed Point Fraction Part Type
Purpose - Denotes the fractional part definition of a CGI fixed point real type.

```
type FIXED_POINT_WHOLE_PART_TYPE is range
  MIN_FIXED_POINT_WHOLE_VALUE..MAX_FIXED_POINT_WHOLE_VALUE ;
```

Type - Fixed Point Whole Part Type
Purpose - Denotes the whole part definition of a CGI fixed point real type.

```
type FIXED_POINT_REAL_TYPE is
  record
    WHOLE_PART    : FIXED_POINT_WHOLE_PART_TYPE ;
    FRACTION_PART : FIXED_POINT_FRACTION_PART_TYPE ;
  end record ;
```

Type - Fixed Point Real Type
Purpose - Denotes the CGI fixed point real type.

```
subtype FLOATING_POINT_REAL_TYPE is digits REAL_DIGITS ;
```

Subtype - Floating Point Real Type
Purpose - Denotes the CGI floating point real type.

```
type INDEX_TYPE is range MIN_INDEX..MAX_INDEX ;
```

Subtype - Index Type
Purpose - Basic data type, index. Realized as an integer, but subject to INDEX PRECISION within a data stream encoding as specified by INDEX PRECISION REQUIREMENT. Often the range of permitted values for a particular instance of an index data type will be specified by the IS 9636-1. For some instances, positive values are reserved for standard use while negative ones are available for private use.

Reference - ISO/IEC 9636-1 : IX data type

```
type INPUT_COLOUR_INDEX_TYPE is range 0..MAX_INPUT_COLOUR_INDEX ;
```

Type - Input Colour Index Type
Purpose - Specifies an input colour index type which is used to achieve colour values for raster input devices.

```
type INPUT_COLOUR_TYPE( COLOUR : YES_NO_FLAG_TYPE := NO ) is
  record
    case COLOUR is
      when NO  => INDEXED_REALIZATION : INPUT_COLOUR_INDEX_TYPE ;
      when YES => DIRECT_REALIZATION  : DIRECT_INPUT_COLOUR_VALUE_TYPE ;
    end case ;
  end record ;
```

Type - Input Colour Type
Purpose - The basic data type ICO, input colour, obtains its representation and precision based on the Colour and Bits per Colour entries in the Raster LID State List. When Colour is YES, realization is identical to CD with local colour precision = Bits per Colour. When Colour is NO, realization is identical to CI with local colour index precision = Bits per Colour.

Reference - ISO/IEC 9636-1 : ICO data type

```
type INTEGER_TYPE is range MIN_INTEGER..MAX_INTEGER ;
```

Type - Integer Type
Purpose - Basic data type, integer. Subject to INTEGER PRECISION within a data stream encoding as specified by PRECISION REQUIREMENT.

Reference - ISO/IEC 9636-1 : I data type

```
type REAL_TYPE( REAL_MODE : REAL_MODE_TYPE := FLOATING_POINT ) is
  record
    case REAL_MODE is
      when FLOATING_POINT => FLOAT_DATA : FLOATING_POINT_REAL_TYPE ;
      when FIXED_POINT    => FIXED_DATA  : FIXED_POINT_REAL_TYPE ;
    end case ;
  end record ;
```

Type - Real Type
Purpose - Basic data type, real. Subject to REAL PRECISION within a data stream encoding as specified by REAL PRECISION REQUIREMENTS.

Reference - ISO/IEC 9636-1 : R data type

```
type VDC_FIXED_POINT_FRACTION_PART_TYPE is range
  0..VDC_MAX_FIXED_POINT_FRACTION_VALUE ;
```

Type - VDC Fixed Point Fraction Part Type
Purpose - Denotes the fractional part definition of a CGI fixed point VDC real type.

```
type VDC_FIXED_POINT_WHOLE_PART_TYPE is range
  VDC_MIN_FIXED_POINT_WHOLE_VALUE..VDC_MAX_FIXED_POINT_WHOLE_VALUE ;
```

Type - VDC Fixed Point Whole Part Type
Purpose - Denotes the whole part definition of a CGI fixed point VDC real type.

```
type VDC_FIXED_POINT_REAL_TYPE is
  record
```

```
    WHOLE_PART     : VDC_FIXED_POINT_WHOLE_PART_TYPE ;
    FRACTION_PART : VDC_FIXED_POINT_FRACTION_PART_TYPE ;
  end record ;
```

Type - VDC Fixed Point Real Type
Purpose - Denotes the CGI fixed point VDC real type.

```
subtype VDC_FLOATING_POINT_REAL_TYPE is digits VDC_REAL_DIGITS ;
```

Subtype - VDC Floating Point Real Type
Purpose - Denotes the CGI VDC floating point real type.

```
type VDC_INTEGER_TYPE is range VDC_MIN_INTEGER..VDC_MAX_INTEGER ;
```

Type - VDC Integer Type
Purpose - Denotes the CGI VDC integer type.

```
type VDC_TYPE( VDC_TYPE : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    case VDC_TYPE is
      when INTEGER_VDC          => INTEGER_VDC : VDC_INTEGER_TYPE ;
      when FIXED_POINT_VDC      => FIXED_VDC   : VDC_FIXED_POINT_REAL_TYPE ;
      when FLOATING_POINT_VDC => FLOAT_VDC   : VDC_FLOATING_POINT_REAL_TYPE;
    end case ;
  end record ;
```

Type - VDC Type
Purpose - Virtual Device Coordinate in VDC space. Real or Integer depending on the VDC TYPE in effect. Subject to VDC INTEGER PRECISION or VDC REAL PRECISION within a data stream encoding as specified by VDC INTEGER PRECISION REQUIREMENT or VDC REAL PRECISION REQUIREMENTS.

Reference - ISO/IEC 9636-1 : VDC data type

6.3 Representation of CGI strings

A CGI string is a sequence of characters represented by values which are to be interpreted according to a set of character set and character coding announcer controls (input or output).

The basic unit for the CGI string will be an eight bit octet type. The octet type and the octet array type are defined as follows:

```
type OCTET_TYPE is range 0..( 2 ** 8 ) - 1 ;
```

Type - Octet Type
Purpose - Specifies an 8-bit basic unit of the CGI.

```
type OCTET_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of OCTET_TYPE ;
```

Type - Octet Array Type
Purpose - Specifies an array of Octet Types.

The CGI string as defined by this language binding will consist of a character count followed by an octet array containing the string data. Although a CGI string is defined as a sequence of characters, character sets and definitions may vary from country to country in both the number of octets necessary to represent one character and the definition of the characters themselves. It is the intent of this language binding that each CGI implementation provide a defined set of characters for a client's use or that in the case the client is using the Ada string type, a utility be provided to convert the Ada string to a CGI string and vica versa. The bindings for these utilities are presented in the Binding Defined Utility Functions clause of this standard.

The following declarations provide the definition of the CGI string:

```
subtype STRING_LENGTH_TYPE is ARRAY_INDEX_RANGE_TYPE
  range 0..MAX_STRING_LENGTH ;
```

Type - String Length Type
Purpose - Specifies a constraint for the definition of the CGI string. This constraint is determined from the client specified maximum string length.

```
subtype CHAR_SET_LENGTH_TYPE is ARRAY_INDEX_RANGE_TYPE
  range 0..MAX_CHAR_SET_LENGTH ;
```

Type - Character Set Length Type
Purpose - Specifies a constraint for the definition of the CGI character set string. This constraint is determined from the client specified maximum character set length.

```
subtype FIXED_STRING_LENGTH_TYPE is ARRAY_INDEX_RANGE_TYPE
  range 0..MAX_FIXED_STRING_LENGTH ;
```

Type - Fixed String Length Type
Purpose - Specifies a constraint for the definition of the CGI fixed string. This constraint is determined from the client specified maximum fixed string length.

```
type STRING_TYPE( LENGTH : STRING_LENGTH_TYPE := MAX_STRING_LENGTH ) is
  record
    CHARS_IN_STRING : STRING_LENGTH_TYPE := 0 ;
    STRING_DATA     : OCTET_ARRAY_TYPE( 1..LENGTH ) ;
  end record ;
```

Type - String Type
Purpose - Basic data type string. A sequence of characters represented by values which have to be interpreted according to a set of character set and character coding announcer controls (input or output). Note that a length of zero will specify a null string array.

Reference - ISO/IEC 9636-1 : S data type

```
type CHAR_SET_STRING_TYPE
  ( LENGTH : CHAR_SET_LENGTH_TYPE := MAX_CHAR_SET_LENGTH ) is
  record
    CHARS_IN_STRING : CHAR_SET_LENGTH_TYPE := 0 ;
    STRING_DATA     : OCTET_ARRAY_TYPE( 1..LENGTH ) ;
  end record ;
```

Type - Character String Type
Purpose - This type will be used in conjunction with a character set ID to define the ISO/IEC 9636 abstract character set (CS) data type. Note that a length of zero will specify a null string array.

```
type FIXED_STRING_TYPE
  ( LENGTH : FIXED_STRING_LENGTH_TYPE := MAX_FIXED_STRING_LENGTH ) is
  record
    CHARS_IN_STRING : FIXED_STRING_LENGTH_TYPE := 0 ;
    STRING_DATA     : OCTET_ARRAY_TYPE( 1..LENGTH ) ;
  end record ;
```

Type - Fixed String Type
Purpose - Basic data type string with fixed representation. Using the fixed character set (ISO 646) and not subject to CHARACTER CODING ANNOUNCER. (e.g. font names). Note that a length of 0 will specify a null string array.

Reference - ISO/IEC 9636-1 : SF data type

6.4 Representation of CGI data records

According to the ISO/IEC 9636-1, Data records for the CGI employ the basic data types, plus VC and VDC, and have internal structure. A data record is an ordered set of sub-sequences of parameters of a given data type. Each such sub-sequence consists of an identifier for the data type, a count of the number of parameters in the sub-sequence, and the list of parameters itself. Any data types may be used for any sub-sequence and different sub-sequences may use the same data type.

An implementation may wish to define the data record type in a variety of ways. For example, an implementation which is not concerned with memory usage may wish to bind the data record type as a variant record with components for each of the 13 varying data types. An implementation which generates a data stream may wish to bind the data record type to a representation which closely matches the selected encoding protocol requirements.

Since utility functions are provided for the data record type in the Data Record Utilities package, it should never be necessary for the client to know the internal structure of the data record type. For this reason, the data record type will be specified by this binding as an Ada private type. In this way, an implementation may define the data record type as it best sees fit. It will be necessary, however, to package all CGI data record types within the data record utilities package in order that the utility functions may access the data record implementation.

---

type DATA_RECORD_TYPE is private ;

Type - Data Record Type

Purpose - Basic Data Type Data Record. Data records for the CGI employ basic data types defined here, plus VC and VDC, and have internal structure. A data record is an ordered set of sub-sequences of parameters of a given type. Each such sub-sequence consists of an identifier for the data type, a count of the number of parameters in the sub-sequence, and the list of parameters itself. Any data types may be used for any sub-sequence and different sub-sequences may use the same data type. In data stream encodings, type (for VDC) and precisions apply in the same manner as for normal parameters of a given data type. For VC, viewport specification mode applies.

Reference - ISO/IEC 9636-1 : D data type

6.5 Representation of CGI abstract data types

This subclause contains the Ada mapping to the abstract CGI data types. The Abstract data types listed within this part are all dependent upon the previously defined Basic CGI data types. These type definitions are represented as indicated within the ISO/IEC 9636-1.

The abstract CGI types may vary with the current selection mode controls in effect. For example, the Colour Specifier type may be indexed or direct as indicated by the Colour Selection Mode in effect. These types have been mapped to variant records within this language binding. Since the valid parts of these variants may change throughout the CGI session, it is the intention of this Ada Language Binding that all variants be defaulted. The selected defaults for these variants will be determined according to the various specification modes listed within the ISO/IEC 9636. The first part of this subclause will define the enumerations which will be used for the variant parts of these records.

Types which are abstractly defined from the Fixed Integer type will be declared with the constraints imposed in the CGI configuration package. Since the fixed integer data type may be selectable within an implementation, the precision of these types may vary from binding to binding.

These type definitions have been listed in alphabetical order.

CGI Abstract Data Type Variant Declarations

```
type COLOUR_SELECTION_MODE_TYPE is ( INDEXED, DIRECT ) ;
```

Type - Colour Selection Mode Type
Purpose - Determines whether colour will be specified as indexed or direct.

```
type SPECIF_MODE_TYPE is ( VDC, SCALED ) ;
```

Type - Specif Mode Type
Purpose - Determines whether variables of type SIZE_SPECIF will be applied a measure in VDC or scaled mode.

```
type VIEWPORT_MODE_TYPE is
  ( INTEGER_VIEWPORT, FIXED_POINT_VIEWPORT, FLOATING_POINT_VIEWPORT ) ;
```

Type - Viewport Mode Type
Purpose - Determines whether the device viewport will be accessed (via viewport coordinates) as an integer or real viewport.

CGI Abstract Data Types

---

type ASN_TYPE is new CSN_TYPE ;

Type - ASN Type
Purpose - Attribute set name for a set of attribute values being SAVED or RESTORED.

Reference - ISO/IEC 9636-1 : ASN data type

---

type BITMAP_ID_TYPE is new CSN_TYPE ;

Type - Bitmap ID Type
Purpose - Bitmap identifier for built-in or client-defined bitmap.

Reference - ISO/IEC 9636-1 : BN data type

The definition of the character set type is dependent upon the following enumeration for the character set ID. This declaration has been included here as opposed to the Enumerated Data subclause of this document for clarity.

```
type CHAR_SET_ID_TYPE is ( CHAR_SET_94,
                           CHAR_MULTIBYTE_SET_94,
                           CHAR_SET_96,
                           CHAR_MULTIBYTE_SET_96,
                           COMPLETE_CODE ) ;
```

Type - Char Set ID Type
Purpose - Defines the valid character set identifiers.

```
type CHAR_SET_TYPE
  ( LENGTH : CHAR_SET_LENGTH_TYPE := MAX_CHAR_SET_LENGTH ) is
  record
    CHAR_SET_ID     : CHAR_SET_ID_TYPE ;
    CHAR_SET_STRING : CHAR_SET_STRING_TYPE( LENGTH ) ;
  end record ;
```

Type - Char Set Type
Purpose - Character set type(E) and the designator sequence tail string(S). The symbolic names for character set type are (94 CHARACTER SET, 94 CHARACTER MULTIBYTE SET, 96 CHARACTER SET, 96 CHARACTER MULTIBYTE SET, COMPLETE CODE). The tail string is composed of characters extracted from registered designating escape sequences. (See annex F of Part 3 in the ISO/IEC 9636-1). This string is not subject to code extension techniques.

Reference - ISO/IEC 9636-1 : CS data type

```
type COLOUR_SPECIFIER_TYPE
  ( COLOUR_SELECTION_MODE : COLOUR_SELECTION_MODE_TYPE := INDEXED ) is
  record
    case COLOUR_SELECTION_MODE is
      when DIRECT   => DIRECT_COLOUR : DIRECT_COLOUR_VALUE_TYPE;
      when INDEXED => INDEX_COLOUR   : COLOUR_INDEX_TYPE ;
    end case ;
  end record ;
```

Type - Colour Specifier Type
Purpose - Colour specifier in the relevant COLOUR SELECTION MODE and subject to the corresponding precision, or, for some functions, subject to local colour precision.

Reference - ISO/IEC 9636-1 : CO data type

```
type DEVICE_COORD_TYPE is range MIN_DEVICE_COORD..MAX_DEVICE_COORD ;
```

Type - Device Coord Type
Purpose - Defines the device coordinate in native device units in the physical device coordinate system.

Reference - ISO/IEC 9636-1 : DC data type

```
type DEVICE_POINT_TYPE is
  record
    X : DEVICE_COORD_TYPE ;
    Y : DEVICE_COORD_TYPE ;
  end record ;
```

Type - Device Point Type
Purpose - A pair of DC values which represent in native device units the x and y coordinates of a point in the physical device coordinate system.

Reference - ISO/IEC 9636-1 : DP data type

```
type ERROR_ID_TYPE is
  record
    ERROR_CLASS : ERROR_CLASS_TYPE ;
    ERROR_NUM   : INTRINSIC_NAME_TYPE ;
  end record ;
```

Type - Error ID Type
Purpose - Error identifier. A composite type consisting of the error class (IN) and the error number (IN) within the class as specified by the ISO/IEC 9636-1 (if positive) or by private use (if negative).

Reference - ISO/IEC 9636-1 : EI data type

```
type LOST_ERROR_COUNT_TYPE is range 2..MAX_LOST_ERROR_COUNT ;
```

Type - Lost Errors Count Type
Purpose - This type will be used to specify the number of lost errors for the error report type when the reports lost variant is true. The minimum range has been set to 2 since the number of lost errors can never be less than 2.

```
type ERROR_REPORT_TYPE( REPORTS_LOST : BOOLEAN := FALSE ) is
  record
    ERROR_ID : ERROR_ID_TYPE ;
    case REPORTS_LOST is
      when TRUE  => LOST_ERROR_COUNT    : LOST_ERROR_COUNT_TYPE ;
      when FALSE => FUNCTION_FOR_ERROR : FUNCTION_ID_TYPE ;
    end case ;
  end record ;
```

Type - Error Report Type
Purpose - Error report. A composite type consisting of an error identifier (EI) followed by either a count of errors lost (IF) for ERROR REPORTS LOST error, or a function identifier (IN) for all other errors.

Reference - ISO/IEC 9636-1 : ER data type

```
subtype EVENT_REPORTS_LIST_TYPE is ACCESS_DATA_RECORD_TYPE ;
```

Subtype - Event Reports List Type
Purpose - Defines the Event reports list (a sequence of events returned as a data record).

Reference - ISO/IEC 9636-1 : EVL data type

```
type EVENT_TYPE(
  INPUT_CLASS : CGI_TYPES.INPUT_CLASS_TYPE := CGI_TYPES.LOCATOR_LID ) is
  record
    LID_INDEX       : INDEX_TYPE ;
    TIMESTAMP       : REAL_TYPE ;
    TRIGGER         : INDEX_TYPE ;
    MEASURE_VALIDITY : VALIDITY_FLAG_TYPE ;
    case INPUT_CLASS is
      when LOCATOR_LID   => LOCATOR_MEASURE   : VDC_POINT_TYPE ;
      when STROKE_LID    => STROKE_MEASURE    : ACCESS_POINT_LIST_TYPE ;
      when VALUATOR_LID => VALUATOR_MEASURE : REAL_TYPE ;
      when CHOICE_LID    => CHOICE_MEASURE    : INTEGER_TYPE ;
      when PICK_LID      => PICK_MEASURE      : ACCESS_PICK_VALUE_ARRAY_TYPE ;
      when STRING_LID    => STRING_MEASURE    : ACCESS_STRING_TYPE ;
      when RASTER_LID    => XCOUNT            : INTEGER_TYPE ;
```

```
                                YCOUNT                  : INTEGER_TYPE ;
                                RASTER_MEASURE          : ACCESS_INPUT_COLOUR_ARRAY_TYPE ;
        when GENERAL_LID    => GENERAL_MEASURE          : ACCESS_DATA_RECORD_TYPE ;
    end case ;
  end record ;
```

Type - Event Type
Purpose - Event. A record of returned input data, the structure of which depends on the input class associated with the event.

Reference - ISO/IEC 9636-1 : EV data type

```
type FUNCTION_ID_TYPE is new INTRINSIC_NAME_TYPE ;
```

Type - Function ID Type
Purpose - Function identifier. Values are specified by Annex A of the ISO/IEC 9636-1 to identify standardized functions. Negative values are specified by implementors to identify extensions (private functions).

Reference - ISO/IEC 9636-1 : FN data type

```
type INPUT_SURFACE_COORD_TYPE is range
  MIN_INPUT_SURFACE_COORD..MAX_INPUT_SURFACE_COORD ;
```

Type - Input Surface Coord Type
Purpose - Defines the input surface coordinate.

Reference - ISO/IEC 9636-1 : ISC data type

```
type INPUT_SURFACE_POINT_TYPE is
  record
    X : INPUT_SURFACE_COORD_TYPE ;
    Y : INPUT_SURFACE_COORD_TYPE ;
  end record ;
```

Type - Input Surface Point Type
Purpose - Specifies a pair of ISC values giving the x and y coordinates of a point in the input coordinate space.

Reference - ISO/IEC 9636-1 : ISP data type

```
type INTRINSIC_NAME_TYPE is range
  MIN_INTRINSIC_NAME..MAX_INTRINSIC_NAME ;
```

Type - Intrinsic Name Type

Purpose - Intrinsic name. Specified by the ISO/IEC 9636-1 or the implementation. Each use of this type is always within a context which defines with which set of entries it is concerned.

Reference - ISO/IEC 9636-1 : IN data type

```
type PICK_ID_TYPE is new CSN_TYPE ;
```

Type - Pick ID Type
Purpose - Defines the pick identifier type.

Reference - ISO/IEC 9636-1 : PN data type

```
type PICK_VALUE_TYPE is
  record
    SEGMENT_ID : SEGMENT_ID_TYPE ;
    PICK_ID    : PICK_ID_TYPE ;
  end record ;
```

Type - Pick Value Type
Purpose - Pick value. A composite value consisting of a segment identifier (SN) followed by a pick identifier (PN).

Reference - ISO/IEC 9636-1 : PV data type

```
type PROFILE_ID_TYPE is new INTRINSIC_NAME_TYPE ;
```

Type - Profile ID Type
Purpose - Defines the profile identifier type.

Reference - ISO/IEC 9636-1 : PRN data type

```
type SEGMENT_ID_TYPE is new CSN_TYPE ;
```

Type - Segment ID Type
Purpose - Defines the segment identifier type.

Reference - ISO/IEC 9636-1 : SN data type

```
type SIZE_SPECIF_TYPE( SPECIF_MODE : SPECIF_MODE_TYPE := VDC ) is
  record
    case SPECIF_MODE is
      when VDC      => VDC_SIZE      : VDC_TYPE ;
      when SCALED => SCALED_SIZE : REAL_TYPE ;
    end case ;
```

```
  end record ;
```

Type - Size Specif Type
Purpose - Size Specification. A real or VDC depending on the particular specification mode. Context determines whether this is for edge width, line width, or marker size and therefore which of the specification modes is relevant.

Reference - ISO/IEC 9636-1 : SS data type

```
type VDC_POINT_TYPE( VDC_TYPE : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    case VDC_TYPE is
      when INTEGER_VDC =>
        INTEGER_DATA  : VDC_INTEGER_PAIR_TYPE ;
      when FIXED_POINT_VDC =>
        FIXED_DATA    : VDC_FIXED_PAIR_TYPE ;
      when FLOATING_POINT_VDC =>
        FLOAT_DATA    : VDC_FLOAT_PAIR_TYPE ;
    end case ;
  end record ;
```

Type - VDC Point Type
Purpose - Specifies a pair of VDC values giving the x and y coordinates of a point in VDC space.

Reference - ISO/IEC 9636-1 : P data type

```
type VIEWPORT_COORD_TYPE
  ( VIEWPORT_TYPE : VIEWPORT_MODE_TYPE := INTEGER_VIEWPORT ) is
  record
    case VIEWPORT_TYPE is
      when INTEGER_VIEWPORT =>
        INTEGER_VIEWPORT : INTEGER_TYPE ;
      when FIXED_POINT_VIEWPORT =>
        FIXED_VIEWPORT    : FIXED_POINT_REAL_TYPE ;
      when FLOATING_POINT_VIEWPORT =>
        FLOAT_VIEWPORT    : FLOATING_POINT_REAL_TYPE ;
    end case ;
  end record ;
```

Type - Viewport Coord Type
Purpose - Viewport coordinate in viewport space. Real or integer depending on DEVICE VIEWPORT SPECIFICATION MODE.

Reference - ISO/IEC 9636-1 : VC data type

```
type VIEWPORT_POINT_TYPE
```

```
( VIEWPORT_TYPE : VIEWPORT_MODE_TYPE := INTEGER_VIEWPORT ) is
record
  case VIEWPORT_TYPE is
    when INTEGER_VIEWPORT =>
      INTEGER_DATA  : INTEGER_PAIR_TYPE ;
    when FIXED_POINT_VIEWPORT =>
      FIXED_DATA    : FIXED_PAIR_TYPE ;
    when FLOATING_POINT_VIEWPORT =>
      FLOAT_DATA    : FLOAT_PAIR_TYPE ;
  end case ;
end record ;
```

Type - Viewport Point Type
Purpose - A pair of VC values giving x and y coordinates of a point in viewport space.

Reference - ISO/IEC 9636-1 : VP data type

6.6 Representation of CGI enumerated data types

The following defines the various CGI Enumerated Data types.

Basic data type, enumerated (Reference - ISO/IEC 9636-1 : E data type). Can take any value from a finite prescribed set for the particular instance as determined by the context. The set is specified by listing the identifiers which denote the members.

---

type ACTION_REQUIRED_FLAG_TYPE is (NO_ACTION_REQUIRED, ACTION_REQUIRED) ;

Type - Action Required Flag Type
Purpose - Notification to a graphics device that an action is or is not required for message data.

```
type ASF_TYPE is ( LINE_TYPE_ASF,
                   LINE_WIDTH_ASF,
                   LINE_COLOUR_ASF,
                   MARKER_TYPE_ASF,
                   MARKER_SIZE_ASF,
                   MARKER_COLOUR_ASF,
                   TEXT_FONT_INDEX_ASF,
                   TEXT_PREC_ASF,
                   CHAR_EXPAN_FACTOR_ASF,
                   CHAR_SPACING_ASF,
                   TEXT_COLOUR_ASF,
                   INTERIOR_STYLE_ASF,
                   FILL_COLOUR_ASF,
                   HATCH_INDEX_ASF,
                   PATTERN_INDEX_ASF,
                   EDGE_TYPE_ASF,
                   EDGE_WIDTH_ASF,
                   EDGE_COLOUR_ASF,
                   EDGE_VISIBILITY_ASF,
                   FILL_BITMAP_ASF ) ;
```

Type - ASF Type
Purpose - Defines the various Aspect Source Flags for the CGI.

---

type ASF_VALUE_TYPE is (INDIVIDUAL, BUNDLED) ;

Type - ASF Value Type
Purpose - Defines the two possible values of an Aspect Source Flag.

---

type BACKGROUND_COLOUR_CAP_TYPE is (NONE, INDEX_0, INDEXED, FULL) ;

Type - Background Colour Cap Type
Purpose - Specifies the background colour capabilities of a graphics device.

---

type BLOCK_CONTROL_TYPE is (NOT_BLOCKED, BLOCKED) ;

Type - Block Control Type
Purpose - Specifies the block control for the CGI event queue.

---

type BITMAP_MODE_TYPE is (INDEXED, DIRECT, MIXED) ;

Type - Bitmap Mode Type
Purpose - Specifies the colour mode supported for a particular bitmap.

---

type BITMAP_TRUNCATION_TYPE is (NOT_TRUNCATED, TRUNCATED) ;

Type - Bitmap Truncation Support Type
Purpose - Specifies the truncation supported by a raster device when rendering bitmaps.

Type BUNDLE_TABLE_TYPE is (LINE, MARKER, TEXT, FILL, EDGE) ;

Type - Bundle Table Type
Purpose - Defines the various bundle tables provided for in the CGI

---

type CELL_ARRAY_ALIGNMENT_CAP_TYPE is (AXIS_ALIGNED, SKEWED) ;

Type - Cell Array Alignment Cap Type
Purpose - Specifies the alignment type of cell array objects for a graphics device.

---

type CELL_ARRAY_FILL_CAP_TYPE is (OUTLINED, FILLED) ;

Type - Cell Array Fill Cap Type
Purpose - Specifies the fill capabilities for cell array objects.

---

type CELL_ARRAY_TECHNIQUE_TYPE is (OTHER, EXACT) ;

Type - Cell Array Technique Type
Purpose - Specifies the rendering technique which will be applied for cell array objects by a graphics device.

---

type CLEAR_DRAWING_SURFACE_TYPE is (UNCONDITIONAL, CONDITIONAL) ;

Type - Clear Drawing Surface Type

Purpose - Specifies the selections for clearing the drawing surface.

---

type CLIP_EFFECTIVENESS_TYPE is (NONE, CIRCUMSCRIBED, FULL) ;

Type - Clip Effectiveness Type
Purpose - Specifies the transformation clip effectiveness.

---

type CLIP_INHERITANCE_FILTER_TYPE is (STATE_LIST, INTERSECTION) ;

Type - Clip Inheritance Filter Type
Purpose - Specifies whether the effective clip region associated with an object in a copied segment is ignored or intersected with the clip rectangle.

---

type CLIP_MODE_TYPE is (LOCUS, SHAPE, LOCUS_THEN_SHAPE) ;

Type - Clip Mode Type
Purpose - Specifies the valid clipping modes for graphical objects.

---

type CLOSE_TYPE is (PIE, CHORD) ;

Type - Close Type
Purpose - Defines the manner in which fill primitives may be closed.

---

type CODING_TECHNIQUE_TYPE is (BASIC_7_BIT,
BASIC_8_BIT,
EXTENDED_7_BIT,
EXTENDED_8_BIT,
PRIVATE_CODING) ;

Type - Coding Technique Type
Purpose - Specifies the character coding announcers for a CGI string.

---

type COLOUR_REALIZATION_TYPE is (ADDITIVE, SUBTRACTIVE) ;

Type - Colour Realization Type
Purpose - Specifies the colour realization of a graphics device.

---

type COLOUR_SELECTION_MODE_AVAIL_TYPE is (INDEXED_ONLY, INDEXED_AND_DIRECT) ;

Type - Colour Selection Mode Avail Type
Purpose - Specifies the colour selection mode availabilities.

type COMPOUND_OBJECT_CAP_TYPE is (NONE, GLOBAL, LOCAL) ;

Type - Compound Object Cap Type
Purpose - Specifies the capability for compound objects by a graphics device.

---

type DEFERRAL_MODE_TYPE is (ASTI, BNI, ASAP) ;

Type - Deferral Mode Type
Purpose - Specifies the three CGI deferral buffer modes.

---

type DEPTH_TYPE is (MAPPED, FULL_DEPTH) ;

Type - Depth Type
Purpose - Specifies the allowed depths for CGI bitmaps.

---

type DEPTHS_SUPPORTED_TYPE is (FULL_DEPTH, MAPPED_AND_FULL_DEPTH) ;

Type - Depths Supported Type
Purpose - Specifies the depth of bitmaps supported by a raster device.

---

type DETECTABILITY_TYPE is (UNDETECTABLE, DETECTABLE) ;

Type - Detectability Type
Purpose - Specifies the detectability attribute of a segment.

---

type DEVICE_CLASS_TYPE is (OUTPUT, INPUT, OUTIN) ;

Type - Device Class Type
Purpose - Specifies the valid CGI graphical device classes.

---

type DISABLED_ENABLED_FLAG_TYPE is (DISABLED, ENABLED) ;

Type - Disabled Enabled Flag Type
Purpose - Provides an enabled/disabled indication for various CGI functions.

---

type DISPLAYABILITY_TYPE is (NON_DISPLAYABLE, DISPLAYABLE) ;

Type - Displayability Type
Purpose - Specifies the allowed displayability for CGI bitmaps.

type DISPLAY_BITMAP_DATA_TYPE is (CLEARED, PRESERVED) ;

Type - Display Bitmap Data Type
Purpose - Specifies previous display bitmap data for a raster device.

---

type DISPLAY_TYPE is (VECTOR, RASTER, OTHER) ;

Type - Display Type
Purpose - Specifies the display type for a CGI device.

---

type DRAWING_MODE_SUPPORT_TYPE is (NONE, SOME, ALL_MODES) ;

Type - Drawing Mode Support Type
Purpose - Specifies the drawing modes which are supported by a raster device.

---

type DRAWING_SURFACE_CLIP_INDICATOR_TYPE is (DSCRECT, VIEWPORT, OFF) ;

Type - Drawing Surface Clip Indicator Type
Purpose - Specifies the mode of drawing surface clipping for a graphics device.

---

type DRAWING_SURFACE_STATE_TYPE is (DIRTY, CLEAN) ;

Type - Drawing Surface State Type
Purpose - Denotes the state of the drawing surface.

---

type DYN_MOD_FLAG_TYPE is (IRG, CBS, IMM) ;

Type - Dyn Mod Flag Type
Purpose - Determines how modification through regeneration will be performed by a graphics device.

---

type ECHO_ENTITY_ECHO_CONTROL_TYPE is (ECHO_OFF, ECHO_ON) ;

Type - Echo Entity Echo Control Type
Purpose - Specifies the echo control for a remote echo.

---

type ECHO_ENTITY_PROMPT_CONTROL_TYPE is (PROMPT_OFF, PROMPT_ON) ;

Type - Echo Entity Prompt Control Type
Purpose - Specifies the prompt control for a remote echo.

---

type ECHO_ENTITY_STATE_TYPE is (READY, ACTIVE) ;

Type - Echo Entity Echo Control Type
Purpose - Specifies the echo control for a remote echo.

Type EDGE_OUT_FLAG_TYPE is (INVISIBLE, VISIBLE, CLOSE_INVISIBLE, CLOSE_VISIBLE) ;

Type - Edge Out Flag Type
Purpose - Specifies point to point edges for polygon sets.

---

type EDGE_VISIBILITY_TYPE is (VISIBLE, INVISIBLE) ;

Type - Edge Visibility Type
Purpose - Specifies the edge visibility for fill objects.

---

type EDGE_WIDTH_REALIZATION_TYPE is (INTERIOR, CENTRED) ;

Type - Edge Width Realization Type
Purpose - Specifies the edge width realization for fill objects.

---

type EFFECTIVE_PRIORITY_TYPE is (OTHER, TIME_ORDER_REOPEN, TIME_ORDER) ;

Type - Effective Priority Type
Purpose - Specifies effective relative display priority support.

---

type ERROR_HANDLING_FLAG_TYPE is (ON, REPORTING_OFF, DETECTION_OFF) ;

Type - Error Handling Flag Type
Purpose - Specifies the available error handling levels.

---

type EVENT_QUEUE_STATE_TYPE is (RELEASED,
EMPTY_NO_LID_ENABLED,
NOT_EMPTY_NO_LID_ENABLED,
EMPTY_LID_ENABLED,
NOT_EMPTY_LID_ENABLED) ;

Type - Event Queue State Type
Purpose - Specifies the states of the CGI event queue.

---

type EVENT_STATUS_TYPE is (TIMEOUT, EVENT, OVERFLOW, BREAK) ;

Type - Event Status Type
Purpose - Specifies the returned status type for event input.

```
type FILTER_SELECTION_TYPE is ( LINE_TYPE_ASF,
                                LINE_WIDTH_ASF,
                                LINE_COLOUR_ASF,
                                MARKER_TYPE_ASF,
                                MARKER_SIZE_ASF,
                                MARKER_COLOUR_ASF,
                                TEXT_FONT_INDEX_ASF,
                                TEXT_PREC_ASF,
                                CHAR_EXPANSION_FACTOR_ASF,
                                CHAR_SPACING_ASF,
                                TEXT_COLOUR_ASF,
                                INTERIOR_STYLE_ASF,
                                FILL_COLOUR_ASF,
                                HATCH_INDEX_ASF,
                                PATTERN_INDEX_ASF,
                                FILL_BITMAP_ASF,
                                EDGE_TYPE_ASF,
                                EDGE_WIDTH_ASF,
                                EDGE_COLOUR_ASF,
                                EDGE_VISIBILITY_ASF,
                                LINE_BUNDLE_INDEX,
                                LINE_TYPE,
                                LINE_WIDTH,
                                LINE_COLOUR,
                                LINE_CLIPPING_MODE,
                                MARKER_BUNDLE_INDEX,
                                MARKER_TYPE,
                                MARKER_SIZE,
                                MARKER_COLOUR,
                                MARKER_CLIPPING_MODE,
                                TEXT_BUNDLE_INDEX,
                                TEXT_FONT_INDEX,
                                TEXT_COLOUR,
                                CHAR_EXPANSION_FACTOR,
                                CHAR_SPACING,
                                CHAR_HEIGHT,
                                CHAR_ORIENTATION,
                                TEXT_PREC,
                                TEXT_PATH,
                                TEXT_ALIGNMENT,
                                FILL_BUNDLE_INDEX,
                                INTERIOR_STYLE,
                                FILL_COLOUR,
                                HATCH_INDEX,
                                PATTERN_INDEX,
```

FILL_BITMAP,
EDGE_BUNDLE_INDEX,
EDGE_TYPE,
EDGE_WIDTH,
EDGE_COLOUR,
EDGE_VISIBILITY,
EDGE_CLIPPING_MODE,
FILL_REFERENCE_POINT,
PATTERN_SIZE,
AUXILIARY_COLOUR,
TRANSPARENCY,
DRAWING_MODE,
PICK_ID,
LINE_ASFS,
MARKER_ASFS,
TEXT_ASFS,
FILL_ASFS,
EDGE_ASFS,
ALL_ASFS,
LINE_ATTRIBUTES,
MARKER_ATTRIBUTES,
LOCAL_TEXT_ATTRIBUTES,
GLOBAL_TEXT_ATTRIBUTES,
FILL_ATTRIBUTES,
EDGE_ATTRIBUTES,
PATTERN_ATTRIBUTES,
OUTPUT_CONTROL,
PICK_ATTRIBUTES,
ALL_ATTRIBUTES,
ALL_ATTRIBUTES_AND_ASFS) ;

Type - Filter Selection Type
Purpose - Specifies the filter selection values of the inheritance filter.

---

type FINAL_FLAG_TYPE is (NOT_FINAL, FINAL) ;

Type - Final Flag Type
Purpose - Specifies continuation status for CGI functions.

---

type HARD_SOFT_COPY_TYPE is (HARD, SOFT) ;

Type - Hard Soft Copy Class Type
Purpose - Specifies the drawing surface of a CGI device.

---

type HIGHLIGHTING_TYPE is (NORMAL, HIGHLIGHTED) ;

Type - Highlighting Type
Purpose - Specifies the highlighting attribute of a segment.

```
type HOR_ALIGNMENT_FLAG_TYPE is ( LEFT, CENTRE, RIGHT ) ;
```

Type - Hor Alignment Flag Type
Purpose - Used to specify the horizontal alignment for the graphics device's viewport.

```
type HOR_ALIGNMENT_TYPE is ( NORMAL_HOR, LEFT, CENTRE, RIGHT, CONTINUOUS_HOR ) ;
```

Type - Hor Alignment Type
Purpose - Used to select/set a graphics device's horizontal alignment.

```
type IMPLICIT_SEGMENT_REGEN_MODE_TYPE is ( SUPPRESSED, UQUM, ALLOWED ) ;
```

Type - Implicit Segment Regen Mode Type
Purpose - Provides the definition of segment regeneration.

```
type INHERITANCE_FILTER_TYPE is ( STATE_LIST, SEGMENT ) ;
```

Type - Inheritance Filter Type
Purpose - Specifies whether individual attribute values of graphic objects within a copied segment are reassociated from the CGI state lists or the source segment.

```
type INPUT_CLASS_TYPE is ( LOCATOR_LID,
                           STROKE_LID,
                           VALUATOR_LID,
                           CHOICE_LID,
                           PICK_LID,
                           STRING_LID,
                           RASTER_LID,
                           GENERAL_LID ) ;
```

Type - Input Class Type
Purpose - Defines the enumerated type which represents all CGI logical input devices.

```
type INPUT_DEVICE_STATE_TYPE is ( RELEASED,
                                  READY,
                                  REQUEST_PENDING,
                                  ECHO_REQUEST_ENABLED,
                                  ECHO_REQUEST_PENDING,
                                  ECHO_REQUEST_COMPLETED,
                                  EVENTS_ENABLED ) ;
```

Type - Input Device State Type
Purpose - Specifies the input state for an LID.

---

type INQUIRY_SOURCE_FLAG_TYPE is (CURRENT, DEFAULT) ;

Type - Inquiry Source Flag Type
Purpose - Denotes the valid identifiers for the specification of state list inquiry data.

---

type INTERIOR_STYLE_TYPE is (HOLLOW, SOLID, PATTERN, HATCH, EMPTY, BITMAP) ;

Type - Interior Style Type
Purpose - Specifies the registered interior styles which may be used in the definition of a fill primitive.

---

type ISOTROPY_FLAG_TYPE is (NOT_FORCED, FORCED) ;

Type - Isotropy Flag Type
Purpose - Used to Specify the isotropy setting for the graphics device.

---

type LINE_EDGE_CONTINUITY_CAP_TYPE is (RESTART, CONTINUOUS, OTHER) ;

Type - Line Edge Continuity Cap Type
Purpose - Specifies the Line/Edge Continuity Capability for a graphics device.

---

type ON_OFF_FLAG_TYPE is (OFF, ON) ;

Type - On Off Flag Type
Purpose - Provides an on/off indication for various CGI functions.

---

type OUTPUT_STATE_TYPE is (ACTIVE, TEXT_OPEN, FIGURE_OPEN) ;

Type - Output State Type
Purpose - Specifies the graphics device's output state in regard to primitive functions.

---

type PATTERN_FILL_FALLBACK_TYPE is (CONDENSE, TRUNCATE) ;

Type - Pattern Fill Fallback Type
Purpose - Specifies pattern fallback mode for fill objects.

---

type PATTERN_TRAN_SUPPORT_TYPE is (NONE, UNSKEWED, FULL) ;

Type - Pattern Tran Support Type

Purpose - Specifies pattern transformation support.

---

type PICK_APERTURE_SHAPE_TYPE is (RECTANGLE, CIRCLE, ELLIPSE) ;

Type - Pick Aperture Shape Type
Purpose - Specifies the shape of the pick LID aperture.

---

type PIXEL_RELATIVE_LOCATION_TYPE is (ON, BETWEEN) ;

Type - Pixel Relative Location Type
Purpose - Specifies the pixel/coordinate relationship for a CGI device.

---

type PIXEL_VALIDITY_FLAG_TYPE is (NONE, SOME, ALL_VALID) ;

Type - Pixel Validity Flag Type
Purpose - Specifies a general pixel validity status/indication for pixel array transfers.

---

type PROFILE_SUPPORT_FLAG_TYPE is (NO, YES, UNRECOGNIZED) ;

Type - Profile Support Flag Type
Purpose - Provides a indication about the level of support provided for the various CGI profiles by a particular device.

---

type REQUEST_STATUS_TYPE is (TRIGGER_FIRED, BREAK, TIMEOUT) ;

Type - Request Status Type
Purpose - Provides the values for requested event input for various CGI LID input functions.

---

type SEGMENT_OPEN_STATE_TYPE is (NO, YES, OVERFLOW) ;

Type - Segment open State Type
Purpose - Provides the definition of the segment state.

---

type SPOT_CENTRE_INTERPRETATION_TYPE is (NOMINAL, ACTUAL) ;

Type - Spot Centre Interpretation Type
Purpose - Specifies whether a raster device interprets raster spot centre separation as an actual or nominal value.

---

type SURFACE_INTERPRETATION_TYPE is (NOMINAL, ACTUAL, UNLIMITED) ;

Type - Surface Interpretation Type
Purpose - Specifies the size of a graphics device's input surface.

---

type TEXT_PATH_TYPE is (RIGHT, LEFT, UP, DOWN) ;

Type - Text Path Type
Purpose - Used to select/set a graphics device's text path.

---

type TEXT_PREC_TYPE is (STRING_PREC, CHAR_PREC, STROKE_PREC) ;

Type - Text Precision Type
Purpose - Used to select/set a graphics device's text precision.

---

type TIMEOUT_CAP_TYPE is (LIMITED_CAP, FULL_CAP) ;

Type - Timeout Capability Type
Purpose - Specifies the timeout capabilities for device input. The "_CAP" has been added to the type values so as not to conflict with the Ada reserved word "LIMITED".

---

type TIMESTAMP_SUPPORT_TYPE is (NA, TRIGGER_COUNT, CLOCK) ;

Type - Timestamp Support Type
Purpose - Specifies the timestamp capabilities for device input.

---

type TRANSPARENCY_TYPE is (OPAQUE, TRANSPARENT) ;

Type - Transparency Type
Purpose - Specifies the two CGI transparency modes.

---

type UNREPORTED_BREAK_STATE_TYPE is (NO_BREAK, BREAK) ;

Type - Unreported Break State Type
Purpose - Specifies the unreported break state for the CGI event queue.

---

type UNREPORTED_OVERFLOW_STATE_TYPE is (NO_OVERFLOW, OVERFLOW) ;

Type - Unreported Overflow State Type
Purpose - Specifies the unreported overflow state for the CGI event queue.

---

type VALIDITY_FLAG_TYPE is (INVALID, VALID) ;

Type - Validity Flag Type
Purpose - Used to determine whether information contained within a parameter list is valid.

```
type VDC_DIRECTION_TYPE is ( INCREASING_VDC, DECREASING_VDC ) ;
```

Type - VDC Direction Type
Purpose - Specifies the direction for pixel array description (rendering).

```
type VDC_SELECTION_TYPE is ( INTEGER_VDC, REAL_VDC ) ;
```

Type - VDC Selection Type
Purpose - Specifies the selection mode for objects represented as data type VDC within a CGI generator/interpreter implementation.

```
type VDC_SUPPORT_TYPE is ( INTEGER_VDC, REAL_VDC, BOTH ) ;
```

Type - VDC Support Type
Purpose - Specifies the VDC support available from a graphics device.

```
type VERT_ALIGNMENT_FLAG_TYPE is ( BOTTOM, CENTRE, TOP ) ;
```

Type - Vert Alignment Flag Type
Purpose - Used to specify the vertical alignment for the graphics device's viewport.

```
type VERT_ALIGNMENT_TYPE is ( NORMAL_VERT,
                              TOP,
                              CAP,
                              HALF,
                              BASE,
                              BOTTOM,
                              CONTINUOUS_VERT ) ;
```

Type - Vert Alignment Type
Purpose - Used to select/set a graphics device's vertical alignment.

```
type VISIBILITY_TYPE is ( INVISIBLE, VISIBLE ) ;
```

Type - Visibility Type
Purpose - Specifies the visibility attribute of a segment.

```
type VIEWPORT_SPECIF_MODE_TYPE is ( FRACTION_OF_DRAWING_SURFACE,
                                    MILLIMETRES_WITH_SCALE_FACTOR,
```

PHYSICAL_DEVICE_COORDS) ;

Type - Viewport Specif Mode Type
Purpose - Used to Specify the graphics device's virtual coordinate representation.

---

type YES_NO_FLAG_TYPE is (NO, YES) ;

Type - Yes No Flag Type
Purpose - Provides a yes/no indication for various CGI functions.

6.7 CGI Ada record types

The following subclause defines record structures which are used throughout the CGI Ada Language binding. These record structures gain their necessity in the description of various input and output parameter definitions specified throughout the ISO/IEC 9636.

```
type ARC_OFFSET_POINT_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    X_START : VDC_TYPE( SELECTOR ) ;
    Y_START : VDC_TYPE( SELECTOR ) ;
    X_END   : VDC_TYPE( SELECTOR ) ;
    Y_END   : VDC_TYPE( SELECTOR ) ;
  end record ;
```

Type - Arc Offset Point Type
Purpose - Defines x and y offsets for circular arcs.

```
type ASF_RECORD_TYPE is
  record
    ASF       : ASF_TYPE ;
    ASF_VALUE : ASF_VALUE_TYPE ;
  end record ;
```

Type - ASF Record Type
Purpose - Specifies an Aspect Source Flag and its associated value.

```
type BITBLT_DIMENSION_TYPE is
  record
    LENGTH : INTEGER_TYPE ;
    WIDTH  : INTEGER_TYPE ;
  end record ;
```

Type - BITBLT Dimension Type
Purpose - Specifies a BITBLT pattern size in length and width where length and width are specified in pixels.

```
type BITMAP_EXTENT_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    CORNER          : VDC_POINT_TYPE( SELECTOR ) ;
    OPPOSITE_CORNER : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;
```

Type - Bitmap Extent Type
Purpose - Specifies a rectangular bitmap extent in VDC space.

```
type BITMAP_REGION_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    POINT_1 : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_2 : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;
```

Type - Bitmap Region Type
Purpose - Specifies a rectangular bitmap region.

```
type CELL_ARRAY_DIMENSION_TYPE is
  record
    X_DIMENSION : INTEGER_TYPE ;
    Y_DIMENSION : INTEGER_TYPE ;
  end record ;
```

Type - Cell Array Dimension Type
Purpose - Specifies a cell array in which x denotes the number of colours per row and y denotes the number of rows.

```
type CELL_ARRAY_PARALLELOGRAM_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    POINT_P : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_Q : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_R : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;
```

Type - Cell Array Parallelogram Type
Purpose - Specifies a parallelogram which defines a cell array.

```
type CHAR_VECTOR_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    X_COMPONENT : VDC_TYPE( SELECTOR ) ;
    Y_COMPONENT : VDC_TYPE( SELECTOR ) ;
  end record ;
```

Type - Char Vector Type
Purpose - Specifies a vector type to be used for character orientation.

```
type CHAR_ORIENTATION_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    CHAR_UP_VECTOR   : CHAR_VECTOR_TYPE( SELECTOR ) ;
    CHAR_BASE_VECTOR : CHAR_VECTOR_TYPE( SELECTOR ) ;
  end record ;
```

Type - Char Orientation Type
Purpose - Denotes a character orientation by specifying the two character vectors. This type is used for character orientation inquiries.

```
type CIE_CHROMATICITY_TYPE( SELECTOR : REAL_MODE_TYPE := FLOATING_POINT ) is
  record
    U : REAL_TYPE( SELECTOR ) ;
    V : REAL_TYPE( SELECTOR ) ;
    Y : REAL_TYPE( SELECTOR ) ;
  end record ;
```

Type - CIE Chromaticity Type
Purpose - Denotes chromaticity coordinates for a 1931 CIE colour value.

```
type CIE_TRISTIMULUS_VALUE_TYPE
  ( SELECTOR : REAL_MODE_TYPE := FLOATING_POINT ) is
  record
    X : REAL_TYPE( SELECTOR ) ;
    Y : REAL_TYPE( SELECTOR ) ;
    Z : REAL_TYPE( SELECTOR ) ;
  end record ;
```

Type - CIE Tristimulus Value Type
Purpose - Denotes a tristimulus reference value for a 1931 CIE colour specifier.

```
type CLIP_RECTANGLE_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    CORNER          : VDC_POINT_TYPE( SELECTOR ) ;
    OPPOSITE_CORNER : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;
```

Type - Clip Rectangle Type
Purpose - Specifies a clip rectangle by defining two opposite corner points.

```
type COLOUR_VALUE_EXTENT_TYPE is
  record
    MIN_COLOUR_VALUE  : DIRECT_COLOUR_VALUE_TYPE ;
    MAX_COLOUR_VALUE : DIRECT_COLOUR_VALUE_TYPE ;
  end record ;
```

Type - Colour Value Extent Type
Purpose - Specifies the min and max colour values which define the colour value extent for a graphics device.

```
type DEVICE_VIEWPORT_TYPE( SELECTOR : VIEWPORT_MODE_TYPE := INTEGER_VIEWPORT ) is
```

```
  record
    CORNER          : VIEWPORT_POINT_TYPE( SELECTOR ) ;
    OPPOSITE_CORNER : VIEWPORT_POINT_TYPE( SELECTOR ) ;
  end record ;
```

Type - Device Viewport Type
Purpose - Specifies a rectangular viewport in viewport coordinate space for device viewport mapping.

```
type DRAWING_MODE_SPECIFIER_TYPE is
  record
    DRAWING_MODE_CLASS : INDEX_TYPE ;
    DRAWING_OPERATION  : INDEX_TYPE ;
  end record ;
```

Type - Drawing Mode Specifier Type
Purpose - Specifies a bitmap drawing mode (mode, class).

```
type DRAWING_MODE_TRANSPARENCY_TYPE is
  record
    DRAWING_MODE : DRAWING_MODE_SPECIFIER_TYPE ;
    TRANSPARENCY : TRANSPARENCY_TYPE ;
  end record ;
```

Type - Drawing Mode Transparency Type
Purpose - Specifies drawing mode/transparency pair used to specify the level of drawing mode/transparency support for a graphics device.

```
type DSCRECT_TYPE( SELECTOR : VIEWPORT_MODE_TYPE := INTEGER_VIEWPORT ) is
  record
    CORNER          : VIEWPORT_POINT_TYPE( SELECTOR ) ;
    OPPOSITE_CORNER : VIEWPORT_POINT_TYPE( SELECTOR ) ;
  end record ;
```

Type - Drawing Surface Clip Rectangle Type
Purpose - Specifies a drawing surface clip rectangle by defining two opposite corner points.

```
type ECHO_AREA_TYPE( SELECTOR : VIEWPORT_MODE_TYPE := INTEGER_VIEWPORT ) is
  record
    POINT_1 : VIEWPORT_POINT_TYPE( SELECTOR ) ;
    POINT_2 : VIEWPORT_POINT_TYPE( SELECTOR ) ;
  end record ;
```

Type - Echo Area Type
Purpose - Defines an echo area for a logical input device.

```
type ERROR_CONTROL_TYPE is
  record
    ERROR_CLASS          : ERROR_CLASS_TYPE ;
    ERROR_HANDLING_FLAG : ERROR_HANDLING_FLAG_TYPE ;
  end record ;
```

Type - Error Control Type
Purpose - Specifies an error handling control element (class/handling) pair.

```
type EXTENT_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    DELTA_WIDTH  : VDC_TYPE( SELECTOR ) ;
    DELTA_HEIGHT : VDC_TYPE( SELECTOR ) ;
  end record ;
```

Type - Extent Type
Purpose - Specifies a text extent within which text may be rendered.

```
type FIXED_PAIR_TYPE is
  record
    X : FIXED_POINT_REAL_TYPE ;
    Y : FIXED_POINT_REAL_TYPE ;
  end record ;
```

Type - Fixed Pair Type
Purpose - Specifies a pair of fixed point real values giving the x and y coordinates of a point in viewport space.

```
type FLOAT_PAIR_TYPE is
  record
    X : FLOATING_POINT_REAL_TYPE ;
    Y : FLOATING_POINT_REAL_TYPE ;
  end record ;
```

Type - Float Pair Type
Purpose - Specifies a pair of floating point real values giving the x and y coordinates of a point in viewport space.

```
type INHERITANCE_VALUE_TYPE is
  record
    FILTER_SELECTION   : FILTER_SELECTION_TYPE ;
    SELECTION_SETTING : INHERITANCE_FILTER_TYPE ;
  end record ;
```

Type - Inheritance Value Type

Purpose - Specifies a two part inheritance filter value. The first part of this value specifies the Inheritance Filter Selection as defined by the FILTER_SELECTION_TYPE. The second part is the actual setting for the FILTER_SELECTION, either STATE_LIST or SEGMENT.

```
type INPUT_EXTENT_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    FIRST_CORNER  : VDC_POINT_TYPE( SELECTOR ) ;
    SECOND_CORNER : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;
```

Type - Input Extent Type
Purpose - Specifies two opposite corners which define the input extent of an LID.

```
type INPUT_VIEWPORT_TYPE is
  record
    POINT_1 : INPUT_SURFACE_POINT_TYPE ;
    POINT_2 : INPUT_SURFACE_POINT_TYPE ;
    POINT_3 : INPUT_SURFACE_POINT_TYPE ;
  end record ;
```

Type - Input Viewport Type
Purpose - Specifies the input viewport parallelogram for an LID.

```
type INTEGER_PAIR_TYPE is
  record
    X : INTEGER_TYPE ;
    Y : INTEGER_TYPE ;
  end record ;
```

Type - Integer Pair Type
Purpose - Specifies a pair of integer values giving the x and y coordinates of a point in viewport space.

```
type PATTERN_DIMENSION_TYPE is
  record
    ROWS    : INTEGER_TYPE ;
    COLUMNS : INTEGER_TYPE ;
  end record ;
```

Type - Pattern Dimension Type
Purpose - Specifies a pattern size in rows and columns.

```
type PATTERN_SIZE_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
```

```
    HEIGHT_VECTOR : PATTERN_VECTOR_TYPE( SELECTOR ) ;
    WIDTH_VECTOR  : PATTERN_VECTOR_TYPE( SELECTOR ) ;
  end record ;
```

Type - Pattern Size Type
Purpose - Specifies the height and width vectors for pattern sizing within a fill object.

```
type PATTERN_VECTOR_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    X_COMPONENT : VDC_TYPE( SELECTOR ) ;
    Y_COMPONENT : VDC_TYPE( SELECTOR ) ;
  end record ;
```

Type - Pattern Vector Type
Purpose - Specifies a pattern vector for pattern sizing within a fill object.

```
type PHYSICAL_INPUT_SURFACE_SIZE_TYPE is
  ( SELECTOR : REAL_MODE_TYPE := FLOATING_POINT )
  record
    POINT_1 : REAL_TYPE( SELECTOR ) ;
    POINT_2 : REAL_TYPE( SELECTOR ) ;
  end record ;
```

Type - Physical Input Surface Size Type
Purpose - Specifies a physical input surface in millimetres for an LID.

```
type PICK_APERTURE_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    HEIGHT : VDC_TYPE( SELECTOR ) ;
    WIDTH  : VDC_TYPE( SELECTOR ) ;
  end record ;
```

Type - Pick Aperture Type
Purpose - Specifies an aperture in height and width for a pick LID.

```
type PIXEL_DIMENSION_TYPE( SELECTOR : REAL_MODE_TYPE := FLOATING_POINT ) is
  record
    HEIGHT : REAL_TYPE( SELECTOR ) ;
    WIDTH  : REAL_TYPE( SELECTOR ) ;
  end record ;
```

Type - Pixel Dimension Type
Purpose - Specifies a pixel size for a raster device by its height and width in millimetres.

```
type PROFILE_SUPPORT_TYPE is
  record
    PROFILE  : PROFILE_ID_TYPE ;
    SUPPORT : YES_NO_FLAG_TYPE ;
  end record ;
```

Type - Profile Support Type
Purpose - Specifies the level of support for the indicated profile.

```
type RASTER_WINDOW_CORNER_TYPE is
  record
    X : INTEGER_TYPE ;
    Y : INTEGER_TYPE ;
  end record ;
```

Type - Raster Window Corner Type
Purpose - Specifies a corner of a raster source window.

```
type RECTANGLE_DESCRIPTOR_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    CORNER          : VDC_POINT_TYPE( SELECTOR ) ;
    OPPOSITE_CORNER : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;
```

Type - Rectangle Descriptor Type
Purpose - Specifies a rectangular box in VDC space.

```
type SEGMENT_TRANSFORMATION_TYPE is
  record
    SCALING_ROTATION_PORTION : SCALING_ROTATION_MATRIX_TYPE ;
    TRANSLATION_PORTION      : TRANSLATION_MATRIX_TYPE ;
  end record ;
```

Type - Segment Transformation Type
Purpose - Denotes the transformation of a segment by specifying the segment's rotation, scaling, and translation.

```
type SPOT_CENTRE_SEPARATION_TYPE( SELECTOR : REAL_MODE_TYPE := FLOATING_POINT ) is
  record
    X_SEPARATION : REAL_TYPE( SELECTOR ) ;
    Y_SEPARATION : REAL_TYPE( SELECTOR ) ;
  end record ;
```

Type - Spot Centre Separation Type
Purpose - Specifies the spot centre separation for a raster input device in millimetres.

```
type STROKE_SAMPLING_INTERVAL_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    X_DISPLACEMENT_INTERVAL : VDC_TYPE( SELECTOR ) ;
    Y_DISPLACEMENT_INTERVAL : VDC_TYPE( SELECTOR ) ;
  end record ;
```

Type - Stroke Sampling Interval Type
Purpose - Denotes the sampling displacement for x and y of a stroke logical input device.

```
type TEXT_EXTENT_PARALLELOGRAM_TYPE
  ( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    POINT_1 : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_2 : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_3 : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_4 : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;
```

Type - Text Extent Parallelogram Type
Purpose - Specifies a parallelogram within which text would be rendered. The four corner points of the parallelogram are specified in an anti-clockwise order.

```
type VALID_PIXEL_RANGE_TYPE is
  record
    STARTING_INDEX : INTEGER_TYPE ;
    ENDING_INDEX   : INTEGER_TYPE ;
  end record ;
```

Type - Valid Pixel Range Type
Purpose - Specifies a pixel validity range for pixel array transfers.

```
type VDC_EXTENT_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    CORNER          : VDC_POINT_TYPE( SELECTOR ) ;
    OPPOSITE_CORNER : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;
```

Type - VDC Extent Type
Purpose - Specifies a rectangular extent in VDC space for viewport mapping.

```
type VDC_FIXED_PAIR_TYPE is
  record
    X : VDC_FIXED_POINT_REAL_TYPE ;
    Y : VDC_FIXED_POINT_REAL_TYPE ;
```

```
  end record ;
```

Type - VDC Fixed Pair Type
Purpose - Specifies a pair of VDC fixed point real values giving the x and y coordinates of a point in VDC space. This intermediate type will be used to define the VDC point type.

```
type VDC_FLOAT_PAIR_TYPE is
  record
    X : VDC_FLOATING_POINT_REAL_TYPE ;
    Y : VDC_FLOATING_POINT_REAL_TYPE ;
  end record ;
```

Type - VDC Float Pair Type
Purpose - Specifies a pair of VDC floating point real values giving the x and y coordinates of a point in VDC space. This intermediate type will be used to define the VDC point type.

```
type VDC_INTEGER_PAIR_TYPE is
  record
    X : VDC_INTEGER_TYPE ;
    Y : VDC_INTEGER_TYPE ;
  end record ;
```

Type - VDC Integer Pair Type
Purpose - Specifies a pair of VDC integer values giving the x and y coordinates of a point in VDC space. This intermediate type will be used to define the VDC point type.

```
type XY_STEP_TYPE is
  record
    STEPS_IN_X : INTEGER_TYPE ;
    STEPS_IN_Y : INTEGER_TYPE ;
  end record ;
```

Type - XY Step Type
Purpose - Specifies the valid steps in the x and y direction for a pick LID.

6.8 CGI Ada subtypes

The following subclause defines subtypes which are used throughout the CGI Ada Language binding. These subtypes are used to constrain and uniquely identify various elements of the CGI.

```
subtype COLOUR_TABLE_RANGE_TYPE is COLOUR_INDEX_TYPE range
  FIRST_ENTRY_IN_COLOUR_TABLE..LAST_ENTRY_IN_COLOUR_TABLE ;
```

Subtype - Colour Table Range Type
Purpose - Specifies a range of colour index colour table entries. This range is sized based on the client's colour table range specifications.

```
subtype ERROR_CLASS_TYPE is INTRINSIC_NAME_TYPE range 1..9 ;
```

Subtype - Error Class Type
Purpose - Specifies the CGI error class range (1 - 9).

6.9 CGI Ada array types

The following subclause defines arrays which are used throughout the CGI Ada Language binding. These arrays are used to specify various lists of CGI Basic and Abstract data types.

```
subtype ASF_INDEX_TYPE is ARRAY_INDEX_TYPE range 1 .. 20
```

Subtype - ASF Index Type
Purpose - Specifies an array index range which is valid for aspect source flag array data. This subtype is used to declare ASF arrays whose size can be tailored by the CGI client. This type is constrained to provide for the 20 aspect source flags defined in the CGI.

```
type ASF_ARRAY_TYPE is array ( ASF_INDEX_TYPE range <> ) of ASF_TYPE ;
```

Type - ASF Array Type
Purpose - Specifies an array of aspect source flags which may be used when looking up aspect source flag information for a particular primitive attribute.

```
type ASF_RECORD_ARRAY_TYPE is array ( ASF_INDEX_TYPE range <> )
  of ASF_RECORD_TYPE ;
```

Type - ASF Record Array Type
Purpose - Specifies an array of Aspect Source Flag(s) and their associated value(s). This type will be used to set Aspect Source Flag identifiers.

```
type ASF_VALUE_ARRAY_TYPE is array ( ASF_INDEX_TYPE range <> )
  of ASF_VALUE_TYPE ;
```

Type - ASF Value Array Type
Purpose - Specifies an array of aspect source flag values (Individual or Bundled) which may be used when looking up aspect source flag information for a particular primitive attribute.

```
type ASN_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of ASN_TYPE ;
```

Type - ASN Array Type
Purpose - Specifies an array of attribute set name values.

```
type BITMAP_ID_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of BITMAP_ID_TYPE ;
```

Type - Bitmap ID Array Type
Purpose - Specifies an array of bitmap identifiers.

```
type BITMAP_MODE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..3 )
  of BITMAP_MODE_TYPE ;
```

Type - Bitmap Mode Array Type
Purpose - Specifies an array of the three available bitmap modes.

```
type BUNDLE_TABLE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..5 )
  of BUNDLE_TABLE_TYPE ;
```

Type - Bundle Table Array Type
Purpose - Specifies an array of bundle table/attribute groups.

```
type CHAR_CODING_ANNOUNCER_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..5 )
  of CODING_TECHNIQUE_TYPE ;
```

Type - Char Coding Announcer Array Type
Purpose - Specifies an array of character coding announcer.

```
type CHAR_SET_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of CHAR_SET_TYPE ;
```

Type - Char Set Array Type
Purpose - Used to define character set arrays for CGI functions.

```
type CLIP_MODE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..3 ) of CLIP_MODE_TYPE ;
```

Type - Clip Mode Array Type
Purpose - Specifies an array of the three available clipping modes.

```
type COLOUR_ARRAY_TYPE
  ( COLOUR_SELECTION_MODE : COLOUR_SELECTION_MODE_TYPE := INDEXED ;
    NUM_OF_ROWS           : ARRAY_INDEX_TYPE := ARRAY_INDEX_TYPE'LAST ;
    COLOURS_PER_ROW       : ARRAY_INDEX_TYPE := ARRAY_INDEX_TYPE'LAST ) is
  record
    case COLOUR_SELECTION_MODE is
      when INDEXED =>
        INDEX_COLOUR_DATA : INDEX_COLOUR_MATRIX_TYPE
          ( 1..NUM_OF_ROWS, 1..COLOURS_PER_ROW ) ;
      when DIRECT =>
        DIRECT_COLOUR_DATA : DIRECT_COLOUR_MATRIX_TYPE
          ( 1..NUM_OF_ROWS, 1..COLOURS_PER_ROW ) ;
    end case ;
  end record ;
```

Type - Colour Array Type
Purpose - Used to specify an (X x Y) array of pixel colour values, where Y indicates the number of colour rows and X indicates the number of colours per row.

```
type COLOUR_INDEX_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of COLOUR_INDEX_TYPE ;
```

Type - Colour Index Array Type
Purpose - Specifies an array of indices into the colour table of a graphics device.

```
type COLOUR_SPECIFIER_ARRAY_TYPE
  ( COLOUR_SELECTION_MODE : COLOUR_SELECTION_MODE_TYPE := INDEXED ;
    LENGTH                : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
  record
    case COLOUR_SELECTION_MODE is
      when INDEXED => INDEX_COLOUR_DATA  : COLOUR_INDEX_ARRAY_TYPE( 1..LENGTH ) ;
      when DIRECT  => DIRECT_COLOUR_DATA : DIRECT_COLOUR_ARRAY_TYPE( 1..LENGTH ) ;
    end case ;
  end record ;
```

Type - Colour Specifier Array Type
Purpose - Specifies an array of colour specifiers for inquiry functions.

```
type COLOUR_TABLE_TYPE is array ( COLOUR_TABLE_RANGE_TYPE range <> )
  of DIRECT_COLOUR_VALUE_TYPE ;
```

Type - Colour Table Type
Purpose - Specifies a table of direct colour values which may be used to load or request the contents of the graphics device's colour table. This table will be indexed by a colour index value.

```
type CSN_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of CSN_TYPE ;
```

Type - CSN Array Type
Purpose - Specifies an array of client specified name values.

```
type DATA_RECORD_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of DATA_RECORD_TYPE ;
```

Type - Data Record Array Type
Purpose - Specifies an array of data records.

```
type DEFERRAL_MODE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..3 )
  of DEFERRAL_MODE_TYPE ;
```

Type - Deferral Mode Array Type
Purpose - Specifies an array of deferral modes for description table inquiry functions.

```
type DEVICE_COORD_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of DEVICE_COORD_TYPE ;
```

Type - Device Coord Array Type
Purpose - Used to define variable device coordinate arrays for CGI functions.

```
type DIRECT_COLOUR_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of DIRECT_COLOUR_VALUE_TYPE ;
```

Type - Direct Colour Array Type
Purpose - Specifies an array of red-green-blue triple values.

```
type DIRECT_COLOUR_MATRIX_TYPE is array
  ( ARRAY_INDEX_TYPE range <>, ARRAY_INDEX_TYPE range <> )
  of DIRECT_COLOUR_VALUE_TYPE ;
```

Type - Direct Colour Matrix Array Type
Purpose - Used to define a matrix of row X column direct colour values.

```
type DIRECT_INPUT_COLOUR_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of DIRECT_INPUT_COLOUR_VALUE_TYPE ;
```

Type - Direct Input Colour Array Type
Purpose - Used to define a list of direct input colour values.

```
type DRAWING_MODE_TRANSPARENCY_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of DRAWING_MODE_TRANSPARENCY_TYPE ;
```

Type - Drawing Mode Transparency Array Type
Purpose - Specifies an array of Drawing Mode/Transparency pairs.

```
type EDGE_OUT_FLAG_ARRAY_TYPE is array ( ARRAY_INDEX_RANGE <> )
  of EDGE_OUT_FLAG_TYPE ;
```

Type - Edge Out Flag Array Type
Purpose - Specifies an array of edge out flags for a flagged point list of a polygon set.

```
type ERROR_HANDLING_ARRAY_TYPE is array ( ERROR_CLASS_TYPE range <> )
  of ERROR_CONTROL_TYPE ;
```

Type - Error Handling Array Type
Purpose - Defines a list of error class/handling pairs used for setting error handling control.

```
type ERROR_HANDLING_FLAG_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range 1..9 ) of ERROR_HANDLING_FLAG_TYPE ;
```

Type - Error Handling Flag Array Type
Purpose - Defines a list of the 9 error handling flags (one corresponding to each of the error handling classes).

```
type ERROR_QUEUE_TYPE is array
  ( ARRAY_INDEX_TYPE range 1..SIZE_OF_INTERPRETER_ERROR_QUEUE )
  of ERROR_REPORT_TYPE ;
```

Type - Error Queue Type
Purpose - Used to define the CGI error queue. The size of this queue is constrained to the client's specification in the CGI configuration file.

```
subtype FILTER_SELECTION_INDEX_TYPE is ARRAY_INDEX_TYPE range 1 .. 75
```

Subtype - Filter Selection Index Type
Purpose - Specifies an array index range which is valid for filter selection array data. This subtype is used to declare the filter selection list in order that its size may be tailored by the CGI client. This type is constrained to provide for the 75 inheritance filter selection values defined in the CGI.

```
type FILTER_SELECTION_LIST_TYPE is array
  ( FILTER_SELECTION_INDEX_TYPE range <> ) of FILTER_SELECTION_TYPE ;
```

Type - Filter Selection List Type
Purpose - Specifies an array of inheritance filter values for setting the inheritance filter of a graphics device.

```
type FIXED_INTEGER_8_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FIXED_INTEGER_8_TYPE ;
```

Type - Fixed Integer 8 Array Type
Purpose - Specifies an array of 8-bit fixed integer values.

```
type FIXED_INTEGER_16_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FIXED_INTEGER_16_TYPE ;
```

Type - Fixed Integer 16 Array Type
Purpose - Specifies an array of 16-bit fixed integer values.

```
type FIXED_INTEGER_32_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FIXED_INTEGER_32_TYPE ;
```

Type - Fixed Integer 32 Array Type
Purpose - Specifies an array of 32-bit fixed integer values.

```
type FIXED_POINT_REAL_ARRAY_TYPE is array ( ARRAY_INDEX_RANGE <> )
  of FIXED_POINT_REAL_TYPE ;
```

Type - Fixed Point Real Array Type
Purpose - Denotes an array of CGI fixed point real values.

```
type FIXED_STRING_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FIXED_STRING_TYPE ;
```

Type - Fixed String Array Type
Purpose - Used to define variable fixed string arrays for CGI functions.

```
type FLAGGED_POINT_LIST_TYPE
  ( VDC_TYPE : VDC_MODE_TYPE              := INTEGER_VDC ;
    LENGTH   : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
  record
    POINT_LIST : POINT_LIST_TYPE( VDC_TYPE, LENGTH ) ;
    FLAG_LIST  : EDGE_OUT_FLAG_ARRAY_TYPE( 1..LENGTH ) ;
  end record ;
```

Type - Flagged Point List Type
Purpose - Used to define a VDC flagged point list for a polygon set.

```
type FLOATING_POINT_REAL_ARRAY_TYPE is array
  ( ARRAY_INDEX_RANGE <> ) of FLOATING_POINT_REAL_TYPE ;
```

Type - Floating Point Real Array Type
Purpose - Denotes an array of CGI floating point real values.

```
type FUNCTION_ID_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FUNCTION_ID_TYPE ;
```

Type - Function ID Array Type
Purpose - Specifies an array of function identifiers which may be used to ascertain information about a particular device's configuration/setup.

```
type GENERATOR_ERROR_QUEUE_TYPE is array
```

```
  ( ARRAY_INDEX_TYPE range 1...SIZE_OF_GENERATOR_ERROR_QUEUE )
  of ERROR_REPORT_TYPE ;
```

Type - Generator Error Queue Type
Purpose - Used to define an error queue for the CGI generator/language binding. The size of this queue is constrained to the client's specification in the CGI configuration file.

```
type HOR_ALIGNMENT_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..5 )
  of HOR_ALIGNMENT_TYPE ;
```

Type - Hor Alignment Array Type
Purpose - Specifies an array of horizontal alignment specifiers to be used for description table inquiry functions.

```
type INDEX_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of INDEX_TYPE ;
```

Type - Index Array Type
Purpose - Used to define index arrays for CGI functions.

```
type INDEX_COLOUR_MATRIX_TYPE is array
  ( ARRAY_INDEX_TYPE range <>, ARRAY_INDEX_TYPE range <> )
  of COLOUR_INDEX_TYPE ;
```

Type - Index Colour Matrix Array Type
Purpose - Used to define a matrix row X column colour index values.

```
type INHERITANCE_VALUE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of INHERITANCE_VALUE_TYPE ;
```

Type - Inheritance Value Array Type
Purpose - Used to define an array of inheritance filter attribute designators.

```
type INPUT_COLOUR_INDEX_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of INPUT_COLOUR_INDEX_TYPE ;
```

Type - Input Colour Index Array Type
Purpose - Used to define a list of indexed input colour values.

```
type INPUT_COLOUR_ARRAY_TYPE
  ( COLOUR : YES_NO_FLAG_TYPE           := NO ;
    LENGTH : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
  record
    case COLOUR is
      when NO  =>
```

```
        INDEX_COLOUR_DATA   : INPUT_COLOUR_INDEX_ARRAY_TYPE( 1..LENGTH ) ;
      when YES =>
        DIRECT_COLOUR_DATA : DIRECT_INPUT_COLOUR_ARRAY_TYPE( 1..LENGTH ) ;
    end case ;
  end record ;
```

Type - Input Colour Array Type
Purpose - Used to specify an input colour array of for raster devices.

```
type INTEGER_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of INTEGER_TYPE ;
```

Type - Integer Array Type
Purpose - Used to define integer arrays for CGI functions.

```
subtype INTENSITY_ARRAY_TYPE is INTEGER_ARRAY_TYPE( 1..3 ) ;
```

Subtype - Intensity Array Type
Purpose - Specifies the intensities as they relate to the color tuple.

```
type INTERIOR_STYLE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..6 )
  of INTERIOR_STYLE_TYPE ;
```

Type - Interior Style Array Type
Purpose - Used to define the list of interior styles for fill inquiries.

```
type ORIENTATION_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of CHAR_ORIENTATION_TYPE ;
```

Type - Char Orientation Array Type
Purpose - Specifies an array of character orientations for inquiry functions.

```
type PICK_VALUE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of PICK_VALUE_TYPE ;
```

Type - Pick Value Array Type
Purpose - Specifies a variable length pick value array.

```
type POINT_LIST_TYPE
  ( VDC_TYPE  : VDC_MODE_TYPE              := INTEGER_VDC ;
    LENGTH    : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
    record
      case VDC_TYPE is
        when INTEGER_VDC =>
```

```
        INTEGER_DATA : VDC_INTEGER_PAIR_ARRAY_TYPE( 1..LENGTH ) ;
      when FIXED_POINT_VDC =>
        FIXED_DATA   : VDC_FIXED_PAIR_ARRAY_TYPE( 1..LENGTH ) ;
      when FLOATING_POINT_VDC =>
        FLOAT_DATA   : VDC_FLOAT_PAIR_ARRAY_TYPE( 1..LENGTH );
    end case ;
  end record ;
```

Type - Point List Type
Purpose - Defines a VDC Point list for graphical primitive functions.

```
type PREC_REQUIREMENT_TYPE is array ( ARRAY_INDEX_TYPE range 1..3 )
  of INTEGER_TYPE ;
```

Type - Prec Requirement Type
Purpose - Specifies local colour precision for the CGI colour specifiers within a pixel array.

```
type PROFILE_ID_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of PROFILE_ID_TYPE ;
```

Type - Profile ID Array Type
Purpose - Specifies an array of profile identifiers which may be used to ascertain information about a particular device's configuration/setup.

```
type PROFILE_SUPPORT_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of PROFILE_SUPPORT_TYPE ;
```

Type - Profile Support Array Type
Purpose - Specifies an array of profile support types which define the level of support provided by a graphics device for a particular profile.

```
type PROFILE_SUPPORT_FLAG_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of PROFILE_SUPPORT_FLAG_TYPE ;
```

Type - Profile Support Flag Array Type
Purpose - Specifies an array of profile support indicators which may be used to ascertain information about a particular device's configuration/setup.

```
type REAL_ARRAY_TYPE
  ( REAL_MODE : REAL_MODE_TYPE       := FLOATING_POINT ) ;
    LENGTH    : ARRAY_INDEX_RANGE := ARRAY_INDEX_RANGE'LAST ) is
  record
    case REAL_MODE is
      when FLOATING_POINT =>
```

```
        FLOAT_DATA : FLOATING_POINT_REAL_ARRAY_TYPE( 1..LENGTH ) ;
      when FIXED_POINT =>
        FIXED_DATA  : FIXED_POINT_REAL_ARRAY_TYPE( 1..LENGTH ) ;
    end case ;
  end record ;
```

Type - Real Array Type
Purpose - Used to define real arrays for CGI functions.

```
type SCALING_ROTATION_MATRIX_TYPE is array
  ( ARRAY_INDEX_TYPE range 1..2, ARRAY_INDEX_TYPE range 1..2 )
  of REAL_TYPE ;
```

Type - Scaling Rotation Matrix Type
Purpose - Specifies a 2x2 scaling and rotation matrix defined as follows:

(1,1) - Scaling in X
(1,2) - Scaling in Y
(2,1) - Rotation in X
(2,2) - Rotation in Y

```
type SEGMENT_ID_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of SEGMENT_ID_TYPE ;
```

Type - Segment ID Array Type
Purpose - Specifies an array of segment identifier values for segment inquiry functions.

```
type STRING_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of STRING_TYPE ;
```

Type - String Array Type
Purpose - Specifies an array of CGI strings.

```
type TEXT_PATH_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..4 ) of TEXT_PATH_TYPE ;
```

Type - Text Path Array Type
Purpose - Specifies an array of text path specifiers to be used for description table inquiry functions.

```
type TRANSLATION_MATRIX_TYPE is array ( ARRAY_INDEX_TYPE range 1..2 ) of VDC_TYPE ;
```

Type - Translation Matrix Type
Purpose - Specifies a 1x1 translation matrix where the first element denotes the translation in the X direction and the second element denotes the translation in the Y direction.

```
type VDC_ARRAY_TYPE
  ( VDC_TYPE : VDC_MODE_TYPE                := INTEGER_VDC ;
    LENGTH   : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
  record
    case VDC_TYPE is
      when INTEGER_VDC =>
        INTEGER_DATA : VDC_INTEGER_COORD_ARRAY_TYPE( 1..LENGTH ) ;
      when FIXED_POINT_VDC =>
        FIXED_DATA   : VDC_FIXED_COORD_ARRAY_TYPE( 1..LENGTH ) ;
      when FLOATING_POINT_VDC =>
        FLOAT_DATA   : VDC_FLOAT_COORD_ARRAY_TYPE( 1..LENGTH ) ;
    end case ;
  end record ;
```

Type - VDC Array Type
Purpose - Specifies an array of Virtual Device Coordinate values.

```
type VDC_FIXED_COORD_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_FIXED_POINT_REAL_TYPE ;
```

Type - VDC Fixed Coord Coord Array Type
Purpose - Specifies an array of VDC fixed point real values.

```
type VDC_FLOAT_COORD_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_FLOATING_POINT_REAL_TYPE ;
```

Type - VDC Integer Coord Array Type
Purpose - Specifies an array of VDC floating point real values.

```
type VDC_INTEGER_COORD_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_INTEGER_TYPE ;
```

Type - VDC Integer Coord Array Type
Purpose - Specifies an array of VDC integer values.

```
type VDC_FIXED_PAIR_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_FIXED_PAIR_TYPE ;
```

Type - VDC Fixed Pair Array Type
Purpose - Specifies an array of VDC fixed point real x,y coordinate values.

```
type VDC_FLOAT_PAIR_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_FLOAT_PAIR_TYPE ;
```

Type - VDC Float Pair Array Type
Purpose - Specifies an array of VDC floating point real x,y coordinate values.

```
type VDC_INTEGER_PAIR_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_INTEGER_PAIR_TYPE ;
```

Type - VDC Integer Pair Array Type
Purpose - Specifies an array of VDC integer x,y coordinate values.

```
type VERT_ALIGNMENT_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..7 )
  of VERT_ALIGNMENT_TYPE ;
```

Type - Vert Alignment Array Type
Purpose - Specifies an array of vertical alignment specifiers to be used for description table inquiry functions.

```
type VIEWPORT_COORD_ARRAY_TYPE
  ( VIEWPORT_TYPE : VIEWPORT_MODE_TYPE       := INTEGER_VIEWPORT ;
    LENGTH        : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
  record
    case VIEWPORT_TYPE is
      when INTEGER_VIEWPORT =>
        INTEGER_DATA : INTEGER_ARRAY_TYPE( 1..LENGTH ) ;
      when FIXED_POINT_VIEWPORT =>
        FIXED_DATA   : FIXED_POINT_REAL_ARRAY_TYPE( 1..LENGTH ) ;
      when FLOATING_POINT_VIEWPORT =>
        FLOAT_DATA   : FLOATING_POINT_REAL_ARRAY_TYPE( 1..LENGTH ) ;
    end case ;
  end record ;
```

Type - Viewport Coord Array Type
Purpose - Specifies an array of Viewport Coordinate values.

```
type VIEWPORT_SPECIF_MODE_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range 1..3 ) of VIEWPORT_SPECIF_MODE_TYPE ;
```

Type - Viewport Specif Mode Array Type
Purpose - Specifies an array of viewport specification modes for description table inquiry functions.

```
type YES_NO_FLAG_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of YES_NO_FLAG_TYPE ;
```

Type - Yes No Flag Array Type
Purpose - Used to define yes/no flag arrays for CGI functions.

6.10 CGI Ada access types

The following subclause defines the CGI access types which are employed within this Ada Language Binding.

```
type ACCESS_DATA_RECORD_TYPE is access DATA_RECORD ;
```

Type - Access Data Record Type
Purpose - Used as a pointer to a CGI data record.

```
type ACCESS_INPUT_COLOUR_ARRAY_TYPE is access INPUT_COLOUR_ARRAY_TYPE ;
```

Type - Access Input Colour Array Type
Purpose - Used as a pointer to an input array of raster colours.

```
type ACCESS_PICK_VALUE_ARRAY_TYPE is access PICK_VALUE_ARRAY_TYPE ;
```

Type - Access Pick Value Array Type
Purpose - Used as a pointer to an array of pick values.

```
type ACCESS_POINT_LIST_TYPE is access POINT_LIST_TYPE ;
```

Type - Access Point List Type
Purpose - Used as a pointer to a VDC point list.

```
type ACCESS_STRING_TYPE is access STRING_TYPE ;
```

Type - Access String Type
Purpose - Used as a pointer to a CGI string.

6.11 CGI exceptions

The following subclause defines the CGI exceptions which are employed within this Ada Language Binding.

CGI_ERROR : exception ;

Exception - CGI ERROR
Purpose - Specifies the exception which will be raised if an implementation detects a CGI error and the following conditions are true:

- The error handling state for the associated error class is set to REPORTING ON.
- The CGI Configuration value CGI Error Notification is set to TRUE.

The usage of this exception is controlled by the client via the CGI Error Notification Flag in the CGI_CONFIG package. If the client does not wish to be notified of error conditions via the CGI_ERROR exception, the CGI Error Notification Flag in the CGI_CONFIG package should be set to FALSE. If the CGI Error Notification Flag is set to false, errors which occur during the execution of the CGI will be stored in the CGI error queue and the client will not be notified. This is in keeping with the CGI philosophy that all error processing will be controlled synchronously by the client.

---

7 CGI/Ada functions

7.1 Part 2 control functions

Function INITIALIZE

```
procedure INITIALIZE_SESSION ;
```

NOTE - This procedure's name has been changed from its original name "INITIALIZE" as specified in the ISO/IEC 9636 to "INITIALIZE_SESSION" in order to maintain consistence with the TERMINATE function.

Function TERMINATE

```
procedure TERMINATE_SESSION ;
```

NOTE - This procedure's name has been changed from its original name "TERMINATE" as specified in the ISO/IEC 9636 to "TERMINATE_SESSION" so as not to cause conflict with the Ada reserved word terminate.

Function EXECUTE DEFERRED ACTIONS

```
procedure EXECUTE_DEFERRED_ACTIONS ;
```

Function DEFERRAL_MODE

```
procedure SET_DEFERRAL_MODE(
  DEFERRAL_MODE : in DEFERRAL_MODE_TYPE ) ;
```

Function PREPARE DRAWING SURFACE

```
procedure PREPARE_DRAWING_SURFACE(
  CLEAR_DRAWING_SURFACE : in CLEAR_DRAWING_SURFACE_TYPE ) ;
```

Function END PAGE

```
procedure END_PAGE ;
```

Function VDC TYPE

```
procedure SET_VDC_TYPE(
  VDC_TYPE_SELECTOR : VDC_SELECTION_TYPE ) ;
```

Function VDC INTEGER PRECISION REQUIREMENT

```
procedure SET_VDC_INTEGER_PREC_REQUIREMENT(
  LOG_2_MAGNITUDE_OF_UPPER_BOUND : in INTEGER_TYPE ) ;
```

Function VDC REAL PRECISION REQUIREMENTS

```
procedure SET_VDC_REAL_PREC_REQUIREMENTS(
  LOG_2_MAGNITUDE_OF_UPPER_BOUND          : in INTEGER_TYPE ;
  LOG_2_NON_ZERO_MAGNITUDE_MIN            : in INTEGER_TYPE ;
  LOG_2_RELATIVE_PREC_REQUIREMENT         : in INTEGER_TYPE ;
  LOG_2_MIN_MAGNITUDE_FOR_RELATIVE_PREC : in INTEGER_TYPE ;
  LOG_2_TYPICAL_MAGNITUDE                 : in INTEGER_TYPE ) ;
```

Function VDC EXTENT

```
procedure SET_VDC_EXTENT(
  VDC_EXTENT : in VDC_EXTENT_TYPE ) ;
```

Function DEVICE VIEWPORT

```
procedure SET_DEVICE_VIEWPORT(
  VIEWPORT : in DEVICE_VIEWPORT_TYPE ) ;
```

Function DEVICE VIEWPORT SPECIFICATION MODE

```
procedure SET_DEVICE_VIEWPORT_SPECIF_MODE(
  VC_MODE                : in VIEWPORT_SPECIF_MODE_TYPE ;
  METRIC_SCALE_FACTOR : in REAL_TYPE ) ;
```

Function DEVICE VIEWPORT MAPPING

```
procedure SET_DEVICE_VIEWPORT_MAPPING(
  ISOTROPY_FLAG          : in ISOTROPY_FLAG_TYPE ;
  HOR_ALIGNMENT_FLAG  : in HOR_ALIGNMENT_FLAG_TYPE ;
  VERT_ALIGNMENT_FLAG : in VERT_ALIGNMENT_FLAG_TYPE ) ;
```

Function DRAWING SURFACE CLIP RECTANGLE

```
procedure SET_DRAWING_SURFACE_CLIP_RECTANGLE(
```

DSCRECT : in DSCRECT_TYPE) ;

Function DRAWING SURFACE CLIP INDICATOR

```
procedure SET_DRAWING_SURFACE_CLIP_INDICATOR(
  DRAWING_SURFACE_CLIP_INDICATOR : in DRAWING_SURFACE_CLIP_INDICATOR_TYPE ) ;
```

Function DEQUEUE ERROR REPORTS

```
procedure DEQUEUE_ERROR_REPORTS(
  NUM_OF_REPORTS_REQUESTED   : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY          : out VALIDITY_FLAG_TYPE ;
  REPORTS_REMAINING_IN_QUEUE : out INTEGER_TYPE ;
  LIST_OF_ERROR_REPORTS      : out ERROR_QUEUE_TYPE ) ;
```

Function ERROR HANDLING CONTROL

```
procedure SET_ERROR_HANDLING_CONTROL(
  ERROR_CLASS_HANDLING_PAIRS : in ERROR_HANDLING_ARRAY_TYPE ) ;
```

Function INTEGER PRECISION REQUIREMENT

```
procedure SET_INTEGER_PREC_REQUIREMENT(
  LOG_2_MAGNITUDE_OF_UPPER_BOUND : in INTEGER_TYPE ) ;
```

Function REAL PRECISION REQUIREMENTS

```
procedure SET_REAL_PREC_REQUIREMENTS(
  LOG_2_MAGNITUDE_OF_UPPER_BOUND          : in INTEGER_TYPE ;
  LOG_2_NON_ZERO_MAGNITUDE_MIN            : in INTEGER_TYPE ;
  LOG_2_RELATIVE_PREC_REQUIREMENT         : in INTEGER_TYPE ;
  LOG_2_MIN_MAGNITUDE_FOR_RELATIVE_PREC : in INTEGER_TYPE ;
  LOG_2_TYPICAL_MAGNITUDE                 : in INTEGER_TYPE ) ;
```

Function INDEX PRECISION REQUIREMENT

```
procedure SET_INDEX_PREC_REQUIREMENT(
  LOG_2_MAGNITUDE_OF_UPPER_BOUND : in INTEGER_TYPE ) ;
```

Function COLOUR PRECISION REQUIREMENT

```
procedure SET_COLOUR_PREC_REQUIREMENT(
```

```
    LOG_2_MAGNITUDE_OF_UPPER_BOUND : in INTEGER_TYPE ) ;
```

Function COLOUR INDEX PRECISION REQUIREMENT

```
procedure SET_COLOUR_INDEX_PREC_REQUIREMENT(
    LOG_2_MAGNITUDE_OF_UPPER_BOUND : in INTEGER_TYPE ) ;
```

Function CLIENT SPECIFIED NAME PRECISION REQUIREMENT

```
procedure SET_CSN_PREC_REQUIREMENT(
    LOG_2_MAGNITUDE_OF_UPPER_BOUND : in INTEGER_TYPE ) ;
```

Function MESSAGE

```
procedure MESSAGE(
    ACTION_REQUIRED_FLAG : in ACTION_REQUIRED_FLAG_TYPE ;
    MESSAGE_TEXT         : in STRING_TYPE ) ;
```

Function ESCAPE

```
procedure ESCAPE(
    ESCAPE_FUNCTION_ID : in INTEGER_TYPE ;
    DATA_RECORD        : in DATA_RECORD_TYPE ) ;
```

Function GET ESCAPE

```
procedure GET_ESCAPE(
    ESCAPE_FUNCTION_ID    : in  INTEGER_TYPE ;
    DATA_RECORD           : in  DATA_RECORD_TYPE ;
    RESPONSE_VALIDITY     : out VALIDITY_FLAG_TYPE ;
    OUTPUT_DATA_RECORD    : out DATA_RECORD_TYPE ) ;
```

Function STATE LIST INQUIRY SOURCE

```
procedure SET_STATE_LIST_INQUIRY_SOURCE(
    SOURCE : in INQUIRY_SOURCE_FLAG_TYPE ) ;
```

Function INQUIRE DEVICE IDENTIFICATION

```
procedure INQ_DEVICE_ID(
    MAX_CHARS_PER_STRING : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY    : out VALIDITY_FLAG_TYPE ;
```

```
    DEVICE_CLASS              : out DEVICE_CLASS_TYPE ;
    DEVICE_IDENTIFICATION     : out STRING_TYPE ) ;
```

Function INQUIRE DEVICE DESCRIPTION

```
procedure INQ_DEVICE_DESCRIPTION(
    RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
    HARD_SOFT_COPY_CLASS                       : out HARD_SOFT_COPY_TYPE ;
    DISPLAY_TYPE                               : out DISPLAY_TYPE ;
    DYN_MOD_ACCEPTED_FOR_VDC_TO_DEV_MAP        : out DYN_MOD_FLAG_TYPE ;
    SPONTANEOUS_DISPLAY_CHANGE_POSSIBLE        : out YES_NO_FLAG_TYPE ;
    DISPLAY_SURFACE_BOTTOM_LEFT                : out DEVICE_POINT_TYPE ;
    DISPLAY_SURFACE_UPPER_RIGHT                : out DEVICE_POINT_TYPE ;
    DISPLAY_SURFACE_WIDTH                      : out REAL_TYPE ;
    DISPLAY_SURFACE_HEIGHT                     : out REAL_TYPE ;
    PIXEL_LOCATION_RELATIVE_TO_COORDS          : out PIXEL_RELATIVE_LOCATION_TYPE ) ;
```

Function LOOKUP FUNCTION SUPPORT

```
procedure LOOKUP_FUNCTION_SUPPORT(
    LIST_OF_FUNCTION_IDS          : in  FUNCTION_ID_ARRAY_TYPE ;
    RESPONSE_VALIDITY             : out VALIDITY_FLAG_TYPE ;
    LIST_OF_SUPPORT_INDICATORS    : out YES_NO_FLAG_ARRAY_TYPE ) ;
```

Function LOOKUP PROFILE SUPPORT

```
procedure LOOKUP_PROFILE_SUPPORT(
    LIST_OF_PROFILE_IDS           : in  PROFILE_ID_ARRAY_TYPE ;
    RESPONSE_VALIDITY             : out VALIDITY_FLAG_TYPE ;
    LIST_OF_SUPPORT_INDICATORS    : out PROFILE_SUPPORT_FLAG_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF PROFILE SUPPORT INDICATORS

```
procedure INQ_PROFILE_SUPPORT_INDICATORS(
    NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE   : out INTEGER_TYPE ;
    LIST_OF_PROFILE_SUPPORT_INDICATORS    : out PROFILE_SUPPORT_ARRAY_TYPE ) ;
```

Function INQUIRE SUPPORTED VDC TYPES

```
procedure INQ_SUPPORTED_VDC_TYPES(
    RESPONSE_VALIDITY : out VALIDITY_FLAG_TYPE ;
```

```
    VDC_TYPE_SUPPORT  : out VDC_SUPPORT_TYPE ) ;
```

Function INQUIRE DEVICE CONTROL CAPABILITY

```
procedure INQ_DEVICE_CONTROL_CAP(
   RESPONSE_VALIDITY                    : out VALIDITY_FLAG_TYPE ;
   VIEWPORT_SPECIF_MODE_ELEMENTS        : out INTEGER_TYPE ;
   VIEWPORT_SPECIF_MODE_ARRAY           : out VIEWPORT_SPECIF_MODE_ARRAY_TYPE;
   MESSAGE_ACTION_DETECTION_SUPPORT     : out YES_NO_FLAG_TYPE ;
   MAX_NUM_OF_CHARS_FOR_MESSAGE         : out INTEGER_TYPE ;
   DEVICE_VIEWPORT_MIRRORING_SUPPORT    : out YES_NO_FLAG_TYPE ;
   DEFERRAL_MODE_ELEMENTS               : out INTEGER_TYPE ;
   DEFERRAL_MODE_ARRAY                  : out DEFERRAL_MODE_ARRAY_TYPE ;
   SIZE_OF_ERROR_QUEUE                  : out INTEGER_TYPE ) ;
```

Function LOOKUP ESCAPE SUPPORT

```
procedure LOOKUP_ESCAPE_SUPPORT(
   LIST_OF_ESCAPE_FUNCTION_IDS : in  INTEGER_ARRAY_TYPE ;
   RESPONSE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
   LIST_OF_SUPPORT_INDICATORS  : out YES_NO_FLAG_ARRAY_TYPE ) ;
```

Function LOOKUP GET ESCAPE SUPPORT

```
procedure LOOKUP_GET_ESCAPE_SUPPORT(
   LIST_OF_GET_ESCAPE_FUNCTION_IDS : in  INTEGER_ARRAY_TYPE ;
   RESPONSE_VALIDITY               : out VALIDITY_FLAG_TYPE ;
   LIST_OF_SUPPORT_INDICATORS      : out YES_NO_FLAG_ARRAY_TYPE ) ;
```

Function INQUIRE CONTROL STATE

```
procedure INQ_CONTROL_STATE(
   RESPONSE_VALIDITY              : out VALIDITY_FLAG_TYPE ;
   DEVICE_DRAWING_SURFACE_STATE   : out DRAWING_SURFACE_STATE_TYPE ;
   DEFERRAL_MODE                  : out DEFERRAL_MODE_TYPE ;
   VDC_TYPE                       : out VDC_MODE_TYPE ;
   VIEWPORT_SPECIF_MODE           : out VIEWPORT_SPECIF_MODE_TYPE ;
   VIEWPORT_METRIC_SCALE_FACTOR   : out REAL_TYPE ;
   NUM_OF_QUEUED_ERROR_REPORTS    : out INTEGER_TYPE ) ;
```

Function INQUIRE CURRENT PRECISION REQUIREMENTS

```
procedure INQ_CURRENT_PREC_REQUIREMENTS(
   RESPONSE_VALIDITY                                  : out VALIDITY_FLAG_TYPE ;
```

```
   VDC_INTEGER_LOG2_MAGNITUDE_OF_UPPER_BOUND      : out INTEGER_TYPE ;
   VDC_REAL_LOG2_MAGNITUDE_OF_UPPER_BOUND         : out INTEGER_TYPE ;
   VDC_REAL_LOG2_NON_ZERO_MAGNITUDE_MIN           : out INTEGER_TYPE ;
   VDC_REAL_LOG2_RELATIVE_PREC_REQUIREMENT        : out INTEGER_TYPE ;
   VDC_REAL_LOG2_MIN_MAGNITUDE_FOR_RELATIVE_PREC  : out INTEGER_TYPE ;
   VDC_REAL_LOG2_TYPICAL_MAGNITUDE                : out INTEGER_TYPE ;
   INTEGER_LOG2_MAGNITUDE_OF_UPPER_BOUND          : out INTEGER_TYPE ;
   REAL_LOG2_MAGNITUDE_OF_UPPER_BOUND             : out INTEGER_TYPE ;
   REAL_LOG2_NON_ZERO_MAGNITUDE_MIN               : out INTEGER_TYPE ;
   REAL_LOG2_RELATIVE_PREC_REQUIREMENT            : out INTEGER_TYPE ;
   REAL_LOG2_MIN_MAGNITUDE_FOR_RELATIVE_PREC      : out INTEGER_TYPE ;
   REAL_LOG2_TYPICAL_MAGNITUDE                    : out INTEGER_TYPE ;
   INDEX_LOG2_MAGNITUDE_OF_UPPER_BOUND            : out INTEGER_TYPE ;
   COLOUR_LOG2_MAGNITUDE_OF_UPPER_BOUND           : out INTEGER_TYPE ;
   COLOUR_INDEX_LOG2_MAGNITUDE_OF_UPPER_BOUND     : out INTEGER_TYPE ;
   CSN_LOG2_MAGNITUDE_OF_UPPER_BOUND              : out INTEGER_TYPE ) ;
```

Function INQUIRE MISCELLANEOUS CONTROL STATE

```
procedure INQ_MISCELLANEOUS_CONTROL_STATE(
   RESPONSE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
   STATE_LIST_INQUIRY_SOURCE : out INQUIRY_SOURCE_FLAG_TYPE ) ;
```

Function INQUIRE VDC TO DEVICE MAPPING

```
procedure INQ_VDC_TO_DEVICE_MAPPING(
   RESPONSE_VALIDITY              : out VALIDITY_FLAG_TYPE ;
   VDC_EXTENT                     : out VDC_EXTENT_TYPE ;
   ISOTROPY                       : out ISOTROPY_FLAG_TYPE ;
   HOR_ALIGNMENT                  : out HOR_ALIGNMENT_FLAG_TYPE ;
   VERT_ALIGNMENT                 : out VERT_ALIGNMENT_FLAG_TYPE ;
   VIEWPORT_SPECIF_MODE           : out VIEWPORT_SPECIF_MODE_TYPE ;
   VIEWPORT_METRIC_SCALE_FACTOR : out REAL_TYPE ;
   REQUESTED_DEVICE_VIEWPORT      : out DEVICE_VIEWPORT_TYPE ;
   EFFECTIVE_VIEWPORT             : out DEVICE_VIEWPORT_TYPE ;
   SURFACE_CLIP_INDICATOR         : out DRAWING_SURFACE_CLIP_INDICATOR_TYPE;
   DSCRECT_SPECIF_MODE            : out VIEWPORT_SPECIF_MODE_TYPE ;
   DSCRECT_METRIC_SCALE_FACTOR    : out REAL_TYPE ;
   DSCRECT                        : out DEVICE_VIEWPORT_TYPE ) ;
```

Function INQUIRE ERROR HANDLING

```
procedure INQ_ERROR_HANDLING(
   RESPONSE_VALIDITY       : out VALIDITY_FLAG_TYPE ;
   ERROR_HANDLING_FLAGS : out ERROR_HANDLING_FLAG_ARRAY_TYPE ) ;
```

7.2 Part 3 output functions

Function POLYLINE

```
procedure POLYLINE(
  POINT_LIST  : in POINT_LIST_TYPE ;
  FINAL_FLAG : in FINAL_FLAG_TYPE := FINAL ) ;
```

Function DISJOINT POLYLINE

```
procedure DISJOINT_POLYLINE(
  POINT_LIST : in POINT_LIST_TYPE ) ;
```

Function CIRCULAR ARC 3 POINT

```
procedure CIRCULAR_ARC_3_POINT(
  STARTING_POINT       : in VDC_POINT_TYPE ;
  INTERMEDIATE_POINT : in VDC_POINT_TYPE ;
  ENDING_POINT         : in VDC_POINT_TYPE ) ;
```

Function CIRCULAR ARC CENTRE

```
procedure CIRCULAR_ARC_CENTRE(
  CENTRE_POINT       : in VDC_POINT_TYPE ;
  ARC_OFFSET_POINTS : in ARC_OFFSET_POINT_TYPE ;
  RADIUS             : in VDC_TYPE ) ;
```

Function CIRCULAR ARC CENTRE REVERSED

```
procedure CIRCULAR_ARC_CENTRE_REVERSED(
  CENTRE_POINT       : in VDC_POINT_TYPE ;
  ARC_OFFSET_POINTS : in ARC_OFFSET_POINT_TYPE ;
  RADIUS             : in VDC_TYPE ) ;
```

Function ELLIPTICAL ARC

```
procedure ELLIPTICAL_ARC(
  CENTRE_POINT                          : in VDC_POINT_TYPE ;
  FIRST_CONJUGATE_RADIUS_ENDPOINT    : in VDC_POINT_TYPE ;
  SECOND_CONJUGATE_RADIUS_ENDPOINT : in VDC_POINT_TYPE ;
```

```
    ARC_OFFSET_POINTS                    : in ARC_OFFSET_POINT_TYPE ) ;
```

Function CONNECTING EDGE

```
procedure CONNECTING_EDGE ;
```

Function POLYMARKER

```
procedure POLYMARKER(
   POINT_LIST : in POINT_LIST_TYPE ) ;
```

Function TEXT

```
procedure TEXT(
   POINT              : in VDC_POINT_TYPE ;
   TEXT_FINAL_FLAG : in FINAL_FLAG_TYPE ;
   TEXT_STRING        : in STRING_TYPE ) ;
```

Function RESTRICTED TEXT

```
procedure RESTRICTED_TEXT(
   EXTENT             : in EXTENT_TYPE ;
   POINT              : in VDC_POINT_TYPE ;
   TEXT_FINAL_FLAG : in FINAL_FLAG_TYPE ;
   TEXT_STRING        : in STRING_TYPE ) ;
```

Function APPEND TEXT

```
procedure APPEND_TEXT(
   TEXT_FINAL_FLAG : in FINAL_FLAG_TYPE ;
   TEXT_STRING        : in STRING_TYPE ) ;
```

Function POLYGON

```
procedure POLYGON(
   POINT_LIST   : in POINT_LIST_TYPE ;
   FINAL_FLAG : in FINAL_FLAG_TYPE := FINAL ) ;
```

Function POLYGON SET

```
procedure POLYGON_SET(
   FLAGGED_POINT_LIST : in FLAGGED_POINT_LIST_TYPE ) ;
```

Function RECTANGLE

```
procedure RECTANGLE(
  CORNER_POINTS : in RECTANGLE_DESCRIPTOR_TYPE ) ;
```

Function CIRCLE

```
procedure CIRCLE(
  CENTRE_POINT : in VDC_POINT_TYPE ;
   RADIUS      : in VDC_TYPE ) ;
```

Function CIRCULAR ARC 3 POINT CLOSE

```
procedure CIRCULAR_ARC_3_POINT_CLOSE(
  STARTING_POINT      : in VDC_POINT_TYPE ;
  INTERMEDIATE_POINT  : in VDC_POINT_TYPE ;
  ENDING_POINT        : in VDC_POINT_TYPE ;
  CLOSE_TYPE          : in CLOSE_TYPE ) ;
```

Function CIRCULAR ARC CENTRE CLOSE

```
procedure CIRCULAR_ARC_CENTRE_CLOSE(
  CENTRE_POINT       : in VDC_POINT_TYPE ;
  ARC_OFFSET_POINTS  : in ARC_OFFSET_POINT_TYPE ;
  RADIUS             : in VDC_TYPE ;
  CLOSE_TYPE         : in CLOSE_TYPE ) ;
```

Function ELLIPSE

```
procedure ELLIPSE(
  CENTRE_POINT                          : in VDC_POINT_TYPE ;
  FIRST_CONJUGATE_RADIUS_ENDPOINT       : in VDC_POINT_TYPE ;
  SECOND_CONJUGATE_RADIUS_ENDPOINT      : in VDC_POINT_TYPE ) ;
```

Function ELLIPTICAL ARC CLOSE

```
procedure ELLIPTICAL_ARC_CLOSE(
  CENTRE_POINT                          : in VDC_POINT_TYPE ;
  FIRST_CONJUGATE_RADIUS_ENDPOINT       : in VDC_POINT_TYPE ;
  SECOND_CONJUGATE_RADIUS_ENDPOINT      : in VDC_POINT_TYPE ;
  ARC_OFFSET_POINTS                     : in ARC_OFFSET_POINT_TYPE ;
  CLOSE_TYPE                            : in CLOSE_TYPE ) ;
```

Function CELL ARRAY

```
procedure CELL_ARRAY(
   CELL_ARRAY_PARALLELOGRAM        : in CELL_ARRAY_PARALLELOGRAM_TYPE ;
   DIMENSIONS                      : in CELL_ARRAY_DIMENSION_TYPE ;
   LOCAL_COLOUR_PREC_REQUIREMENT : in INTEGER_TYPE ;
   CELL_COLOUR_SPECIFIERS          : in COLOUR_ARRAY_TYPE ;
   FINAL_FLAG                      : in FINAL_FLAG_TYPE := FINAL ) ;
```

Function GENERALIZED DRAWING PRIMITIVE

```
procedure GENERALIZED_DRAWING_PRIMITIVE(
   GDP_ID          : in INTEGER_TYPE ;
   POINT_LIST      : in POINT_LIST_TYPE ;
   DATA_RECORD : in DATA_RECORD_TYPE ;
   FINAL_FLAG      : in FINAL_FLAG_TYPE := FINAL ) ;
```

Function LINE BUNDLE INDEX

```
procedure SET_LINE_BUNDLE_INDEX(
   LINE_BUNDLE_INDEX : in INDEX_TYPE ) ;
```

Function LINE TYPE

```
procedure SET_LINE_TYPE(
   LINE_TYPE : in INDEX_TYPE ) ;
```

Function LINE WIDTH

```
procedure SET_LINE_WIDTH(
   LINE_WIDTH_SPECIFIER : in SIZE_SPECIF_TYPE ) ;
```

Function LINE COLOUR

```
procedure SET_LINE_COLOUR(
   LINE_COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function LINE CLIPPING MODE

```
procedure SET_LINE_CLIPPING_MODE(
   CLIPPING_MODE : in CLIP_MODE_TYPE ) ;
```

Function MARKER BUNDLE INDEX

```
procedure SET_MARKER_BUNDLE_INDEX(
  MARKER_BUNDLE_INDEX : in INDEX_TYPE ) ;
```

Function MARKER TYPE

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

Function MARKER SIZE

```
procedure SET_MARKER_SIZE(
  MARKER_SIZE_SPECIFIER : in SIZE_SPECIF_TYPE ) ;
```

Function MARKER COLOUR

```
procedure SET_MARKER_COLOUR(
  MARKER_COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function MARKER CLIPPING MODE

```
procedure SET_MARKER_CLIPPING_MODE(
  CLIPPING_MODE : in CLIP_MODE_TYPE ) ;
```

Function TEXT BUNDLE INDEX

```
procedure SET_TEXT_BUNDLE_INDEX(
  TEXT_BUNDLE_INDEX : in INDEX_TYPE ) ;
```

Function TEXT FONT INDEX

```
procedure SET_TEXT_FONT_INDEX(
  FONT_INDEX : in INDEX_TYPE ) ;
```

Function TEXT PRECISION

```
procedure SET_TEXT_PREC(
  TEXT_PREC : in TEXT_PREC_TYPE ) ;
```

Function CHARACTER EXPANSION FACTOR

```
procedure SET_CHAR_EXPAN_FACTOR(
  CHAR_EXPAN_FACTOR : in REAL_TYPE ) ;
```

Function CHARACTER SPACING

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

Function TEXT COLOUR

```
procedure SET_TEXT_COLOUR(
  TEXT_COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function CHARACTER HEIGHT

```
procedure SET_CHAR_HEIGHT(
  CHAR_HEIGHT : in VDC_TYPE ) ;
```

Function CHARACTER ORIENTATION

```
procedure SET_CHAR_ORIENTATION(
  CHAR_UP_VECTOR    : in CHAR_VECTOR_TYPE ;
  CHAR_BASE_VECTOR : in CHAR_VECTOR_TYPE ) ;
```

Function TEXT PATH

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

Function TEXT ALIGNMENT

```
procedure SET_TEXT_ALIGNMENT(
  HOR_ALIGNMENT                : in HOR_ALIGNMENT_TYPE ;
  VERT_ALIGNMENT               : in VERT_ALIGNMENT_TYPE ;
  CONTINUOUS_HOR_ALIGNMENT  : in REAL_TYPE ;
  CONTINUOUS_VERT_ALIGNMENT : in REAL_TYPE ) ;
```

Function CHARACTER SET INDEX

```
procedure SET_CHAR_SET_INDEX(
  CHAR_SET_INDEX : in INDEX_TYPE ) ;
```

Function ALTERNATE CHARACTER SET INDEX

```
procedure SET_ALTERNATE_CHAR_SET_INDEX(
  ALTERNATE_CHAR_SET_INDEX : in INDEX_TYPE ) ;
```

Function CHARACTER CODING ANNOUNCER

```
procedure SET_CHAR_CODING_ANNOUNCER(
  CODING_TECHNIQUE : in CODING_TECHNIQUE_TYPE ) ;
```

Function FILL BUNDLE INDEX

```
procedure SET_FILL_BUNDLE_INDEX(
  FILL_BUNDLE_INDEX : in INDEX_TYPE ) ;
```

Function INTERIOR STYLE

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

Function FILL COLOUR

```
procedure SET_FILL_COLOUR(
  FILL_COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function HATCH INDEX

```
procedure SET_HATCH_INDEX(
  HATCH_INDEX : in INDEX_TYPE ) ;
```

Function PATTERN INDEX

```
procedure SET_PATTERN_INDEX(
  PATTERN_INDEX : in INDEX_TYPE ) ;
```

Function FILL REFERENCE POINT

```
procedure SET_FILL_REFERENCE_POINT(
  REFERENCE_POINT : in VDC_POINT_TYPE ) ;
```

Function PATTERN SIZE

```
procedure SET_PATTERN_SIZE(
   PATTERN_HEIGHT_VECTOR : in PATTERN_VECTOR_TYPE ;
   PATTERN_WIDTH_VECTOR  : in PATTERN_VECTOR_TYPE ) ;
```

Function EDGE BUNDLE INDEX

```
procedure SET_EDGE_BUNDLE_INDEX(
   EDGE_BUNDLE_INDEX : in INDEX_TYPE ) ;
```

Function EDGE TYPE

```
procedure SET_EDGE_TYPE(
   EDGE_TYPE_INDICATOR : in INDEX_TYPE ) ;
```

Function EDGE WIDTH

```
procedure SET_EDGE_WIDTH(
   EDGE_WIDTH_SPECIFIER : in SIZE_SPECIF_TYPE ) ;
```

Function EDGE COLOUR

```
procedure SET_EDGE_COLOUR(
   EDGE_COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function EDGE CLIPPING MODE

```
procedure SET_EDGE_CLIPPING_MODE(
   CLIPPING_MODE : in CLIP_MODE_TYPE ) ;
```

Function EDGE VISIBILITY

```
procedure SET_EDGE_VISIBILITY(
   EDGE_VISIBILITY : in EDGE_VISIBILITY_TYPE ) ;
```

Function CLIP INDICATOR

```
procedure SET_CLIP_INDICATOR(
   CLIP_INDICATOR : in ON_OFF_FLAG_TYPE ) ;
```

Function CLIP RECTANGLE

```
procedure SET_CLIP_RECTANGLE(
   CLIP_RECTANGLE : in CLIP_RECTANGLE_TYPE ) ;
```

Function LINE WIDTH SPECIFICATION MODE

```
procedure SET_LINE_WIDTH_SPECIF_MODE(
  LINE_WIDTH_SPECIF_MODE : in SPECIF_MODE_TYPE ) ;
```

Function EDGE WIDTH SPECIFICATION MODE

```
procedure SET_EDGE_WIDTH_SPECIF_MODE(
  EDGE_WIDTH_SPECIF_MODE : in SPECIF_MODE_TYPE ) ;
```

Function MARKER SIZE SPECIFICATION MODE

```
procedure SET_MARKER_SIZE_SPECIF_MODE(
  MARKER_SIZE_SPECIF_MODE : in SPECIF_MODE_TYPE ) ;
```

Function COLOUR SELECTION MODE

```
procedure SET_COLOUR_SELECTION_MODE(
  COLOUR_SELECTION_MODE : in COLOUR_SELECTION_MODE_TYPE ) ;
```

Function COLOUR VALUE EXTENT

```
procedure SET_COLOUR_VALUE_EXTENT(
  MIN_COLOUR_VALUES  : in DIRECT_COLOUR_VALUE_TYPE ;
  MAX_COLOUR_VALUES : in DIRECT_COLOUR_VALUE_TYPE ) ;
```

Function BACKGROUND COLOUR

```
procedure SET_BACKGROUND_COLOUR(
  COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function AUXILIARY COLOUR

```
procedure SET_AUXILIARY_COLOUR(
  AUXILIARY_COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function TRANSPARENCY

```
procedure SET_TRANSPARENCY(
   TRANSPARENCY_SWITCH : in TRANSPARENCY_TYPE ) ;
```

Function COLOUR TABLE

```
procedure SET_COLOUR_TABLE(
   STARTING_INDEX : in COLOUR_INDEX_TYPE ;
   COLOUR_LIST    : in COLOUR_TABLE_TYPE ) ;
```

Function LINE REPRESENTATION

```
procedure SET_LINE_REP(
   LINE_BUNDLE_INDEX      : in INDEX_TYPE ;
   LINE_TYPE_INDICATOR    : in INDEX_TYPE ;
   LINE_WIDTH_SPECIFIER   : in SIZE_SPECIF_TYPE ;
   LINE_COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function MARKER REPRESENTATION

```
procedure SET_MARKER_REP(
   MARKER_BUNDLE_INDEX      : in INDEX_TYPE ;
   MARKER_TYPE_INDICATOR    : in INDEX_TYPE ;
   MARKER_SIZE_SPECIFIER    : in SIZE_SPECIF_TYPE ;
   MARKER_COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function TEXT REPRESENTATION

```
procedure SET_TEXT_REP(
   TEXT_BUNDLE_INDEX      : in INDEX_TYPE ;
   TEXT_FONT_INDEX        : in INDEX_TYPE ;
   TEXT_PREC              : in TEXT_PREC_TYPE ;
   CHAR_SPACING           : in REAL_TYPE ;
   CHAR_EXPAN_FACTOR      : in REAL_TYPE ;
   TEXT_COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function FILL REPRESENTATION

```
procedure SET_FILL_REP(
   FILL_BUNDLE_INDEX      : in INDEX_TYPE ;
   INTERIOR_STYLE         : in INTERIOR_STYLE_TYPE ;
   FILL_COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ;
   HATCH_INDEX            : in INDEX_TYPE ;
```

```
    PATTERN_INDEX          : in INDEX_TYPE ;
    FILL_BITMAP_ID         : in BITMAP_ID_TYPE ;
    FILL_BITMAP_REGION     : in BITMAP_REGION_TYPE ) ;
```

Function EDGE REPRESENTATION

```
procedure SET_EDGE_REP(
    EDGE_BUNDLE_INDEX      : in INDEX_TYPE ;
    EDGE_TYPE_INDICATOR    : in INDEX_TYPE ;
    EDGE_WIDTH_SPECIFIER   : in SIZE_SPECIF_TYPE ;
    EDGE_COLOUR_SPECIFIER  : in COLOUR_SPECIFIER_TYPE ;
    EDGE_VISIBILITY        : in EDGE_VISIBILITY_TYPE ) ;
```

Function DELETE BUNDLE REPRESENTATION

```
procedure DELETE_BUNDLE_REP(
    BUNDLE_TABLE_TYPE : in BUNDLE_TABLE_TYPE ;
    BUNDLE_INDEX      : in INDEX_TYPE ) ;
```

Function ASPECT SOURCE FLAGS

```
procedure SET_ASPECT_SOURCE_FLAGS(
    LIST_OF_ASF_TYPE_VALUE_PAIRS : in ASF_RECORD_ARRAY_TYPE ) ;
```

Function PATTERN TABLE

```
procedure SET_PATTERN_TABLE(
    PATTERN_TABLE_INDEX              : in INDEX_TYPE ;
    PATTERN_DIMENSIONS               : in PATTERN_DIMENSION_TYPE ;
    LOCAL_COLOUR_PREC_REQUIREMENT    : in INTEGER_TYPE ;
    PATTERN_COLOUR_SPECIFIERS        : in COLOUR_ARRAY_TYPE ) ;
```

Function DELETE PATTERN

```
procedure DELETE_PATTERN(
    PATTERN_TABLE_INDEX : in INDEX_TYPE ) ;
```

Function FONT LIST

```
procedure SET_FONT_LIST(
    FONT_NAMES : in FIXED_STRING_ARRAY_TYPE ) ;
```

Function CHARACTER SET LIST

```
procedure SET_CHAR_SET_LIST(
  LIST_OF_CHAR_SETS : in CHAR_SET_ARRAY_TYPE ) ;
```

Function SAVE PRIMITIVE ATTRIBUTES

```
procedure SAVE_PRIMITIVE_ATTRIBUTES(
  ATTRIBUTE_SET_NAME : in ASN_TYPE ) ;
```

Function RESTORE PRIMITIVE ATTRIBUTES

```
procedure RESTORE_PRIMITIVE_ATTRIBUTES(
  ATTRIBUTE_SET_NAME : in ASN_TYPE ) ;
```

Function DELETE PRIMITIVE ATTRIBUTE SAVE SET

```
procedure DELETE_PRIMITIVE_ATTRIBUTE_SAVE_SET(
  ATTRIBUTE_SET_NAME : in ASN_TYPE ) ;
```

Function BEGIN FIGURE

```
procedure BEGIN_FIGURE ;
```

Function END FIGURE

```
procedure END_FIGURE ;
```

Function NEW REGION

```
procedure NEW_REGION ;
```

Function GET TEXT EXTENT

```
procedure GET_TEXT_EXTENT(
  TEXT_POSITION                : in  VDC_POINT_TYPE ;
  CHAR_STRING                  : in  STRING_TYPE ;
  RESPONSE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
  CONCATENATION_VALIDITY       : out VALIDITY_FLAG_TYPE ;
  CONCATENATION_POINT          : out VDC_POINT_TYPE ;
  TEXT_EXTENT_PARALLELOGRAM : out TEXT_EXTENT_PARALLELOGRAM_TYPE ) ;
```

Function INQUIRE PRIMITIVE SUPPORT LEVELS

```
procedure INQ_PRIMITIVE_SUPPORT_LEVELS(
   RESPONSE_VALIDITY                    : out VALIDITY_FLAG_TYPE ;
   MAX_NUM_POLYLINE_POINTS              : out INTEGER_TYPE ;
   MAX_NUM_DISJOINT_POLYLINE_POINTS     : out INTEGER_TYPE ;
   MAX_NUM_POLYGON_POINTS               : out INTEGER_TYPE ;
   MAX_NUM_POLYGON_SET_POINTS           : out INTEGER_TYPE ;
   MAX_NUM_POLYMARKER_POINTS            : out INTEGER_TYPE ;
   MAX_NUM_CHARS_FOR_TEXT               : out INTEGER_TYPE ;
   MAX_NUM_CELL_ARRAY_CELL_COLOURS      : out INTEGER_TYPE ;
   LINE_EDGE_CONTINUITY_CAPABILITY      : out LINE_EDGE_CONTINUITY_CAP_TYPE ;
   CELL_ARRAY_FILL_CAPABILITY           : out CELL_ARRAY_FILL_CAP_TYPE ;
   CELL_ARRAY_ALIGNMENT_CAPABILITY      : out CELL_ARRAY_ALIGNMENT_CAP_TYPE ;
   CELL_ARRAY_RENDERING_TECHNIQUE       : out CELL_ARRAY_TECHNIQUE_TYPE ;
   COMPOUND_TEXT_CAPABILITY             : out COMPOUND_OBJECT_CAP_TYPE ;
   CLOSED_FIGURE_CAPABILITY             : out COMPOUND_OBJECT_CAP_TYPE ) ;
```

Function LOOKUP GDP SUPPORT

```
procedure LOOKUP_GDP_SUPPORT(
   LIST_OF_GDP_IDS                : in  INTEGER_ARRAY_TYPE ;
   RESPONSE_VALIDITY              : out VALIDITY_FLAG_TYPE ;
   LIST_OF_SUPPORT_INDICATORS     : out YES_NO_FLAG_ARRAY_TYPE ) ;
```

Function INQUIRE GDP ATTRIBUTES

```
procedure INQ_GDP_ATTRIBUTES(
   GDP_ID                         : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY              : out VALIDITY_FLAG_TYPE ;
   NUM_VALID_ATTRIBUTE_GROUPS     : out INTEGER_TYPE ;
   ARRAY_OF_ATTRIBUTE_GROUPS      : out BUNDLE_TABLE_ARRAY_TYPE ) ;
```

Function INQUIRE LINE CAPABILITY

```
procedure INQ_LINE_CAPABILITY(
   RESPONSE_VALIDITY                    : out VALIDITY_FLAG_TYPE ;
   NUM_OF_PREDEFINED_LINE_BUNDLES       : out INTEGER_TYPE ;
   NUM_OF_SETTABLE_LINE_BUNDLES         : out INTEGER_TYPE ;
   MAX_LINE_BUNDLE_INDEX                : out INDEX_TYPE ;
   DYN_MOD_ACCEPTED_FOR_LINE_BUNDLES    : out DYN_MOD_FLAG_TYPE ;
   NOMINAL_SCALED_LINE_WIDTH            : out DEVICE_COORD_TYPE ;
   MIN_SCALED_LINE_WIDTH                : out DEVICE_COORD_TYPE ;
   MAX_SCALED_LINE_WIDTH                : out DEVICE_COORD_TYPE ;
   NUM_OF_VALID_LINE_CLIPPING_MODES     : out INTEGER_TYPE ;
```

```
    ARRAY_OF_AVAIL_LINE_CLIPPING_MODES  : out CLIP_MODE_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE LINE TYPES

```
procedure INQ_AVAIL_LINE_TYPES(
   NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
   LIST_OF_LINE_TYPES                    : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE SCALED LINE WIDTHS

```
procedure INQ_AVAIL_SCALED_LINE_WIDTHS(
   NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
   LIST_OF_SCALED_LINE_WIDTHS            : out DEVICE_COORD_ARRAY_TYPE ) ;
```

Function INQUIRE MARKER CAPABILITY

```
procedure INQ_MARKER_CAPABILITY(
   RESPONSE_VALIDITY                       : out VALIDITY_FLAG_TYPE ;
   NUM_OF_PREDEFINED_MARKER_BUNDLES        : out INTEGER_TYPE ;
   NUM_OF_SETTABLE_MARKER_BUNDLES          : out INTEGER_TYPE ;
   MAX_MARKER_BUNDLE_INDEX                 : out INDEX_TYPE ;
   DYN_MOD_ACCEPTED_FOR_MARKER_BUNDLES : out DYN_MOD_FLAG_TYPE ;
   NOMINAL_SCALED_MARKER_SIZE              : out DEVICE_COORD_TYPE ;
   MINIMAL_SCALED_MARKER_SIZE              : out DEVICE_COORD_TYPE ;
   MAX_SCALED_MARKER_SIZE                  : out DEVICE_COORD_TYPE ;
   NUM_OF_VALID_MARKER_CLIPPING_MODES      : out INTEGER_TYPE ;
   ARRAY_OF_AVAIL_MARKER_CLIPPING_MODES  : out CLIP_MODE_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE MARKER TYPES

```
procedure INQ_AVAIL_MARKER_TYPES(
   NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
   LIST_OF_MARKER_TYPES                  : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE SCALED MARKER SIZES

```
procedure INQ_AVAIL_SCALED_MARKER_SIZES(
  NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE   : out INTEGER_TYPE ;
  LIST_OF_SCALED_MARKER_SIZES           : out DEVICE_COORD_ARRAY_TYPE ) ;
```

Function INQUIRE TEXT CAPABILITY

```
procedure INQ_TEXT_CAPABILITY(
  RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
  NUM_OF_PREDEFINED_TEXT_BUNDLES        : out INTEGER_TYPE ;
  NUM_OF_SETTABLE_TEXT_BUNDLES          : out INTEGER_TYPE ;
  MAX_TEXT_BUNDLE_INDEX                 : out INDEX_TYPE ;
  DYN_MOD_ACCEPTED_FOR_TEXT_BUNDLES     : out DYN_MOD_FLAG_TYPE ;
  MAX_LENGTH_OF_FONT_LIST               : out INTEGER_TYPE ;
  DYN_MOD_ACCEPTED_FOR_FONT_LIST        : out DYN_MOD_FLAG_TYPE ;
  MAX_LENGTH_OF_CHAR_SET_LIST           : out INTEGER_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE CHARACTER SETS

```
procedure INQ_AVAIL_CHAR_SETS(
  NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
  MAX_CHARS_PER_STRING                  : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE   : out INTEGER_TYPE ;
  LIST_OF_AVAIL_CHAR_SETS               : out CHAR_SET_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE TEXT FONTS

```
procedure INQ_AVAIL_TEXT_FONTS(
  NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
  MAX_CHARS_PER_STRING                  : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE   : out INTEGER_TYPE ;
  LIST_OF_FONT_NAMES                    : out FIXED_STRING_ARRAY_TYPE ) ;
```

Function INQUIRE FONT CAPABILITIES

```
procedure INQ_FONT_CAPABILITIES(
  FONT_NAME                             : in  FIXED_STRING_TYPE ;
  PREC                                  : in  TEXT_PREC_TYPE ;
```

```
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   MIN_CHAR_EXPANSION_FACTOR                  : out REAL_TYPE ;
   MAX_CHAR_EXPANSION_FACTOR                  : out REAL_TYPE ;
   MIN_CHAR_SPACING                           : out REAL_TYPE ;
   MAX_CHAR_SPACING                           : out REAL_TYPE ;
   MIN_CHAR_HEIGHT                            : out REAL_TYPE ;
   MAX_CHAR_HEIGHT                            : out REAL_TYPE ;
   SKEWED_VECTOR_SUPPORT                      : out YES_NO_FLAG_TYPE ;
   MIRRORED_CHAR_SUPPORT                      : out YES_NO_FLAG_TYPE ;
   NUM_OF_TEXT_PATH_ELEMENTS                  : out INTEGER_TYPE ;
   ARRAY_OF_SUPPORTED_TEXT_PATHS              : out TEXT_PATH_ARRAY_TYPE ;
   NUM_OF_HOR_TEXT_ALIGNMENT_ELEMENTS         : out INTEGER_TYPE ;
   SUPPORTED_HOR_TEXT_ALIGNMENTS              : out HOR_ALIGNMENT_ARRAY_TYPE ;
   NUM_OF_VERT_TEXT_ALIGNMENT_ELEMENTS        : out INTEGER_TYPE ;
   SUPPORTED_VERT_TEXT_ALIGNMENTS             : out VERT_ALIGNMENT_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE CHARACTER EXPANSION FACTORS

```
procedure INQ_AVAIL_CHAR_EXPAN_FACTORS(
   FONT_NAME                                  : in  FIXED_STRING_TYPE ;
   PREC                                       : in  TEXT_PREC_TYPE ;
   NUM_OF_LIST_ELEMENTS_REQUESTED             : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN           : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE        : out INTEGER_TYPE ;
   LIST_OF_CHAR_EXPAN_FACTORS                 : out REAL_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE CHARACTER SPACINGS

```
procedure INQ_AVAIL_CHAR_SPACINGS(
   FONT_NAME                                  : in  FIXED_STRING_TYPE ;
   PREC                                       : in  TEXT_PREC_TYPE ;
   NUM_OF_LIST_ELEMENTS_REQUESTED             : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN           : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE        : out INTEGER_TYPE ;
   LIST_OF_CHAR_SPACINGS                      : out REAL_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE CHARACTER HEIGHTS

```
procedure INQ_AVAIL_CHAR_HEIGHTS(
   FONT_NAME                                  : in  FIXED_STRING_TYPE ;
   PREC                                       : in  TEXT_PREC_TYPE ;
   NUM_OF_LIST_ELEMENTS_REQUESTED             : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN           : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
```

```
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
    LIST_OF_CHAR_HEIGHTS                : out DEVICE_COORD_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE CHARACTER ORIENTATIONS

```
procedure INQ_AVAIL_CHAR_ORIENTATIONS(
    FONT_NAME                           : in  FIXED_STRING_TYPE ;
    PREC                                : in  TEXT_PREC_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED      : in  INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
    LIST_OF_CHAR_ORIENTATIONS           : out ORIENTATION_ARRAY_TYPE ) ;
```

Function INQUIRE FILL CAPABILITY

```
procedure INQ_FILL_CAPABILITY(
    RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
    NUM_OF_PREDEFINED_FILL_BUNDLES      : out INTEGER_TYPE ;
    NUM_OF_SETTABLE_FILL_BUNDLES        : out INTEGER_TYPE ;
    MAX_FILL_BUNDLE_INDEX               : out INDEX_TYPE ;
    DYN_MOD_ACCEPTED_FOR_FILL_BUNDLES   : out DYN_MOD_FLAG_TYPE ;
    NUM_OF_INTERIOR_STYLE_ELEMENTS      : out INTEGER_TYPE ;
    ARRAY_OF_AVAIL_INTERIOR_STYLES      : out INTERIOR_STYLE_ARRAY_TYPE ;
    NUM_OF_PREDEFINED_PATTERNS          : out INTEGER_TYPE ;
    NUM_OF_SETTABLE_PATTERNS            : out INTEGER_TYPE ;
    MAX_PATTERN_INDEX                   : out INDEX_TYPE ;
    DYN_MOD_ACCEPTED_FOR_PATTERN_TABLE  : out DYN_MOD_FLAG_TYPE ;
    PREFERRED_PATTERN_SIZE_DIVISOR      : out INTEGER_TYPE ;
    MAX_PATTERN_SIZE                    : out PATTERN_DIMENSION_TYPE ;
    PATTERN_TRAN_SUPPORT                : out PATTERN_TRAN_SUPPORT_TYPE ;
    PATTERN_FILL_FALLBACK               : out PATTERN_FILL_FALLBACK_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE HATCH STYLES

```
procedure INQ_AVAIL_HATCH_STYLES(
    NUM_OF_LIST_ELEMENTS_REQUESTED      : in  INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
    LIST_OF_HATCH_STYLES                : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE EDGE CAPABILITY

```
procedure INQ_EDGE_CAPABILITY(
```

```
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   NUM_OF_PREDEFINED_EDGE_BUNDLES             : out INTEGER_TYPE ;
   NUM_OF_SETTABLE_EDGE_BUNDLES               : out INTEGER_TYPE ;
   MAX_EDGE_BUNDLE_INDEX                      : out INDEX_TYPE ;
   DYN_MOD_ACCEPTED_FOR_EDGE_BUNDLES          : out DYN_MOD_FLAG_TYPE ;
   NOMINAL_SCALED_EDGE_WIDTH                  : out DEVICE_COORD_TYPE ;
   MIN_SCALED_EDGE_WIDTH                      : out DEVICE_COORD_TYPE ;
   MAX_SCALED_EDGE_WIDTH                      : out DEVICE_COORD_TYPE ;
   NUM_OF_VALID_EDGE_CLIPPING_MODES           : out INTEGER_TYPE ;
   ARRAY_OF_AVAIL_EDGE_CLIPPING_MODES         : out CLIP_MODE_ARRAY_TYPE ;
   REALIZATION_OF_EDGE_WIDTH                  : out EDGE_WIDTH_REALIZATION_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE EDGE TYPES

```
procedure INQ_AVAIL_EDGE_TYPES(
   NUM_OF_LIST_ELEMENTS_REQUESTED             : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN           : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE        : out INTEGER_TYPE ;
   LIST_OF_EDGE_TYPES                         : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE SCALED EDGE WIDTHS

```
procedure INQ_AVAIL_SCALED_EDGE_WIDTHS(
   NUM_OF_LIST_ELEMENTS_REQUESTED             : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN           : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE        : out INTEGER_TYPE ;
   LIST_OF_EDGE_WIDTHS                        : out DEVICE_COORD_ARRAY_TYPE ) ;
```

Function INQUIRE COLOUR CAPABILITY

```
procedure INQ_COLOUR_CAPABILITY(
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   NUM_SIMULT_AVAIL_DIRECT_COLOURS            : out INTEGER_TYPE ;
   NUM_SIMULT_AVAIL_INDEX_COLOURS             : out INTEGER_TYPE ;
   NUM_AVAIL_COLOURS                          : out INTEGER_TYPE ;
   NUM_AVAIL_INTENSITIES                      : out INTENSITY_ARRAY_TYPE ;
   COLOUR_SELECTION_MODE_AVAIL                : out COLOUR_SELECTION_MODE_AVAIL_TYPE;
   DYN_MOD_ACCEPTED_FOR_COLOUR                : out DYN_MOD_FLAG_TYPE ;
   COLOUR_OVERWRITE_CAPABILITY                : out YES_NO_FLAG_TYPE ;
   COLOUR_REALIZATION                         : out COLOUR_REALIZATION_TYPE ;
   MONOCHROMATIC_DEVICE                       : out YES_NO_FLAG_TYPE ;
   BACKGROUND_COLOUR_CAPABILITY               : out BACKGROUND_COLOUR_CAP_TYPE ) ;
```

Function INQUIRE CIE CHARACTERISTICS

```
procedure INQ_CIE_CHARACTERISTICS(
  CIE_1931_CHROMATICITY_COORD_FOR_RED   : out CIE_CHROMATICITY_TYPE ;
  CIE_1931_CHROMATICITY_COORD_FOR_GREEN : out CIE_CHROMATICITY_TYPE ;
  CIE_1931_CHROMATICITY_COORD_FOR_BLUE  : out CIE_CHROMATICITY_TYPE ;
  CIE_1931_TRISTIMULUS_VALUES_FOR_WHITE : out CIE_TRISTIMULUS_VALUE_TYPE ) ;
```

Function INQUIRE MAXIMUM NUMBER OF SIMULTANEOUSLY SAVED ATTRIBUTE SETS

```
procedure INQ_MAX_SIMULTANEOUS_ATTRIBUTE_SETS(
  RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE ;
  MAX_SIMULTANEOUS_ATTRIBUTE_SETS : out FIXED_INTEGER_16_TYPE ) ;
```

Function INQUIRE ARRAY OF SUPPORTED CHARACTER CODING ANNOUNCERS

```
procedure INQ_SUPPORTED_CHAR_CODING_ANNOUNCERS(
  RESPONSE_VALIDITY                 : out VALIDITY_FLAG_TYPE ;
  NUM_VALID_CHAR_CODING_ANNOUNCERS : out INTEGER_TYPE ;
  VALID_CHAR_CODING_ANNOUNCERS      : out CHAR_CODING_ANNOUNCER_ARRAY_TYPE ) ;
```

Function INQUIRE LINE ATTRIBUTES

```
procedure INQ_LINE_ATTRIBUTES(
  RESPONSE_VALIDITY          : out VALIDITY_FLAG_TYPE ;
  LINE_BUNDLE_INDEX          : out INDEX_TYPE ;
  LINE_TYPE                  : out INDEX_TYPE ;
  LINE_WIDTH_SPECIF_MODE     : out SPECIF_MODE_TYPE ;
  LINE_WIDTH                 : out SIZE_SPECIF_TYPE ;
  LINE_COLOUR_SELECTION_MODE : out COLOUR_SELECTION_MODE_TYPE ;
  LINE_COLOUR                : out COLOUR_SPECIFIER_TYPE ;
  LINE_CLIPPING_MODE         : out CLIP_MODE_TYPE ) ;
```

Function INQUIRE LIST OF LINE BUNDLE INDICES

```
procedure INQ_LINE_BUNDLE_INDICES(
  NUM_OF_LIST_ELEMENTS_REQUESTED   : in  INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_STATE_LIST     : out INTEGER_TYPE ;
  LIST_OF_LINE_BUNDLE_INDICES      : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LINE REPRESENTATION

```
procedure INQ_LINE_REP(
   LINE_BUNDLE_INDEX              : in  INDEX_TYPE ;
   RESPONSE_VALIDITY              : out VALIDITY_FLAG_TYPE ;
   LINE_TYPE                      : out INDEX_TYPE ;
   LINE_WIDTH_SPECIF_MODE         : out SPECIF_MODE_TYPE ;
   LINE_WIDTH                     : out SIZE_SPECIF_TYPE ;
   LINE_COLOUR_SELECTION_MODE : out COLOUR_SELECTION_MODE_TYPE ;
   LINE_COLOUR                    : out COLOUR_SPECIFIER_TYPE ) ;
```

Function INQUIRE MARKER ATTRIBUTES

```
procedure INQ_MARKER_ATTRIBUTES(
   RESPONSE_VALIDITY              : out VALIDITY_FLAG_TYPE ;
   MARKER_BUNDLE_INDEX            : out INDEX_TYPE ;
   MARKER_TYPE                    : out INDEX_TYPE ;
   MARKER_SIZE_SPECIF_MODE        : out SPECIF_MODE_TYPE ;
   MARKER_SIZE                    : out SIZE_SPECIF_TYPE ;
   MARKER_COLOUR_SELECTION_MODE : out COLOUR_SELECTION_MODE_TYPE ;
   MARKER_COLOUR                  : out COLOUR_SPECIFIER_TYPE ;
   MARKER_CLIPPING_MODE           : out CLIP_MODE_TYPE ) ;
```

Function INQUIRE LIST OF MARKER BUNDLE INDICES

```
procedure INQ_MARKER_BUNDLE_INDICES(
   NUM_OF_LIST_ELEMENTS_REQUESTED  : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY               : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_STATE_LIST    : out INTEGER_TYPE ;
   LIST_OF_MARKER_BUNDLE_INDICES   : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE MARKER REPRESENTATION

```
procedure INQ_MARKER_REP(
   MARKER_BUNDLE_INDEX            : in  INDEX_TYPE ;
   RESPONSE_VALIDITY              : out VALIDITY_FLAG_TYPE ;
   MARKER_TYPE                    : out INDEX_TYPE ;
   MARKER_SIZE_SPECIF_MODE        : out SPECIF_MODE_TYPE ;
   MARKER_SIZE                    : out SIZE_SPECIF_TYPE ;
   MARKER_COLOUR_SELECTION_MODE : out COLOUR_SELECTION_MODE_TYPE ;
   MARKER_COLOUR                  : out COLOUR_SPECIFIER_TYPE ) ;
```

Function INQUIRE TEXT ATTRIBUTES

```
procedure INQ_TEXT_ATTRIBUTES(
```

```
   RESPONSE_VALIDITY                  : out VALIDITY_FLAG_TYPE ;
   TEXT_BUNDLE_INDEX                  : out INDEX_TYPE ;
   TEXT_FONT_INDEX                    : out INDEX_TYPE ;
   FONT_PREC                          : out TEXT_PREC_TYPE ;
   CHAR_EXPAN_FACTOR                  : out REAL_TYPE ;
   CHAR_SPACING                       : out REAL_TYPE ;
   SELECTION_MODE_OF_TEXT_COLOUR      : out COLOUR_SELECTION_MODE_TYPE ;
   TEXT_COLOUR                        : out COLOUR_SPECIFIER_TYPE ;
   CHAR_HEIGHT                        : out VDC_TYPE ;
   CHAR_UP_VECTOR                     : out CHAR_VECTOR_TYPE ;
   CHAR_BASE_VECTOR                   : out CHAR_VECTOR_TYPE ;
   TEXT_PATH                          : out TEXT_PATH_TYPE ;
   HOR_TEXT_ALIGNMENT                 : out HOR_ALIGNMENT_TYPE ;
   CONT_HOR_ALIGNMENT                 : out REAL_TYPE ;
   VERT_TEXT_ALIGNMENT                : out VERT_ALIGNMENT_TYPE ;
   CONT_VERT_ALIGNMENT                : out REAL_TYPE ;
   CHAR_SET_INDEX                     : out INDEX_TYPE ;
   ALTERNATE_CHAR_SET_INDEX           : out INDEX_TYPE ;
   CHAR_CODING_ANNOUNCER              : out CODING_TECHNIQUE_TYPE ) ;
```

Function INQUIRE LIST OF TEXT BUNDLE INDICES

```
procedure INQ_TEXT_BUNDLE_INDICES(
   NUM_OF_LIST_ELEMENTS_REQUESTED     : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN   : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                  : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_STATE_LIST       : out INTEGER_TYPE ;
   LIST_OF_TEXT_BUNDLE_INDICES        : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE TEXT REPRESENTATION

```
procedure INQ_TEXT_REP(
   TEXT_BUNDLE_INDEX                  : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                  : out VALIDITY_FLAG_TYPE ;
   TEXT_FONT_INDEX                    : out INDEX_TYPE ;
   TEXT_PREC                          : out TEXT_PREC_TYPE ;
   CHAR_EXPAN_FACTOR                  : out REAL_TYPE ;
   CHAR_SPACING                       : out REAL_TYPE ;
   TEXT_COLOUR_SELECTION_MODE         : out COLOUR_SELECTION_MODE_TYPE ;
   TEXT_COLOUR                        : out COLOUR_SPECIFIER_TYPE ) ;
```

Function INQUIRE FILL ATTRIBUTES

```
procedure INQ_FILL_ATTRIBUTES(
   RESPONSE_VALIDITY                  : out VALIDITY_FLAG_TYPE ;
   FILL_BUNDLE_INDEX                  : out INDEX_TYPE ;
```

```
    INTERIOR_STYLE                  : out INTERIOR_STYLE_TYPE ;
    FILL_COLOUR_SELECTION_MODE      : out COLOUR_SELECTION_MODE_TYPE ;
    FILL_COLOUR                     : out COLOUR_SPECIFIER_TYPE ;
    HATCH_INDEX                     : out INDEX_TYPE ;
    PATTERN_INDEX                   : out INDEX_TYPE ;
    FILL_BITMAP_ID                  : out BITMAP_ID_TYPE ;
    FILL_BITMAP_REGION              : out BITMAP_REGION_TYPE ;
    FILL_REFERENCE_POINT            : out VDC_POINT_TYPE ;
    PATTERN_SIZE                    : out PATTERN_SIZE_TYPE ) ;
```

Function INQUIRE PATTERN DIMENSIONS

```
procedure INQ_PATTERN_DIMENSIONS(
    PATTERN_INDEX                          : in  INDEX_TYPE ;
    LOCAL_COLOUR_PREC_REQUIREMENT          : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY                      : out VALIDITY_FLAG_TYPE ;
    NUM_OF_HOR_SAMPLES                     : out INTEGER_TYPE ;
    NUM_OF_VERT_SAMPLES                    : out INTEGER_TYPE ;
    LOCAL_COLOUR_PREC                      : out INTEGER_TYPE ;
    SELECTION_MODE_OF_COLOUR_SPECIFIERS : out COLOUR_SELECTION_MODE_TYPE ) ;
```

Function INQUIRE PATTERN

```
procedure INQ_PATTERN(
    PATTERN_INDEX              : in  INDEX_TYPE ;
    LOCAL_COLOUR_PREC          : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY          : out VALIDITY_FLAG_TYPE ;
    LIST_OF_COLOUR_SPECIFIERS  : out COLOUR_SPECIFIER_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF PATTERN INDICES

```
procedure INQ_PATTERN_INDICES(
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_STATE_LIST     : out INTEGER_TYPE ;
    LIST_OF_PATTERN_INDICES          : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF FILL BUNDLE INDICES

```
procedure INQ_FILL_BUNDLE_INDICES(
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE ;
```

```
   TOTAL_ELEMENTS_IN_STATE_LIST      : out INTEGER_TYPE ;
   LIST_OF_FILL_BUNDLE_INDICES       : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE FILL REPRESENTATION

```
procedure INQ_FILL_REP(
   FILL_BUNDLE_INDEX                 : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                 : out VALIDITY_FLAG_TYPE ;
   INTERIOR_STYLE                    : out INTERIOR_STYLE_TYPE ;
   FILL_COLOUR_SELECTION_MODE        : out COLOUR_SELECTION_MODE_TYPE ;
   FILL_COLOUR                       : out COLOUR_SPECIFIER_TYPE ;
   HATCH_INDEX                       : out INDEX_TYPE ;
   PATTERN_INDEX                     : out INDEX_TYPE ;
   FILL_BITMAP_ID                    : out BITMAP_ID_TYPE ;
   FILL_BITMAP_REGION                : out BITMAP_REGION_TYPE ) ;
```

Function INQUIRE EDGE ATTRIBUTES

```
procedure INQ_EDGE_ATTRIBUTES(
   RESPONSE_VALIDITY                 : out VALIDITY_FLAG_TYPE ;
   EDGE_BUNDLE_INDEX                 : out INDEX_TYPE ;
   EDGE_VISIBILITY                   : out VISIBILITY_TYPE ;
   EDGE_TYPE                         : out INDEX_TYPE ;
   EDGE_WIDTH_SPECIF_MODE            : out SPECIF_MODE_TYPE ;
   EDGE_WIDTH                        : out SIZE_SPECIF_TYPE ;
   EDGE_COLOUR_SELECTION_MODE        : out COLOUR_SELECTION_MODE_TYPE ;
   EDGE_COLOUR                       : out COLOUR_SPECIFIER_TYPE ;
   EDGE_CLIPPING_MODE                : out CLIP_MODE_TYPE ) ;
```

Function INQUIRE LIST OF EDGE BUNDLE INDICES

```
procedure INQ_EDGE_BUNDLE_INDICES(
   NUM_OF_LIST_ELEMENTS_REQUESTED    : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN  : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                 : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_STATE_LIST      : out INTEGER_TYPE ;
   LIST_OF_EDGE_BUNDLE_INDICES       : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE EDGE REPRESENTATION

```
procedure INQ_EDGE_REP(
   EDGE_BUNDLE_INDEX                 : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                 : out VALIDITY_FLAG_TYPE ;
   EDGE_VISIBILITY                   : out VISIBILITY_TYPE ;
   EDGE_TYPE                         : out INDEX_TYPE ;
```

```
   EDGE_WIDTH_SPECIF_MODE          : out SPECIF_MODE_TYPE ;
   EDGE_WIDTH                      : out SIZE_SPECIF_TYPE ;
   EDGE_COLOUR_SELECTION_MODE : out COLOUR_SELECTION_MODE_TYPE ;
   EDGE_COLOUR                     : out COLOUR_SPECIFIER_TYPE ) ;
```

INQUIRE OUTPUT STATE

```
procedure INQ_OUTPUT_STATE(
   RESPONSE_VALIDITY          : out VALIDITY_FLAG_TYPE ;
   OUTPUT_STATE               : out OUTPUT_STATE_TYPE ;
   LINE_WIDTH_SPECIF_MODE     : out SPECIF_MODE_TYPE ;
   MARKER_SIZE_SPECIF_MODE    : out SPECIF_MODE_TYPE ;
   EDGE_WIDTH_SPECIF_MODE     : out SPECIF_MODE_TYPE ) ;
```

Function INQUIRE OBJECT CLIPPING

```
procedure INQ_OBJECT_CLIPPING(
   RESPONSE_VALIDITY : out VALIDITY_FLAG_TYPE ;
   CLIP_INDICATOR    : out ON_OFF_FLAG_TYPE ;
   CLIP_RECTANGLE    : out CLIP_RECTANGLE_TYPE ) ;
```

Function INQUIRE LIST OF ATTRIBUTE SET NAMES IN USE

```
procedure INQ_ATTRIBUTE_SET_NAMES_IN_USE(
   NUM_OF_LIST_ELEMENTS_REQUESTED   : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_STATE_LIST     : out INTEGER_TYPE ;
   LIST_OF_ATTRIBUTE_SET_NAMES      : out ASN_ARRAY_TYPE ) ;
```

Function INQUIRE COLOUR STATE

```
procedure INQ_COLOUR_STATE(
   RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE ;
   COLOUR_SELECTION_MODE            : out COLOUR_SELECTION_MODE_TYPE ;
   COLOUR_VALUE_EXTENT              : out COLOUR_VALUE_EXTENT_TYPE ;
   AUXILIARY_COLOUR_SELECTION_MODE  : out COLOUR_SELECTION_MODE_TYPE ;
   AUXILIARY_COLOUR                 : out COLOUR_SPECIFIER_TYPE ;
   TRANSPARENCY                     : out TRANSPARENCY_TYPE ;
   BACKGROUND_COLOUR_SELECTION_MODE : out COLOUR_SELECTION_MODE_TYPE ;
   BACKGROUND_COLOUR                : out COLOUR_SPECIFIER_TYPE ) ;
```

Function INQUIRE LIST OF COLOUR TABLE ENTRIES

```
procedure INQ_COLOUR_TABLE_ENTRIES(
  NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
  LIST_OF_COLOUR_TABLE_ENTRIES          : out COLOUR_TABLE_TYPE ) ;
```

Function INQUIRE FONT LIST

```
procedure INQ_FONT_LIST(
  NUM_OF_LIST_ELEMENTS_REQUESTED          : in  INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN        : in  INTEGER_TYPE ;
  MAX_CHARS_PER_STRING                    : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY                       : out VALIDITY_FLAG_TYPE ;
  TOTAL_LENGTH_OF_LONGEST_STRING_PRESENT : out INTEGER_TYPE ;
  TOTAL_ELEMENTS_IN_STATE_LIST            : out INTEGER_TYPE ;
  LIST_OF_FONT_NAMES                      : out FIXED_STRING_ARRAY_TYPE ) ;
```

Function INQUIRE CHARACTER SET LIST

```
procedure INQ_CHAR_SET_LIST(
  NUM_OF_LIST_ELEMENTS_REQUESTED          : in  INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN        : in  INTEGER_TYPE ;
  MAX_CHARS_PER_STRING                    : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY                       : out VALIDITY_FLAG_TYPE ;
  TOTAL_LENGTH_OF_LONGEST_STRING_PRESENT : out INTEGER_TYPE ;
  TOTAL_ELEMENTS_IN_STATE_LIST            : out INTEGER_TYPE ;
  LIST_OF_CHAR_SETS                       : out CHAR_SET_ARRAY_TYPE ) ;
```

Function LOOKUP ASPECT SOURCE FLAGS

```
procedure LOOKUP_ASPECT_SOURCE_FLAGS(
  LIST_OF_ASF_TYPES_REQUESTED : in  ASF_ARRAY_TYPE ;
  RESPONSE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
  LIST_OF_ASF_VALUES          : out ASF_VALUE_ARRAY_TYPE ) ;
```

7.3 Part 4 segment functions

Function GET NEW SEGMENT IDENTIFIER

```
procedure GET_NEW_SEGMENT_ID(
   RESPONSE_VALIDITY : out VALIDITY_FLAG_TYPE ;
   SEGMENT_ID        : out SEGMENT_ID_TYPE ) ;
```

Function CREATE SEGMENT

```
procedure CREATE_SEGMENT(
   SEGMENT_ID : in SEGMENT_ID_TYPE ) ;
```

Function REOPEN SEGMENT

```
procedure REOPEN_SEGMENT(
   SEGMENT_ID : in SEGMENT_ID_TYPE ) ;
```

Function CLOSE SEGMENT

```
procedure CLOSE_SEGMENT ;
```

Function COPY SEGMENT

```
procedure COPY_SEGMENT(
   SOURCE_SEGMENT_ID          : in SEGMENT_ID_TYPE ;
   COPY_TRAN                  : in SEGMENT_TRANSFORMATION_TYPE ;
   SEGMENT_TRAN_ASSOCIATION : in YES_NO_FLAG_TYPE ) ;
```

Function DELETE SEGMENT

```
procedure DELETE_SEGMENT(
   SEGMENT_ID : in SEGMENT_ID_TYPE ) ;
```

Function DELETE ALL SEGMENTS

```
procedure DELETE_ALL_SEGMENTS ;
```

Function RENAME SEGMENT

```
procedure RENAME_SEGMENT(
```

```
    OLD_SEGMENT_ID  : in SEGMENT_ID_TYPE ;
    NEW_SEGMENT_ID : in SEGMENT_ID_TYPE ) ;
```

Function DRAW ALL SEGMENTS

```
procedure DRAW_ALL_SEGMENTS ;
```

Function IMPLICIT SEGMENT REGENERATION MODE

```
procedure SET_IMPL_SEG_REGEN_MODE(
   IMPLICIT_SEGMENT_REGEN_MODE : in IMPLICIT_SEGMENT_REGEN_MODE_TYPE ) ;
```

Function RESET REGENERATION PENDING

```
procedure RESET_REGEN_PENDING ;
```

Function PICK IDENTIFIER

```
procedure SET_PICK_ID(
   PICK_ID : in PICK_ID_TYPE ) ;
```

Function SEGMENT VISIBILITY

```
procedure SET_SEGMENT_VISIBILITY(
   SEGMENT_ID : in SEGMENT_ID_TYPE ;
   VISIBILITY    : in VISIBILITY_TYPE ) ;
```

Function SEGMENT TRANSFORMATION

```
procedure SET_SEGMENT_TRANSFORMATION(
   SEGMENT_ID      : in SEGMENT_ID_TYPE ;
   SEGMENT_TRAN : in SEGMENT_TRANSFORMATION_TYPE ) ;
```

Function SEGMENT HIGHLIGHTING

```
procedure SET_SEGMENT_HIGHLIGHTING(
   SEGMENT_ID    : in SEGMENT_ID_TYPE ;
   HIGHLIGHTING : in HIGHLIGHTING_TYPE ) ;
```

Function SEGMENT DISPLAY PRIORITY

```
procedure SET_SEGMENT_DISP_PRIORITY(
   SEGMENT_ID    : in SEGMENT_ID_TYPE ;
   DISP_PRIORITY : in INTEGER_TYPE ) ;
```

Function SEGMENT DETECTABILITY

```
procedure SET_SEGMENT_DETECTABILITY(
   SEGMENT_ID      : in SEGMENT_ID_TYPE ;
   DETECTABILITY : in DETECTABILITY_TYPE ) ;
```

Function SEGMENT PICK PRIORITY

```
procedure SET_SEGMENT_PICK_PRIORITY(
   SEGMENT_ID    : in SEGMENT_ID_TYPE ;
   PICK_PRIORITY : in INTEGER_TYPE ) ;
```

Function SIMULATE PICK

```
procedure SIMULATE_PICK(
   PICK_POINT                          : in  VDC_POINT_TYPE ;
   PICK_APERTURE                       : in  PICK_APERTURE_TYPE ;
   NUM_OF_REQUESTED_PICK_VALUES        : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE;
   TOTAL_NUM_OF_AVAIL_PICK_VALUES      : out INTEGER_TYPE ;
   LIST_OF_PICK_VALUES                 : out PICK_VALUE_ARRAY_TYPE ) ;
```

Function INHERITANCE FILTER

```
procedure SET_INHERITANCE_FILTER(
   FILTER_SELECTION_LIST : in FILTER_SELECTION_LIST_TYPE ;
   SELECTION_SETTING     : in INHERITANCE_FILTER_TYPE ) ;
```

Function CLIPPING INHERITANCE

```
procedure SET_CLIPPING_INHERITANCE(
   SELECTION_SETTING : in CLIP_INHERITANCE_FILTER_TYPE ) ;
```

Function INQUIRE SEGMENT CAPABILITY

```
procedure INQ_SEGMENT_CAPABILITY(
   RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
   NUM_OF_SUPPORTED_DISPLAY_PRIORITIES : out INTEGER_TYPE ;
```

```
   NUM_OF_SUPPORTED_PICK_PRIORITIES          : out INTEGER_TYPE ;
   MAX_NUM_OF_SIMULTANEOUS_SEGMENTS          : out INTEGER_TYPE ;
   MAX_LEN_OF_SIMULATE_PICK_LIST             : out INTEGER_TYPE ;
   DYN_MOD_FOR_ADDING_TO_OPEN_SEGMENT        : out DYN_MOD_FLAG_TYPE ;
   DYN_MOD_FOR_SEGMENT_DELETION              : out DYN_MOD_FLAG_TYPE ;
   DYN_MOD_FOR_SEGMENT_TRANSFORMATION        : out DYN_MOD_FLAG_TYPE ;
   DYN_MOD_FOR_DISPLAY_PRIORITY_CHANGE       : out DYN_MOD_FLAG_TYPE ;
   DYN_MOD_FOR_VISIBILITY_TO_INVISIBLE       : out DYN_MOD_FLAG_TYPE ;
   DYN_MOD_FOR_VISIBILITY_TO_VISIBLE         : out DYN_MOD_FLAG_TYPE ;
   DYN_MOD_FOR_HIGHLIGHTING_CHANGE           : out DYN_MOD_FLAG_TYPE ;
   EFFECTIVE_RELATIVE_DISPLAY_PRIORITY       : out EFFECTIVE_PRIORITY_TYPE ;
   PICK_APERTURE_SHAPE                       : out PICK_APERTURE_SHAPE_TYPE ;
   TRANSFORMED_CLIP_REGION_EFFECTIVENESS     : out CLIP_EFFECTIVENESS_TYPE ) ;
```

Function INQUIRE SEGMENT STATE

```
procedure INQ_SEGMENT_STATE(
   RESPONSE_VALIDITY                 : out VALIDITY_FLAG_TYPE ;
   SEGMENT_OPEN_STATE                : out SEGMENT_OPEN_STATE_TYPE ;
   ID_OF_OPEN_SEGMENT                : out SEGMENT_ID_TYPE ;
   PICK_ID                           : out PICK_ID_TYPE ;
   IMPLICIT_SEGMENT_REGEN_MODE       : out IMPLICIT_SEGMENT_REGEN_MODE_TYPE ;
   REGEN_PENDING                     : out YES_NO_FLAG_TYPE ) ;
```

Function INQUIRE LIST OF INHERITANCE FILTER SETTINGS

```
procedure INQ_INHERITANCE_FILTER_SETTINGS(
   NUM_OF_LIST_ELEMENTS_REQUESTED    : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN  : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                 : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_PRESENT            : out INTEGER_TYPE ;
   LIST_OF_ATTRIBUTE_DESIGNATORS     : out INHERITANCE_VALUE_ARRAY_TYPE ) ;
```

Function INQUIRE CLIPPING INHERITANCE

```
procedure INQ_CLIPPING_INHERITANCE(
   RESPONSE_VALIDITY          : out VALIDITY_FLAG_TYPE ;
   CLIP_INHERITANCE_FILTER    : out CLIP_INHERITANCE_FILTER_TYPE ) ;
```

Function INQUIRE LIST OF SEGMENT IDENTIFIERS IN USE

```
procedure INQ_SEGMENT_IDS_IN_USE(
   NUM_OF_LIST_ELEMENTS_REQUESTED    : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN  : in  INTEGER_TYPE ;
```

```
    RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_PRESENT           : out INTEGER_TYPE ;
    LIST_OF_SEGMENT_IDS              : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE INDIVIDUAL SEGMENT STATE

```
procedure INQ_INDIV_SEGMENT_STATE(
    SEGMENT_ID                   : in  SEGMENT_ID_TYPE ;
    RESPONSE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
    SEGMENT_TRAN                 : out SEGMENT_TRANSFORMATION_TYPE ;
    SEGMENT_VISIBILITY           : out VISIBILITY_TYPE ;
    SEGMENT_HIGHLIGHTING         : out HIGHLIGHTING_TYPE ;
    SEGMENT_DISPLAY_PRIORITY     : out INTEGER_TYPE ;
    SEGMENT_DETECTABILITY        : out DETECTABILITY_TYPE ;
    SEGMENT_PICK_PRIORITY        : out INTEGER_TYPE ) ;
```

7.4 Part 5 input functions

Function INITIALIZE LOGICAL INPUT DEVICE

```
procedure INITIALIZE_LOGICAL_INPUT_DEVICE(
   INPUT_CLASS          : in INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX : in INDEX_TYPE ) ;
```

Function RELEASE LOGICAL INPUT DEVICE

```
procedure RELEASE_LOGICAL_INPUT_DEVICE(
   INPUT_CLASS          : in INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX : in INDEX_TYPE ) ;
```

Function ECHO CONTROLS

```
procedure SET_ECHO_CONTROLS(
   INPUT_CLASS          : in INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX : in INDEX_TYPE ;
   PROMPT_CONTROL     : in DISABLED_ENABLED_FLAG_TYPE ;
   ECHO_CONTROL       : in DISABLED_ENABLED_FLAG_TYPE ;
   ACK_CONTROL        : in DISABLED_ENABLED_FLAG_TYPE ) ;
```

Function PUT CURRENT LOCATOR MEASURE

```
procedure PUT_CURRENT_LOCATOR_MEASURE(
   INPUT_DEVICE_INDEX : in INDEX_TYPE ;
   MEASURE_VALIDITY   : in VALIDITY_FLAG_TYPE ;
   MEASURE_VALUE      : in VDC_POINT_TYPE ) ;
```

Function PUT CURRENT STROKE MEASURE

```
procedure PUT_CURRENT_STROKE_MEASURE(
   INPUT_DEVICE_INDEX : in INDEX_TYPE ;
   MEASURE_VALIDITY   : in VALIDITY_FLAG_TYPE ;
   MEASURE_VALUE      : in POINT_LIST_TYPE ) ;
```

Function PUT CURRENT VALUATOR MEASURE

```
procedure PUT_CURRENT_VALUATOR_MEASURE(
   INPUT_DEVICE_INDEX : in INDEX_TYPE ;
   MEASURE_VALIDITY   : in VALIDITY_FLAG_TYPE ;
   MEASURE_VALUE      : in REAL_TYPE ) ;
```

Function PUT CURRENT CHOICE MEASURE

```
procedure PUT_CURRENT_CHOICE_MEASURE(
  INPUT_DEVICE_INDEX : in INDEX_TYPE ;
  MEASURE_VALIDITY   : in VALIDITY_FLAG_TYPE ;
  MEASURE_VALUE      : in INTEGER_TYPE ) ;
```

Function PUT CURRENT PICK MEASURE

```
procedure PUT_CURRENT_PICK_MEASURE(
  INPUT_DEVICE_INDEX : in INDEX_TYPE ;
  MEASURE_VALIDITY   : in VALIDITY_FLAG_TYPE ;
  MEASURE_VALUE      : in PICK_VALUE_ARRAY_TYPE ) ;
```

Function PUT CURRENT STRING MEASURE

```
procedure PUT_CURRENT_STRING_MEASURE(
  INPUT_DEVICE_INDEX : in INDEX_TYPE ;
  MEASURE_VALIDITY   : in VALIDITY_FLAG_TYPE ;
  MEASURE_VALUE      : in STRING_TYPE ) ;
```

Function PUT CURRENT RASTER MEASURE

```
procedure PUT_CURRENT_RASTER_MEASURE(
  INPUT_DEVICE_INDEX              : in INDEX_TYPE ;
  MEASURE_VALIDITY                : in VALIDITY_FLAG_TYPE ;
  XCOUNT                          : in INTEGER_TYPE ;
  YCOUNT                          : in INTEGER_TYPE ;
  LIST_OF_INPUT_COLOUR_VALUES : in INPUT_COLOUR_ARRAY_TYPE ) ;
```

Function PUT CURRENT GENERAL MEASURE

```
procedure PUT_CURRENT_GENERAL_MEASURE(
  INPUT_DEVICE_INDEX : in INDEX_TYPE ;
  MEASURE_VALIDITY   : in VALIDITY_FLAG_TYPE ;
  MEASURE_VALUE      : in DATA_RECORD_TYPE ) ;
```

Function ECHO DATA

```
procedure SET_ECHO_DATA(
  INPUT_CLASS        : in INPUT_CLASS_TYPE ;
  INPUT_DEVICE_INDEX : in INDEX_TYPE ;
```

```
   PROMPT_TYPE          : in INDEX_TYPE ;
   ECHO_TYPE            : in INDEX_TYPE ;
   ACK_TYPE             : in INDEX_TYPE ;
   ECHO_AREA            : in ECHO_AREA_TYPE ;
   ECHO_DATA_RECORD     : in DATA_RECORD_TYPE ) ;
```

Function LOCATOR DEVICE DATA

```
procedure SET_LOCATOR_DEVICE_DATA(
   INPUT_DEVICE_INDEX : in INDEX_TYPE ;
   INPUT_EXTENT       : in INPUT_EXTENT_TYPE ;
   INPUT_VIEWPORT     : in INPUT_VIEWPORT_TYPE ;
   ECHO_VIEWPORT      : in ECHO_AREA_TYPE ;
   DATA_RECORD        : in DATA_RECORD_TYPE ) ;
```

Function STROKE DEVICE DATA

```
procedure SET_STROKE_DEVICE_DATA(
   INPUT_DEVICE_INDEX                : in INDEX_TYPE ;
   MAX_NUM_OF_POINTS                 : in INTEGER_TYPE ;
   SAMPLING_INTERVAL                 : in STROKE_SAMPLING_INTERVAL_TYPE ;
   MIN_TIME_INTERVAL_PER_SAMPLE      : in INTEGER_TYPE ;
   MAX_TIME_INTERVAL_PER_SAMPLE      : in INTEGER_TYPE ;
   INPUT_EXTENT                      : in INPUT_EXTENT_TYPE ;
   INPUT_VIEWPORT                    : in INPUT_VIEWPORT_TYPE ;
   ECHO_VIEWPORT                     : in ECHO_AREA_TYPE ;
   DATA_RECORD                       : in DATA_RECORD_TYPE ) ;
```

Function VALUATOR DEVICE DATA

```
procedure SET_VALUATOR_DEVICE_DATA(
   INPUT_DEVICE_INDEX    : in INDEX_TYPE ;
   MIN_VALUE_OF_RANGE    : in REAL_TYPE ;
   MAX_VALUE_OF_RANGE    : in REAL_TYPE ;
   DATA_RECORD           : in DATA_RECORD_TYPE ) ;
```

Function CHOICE DEVICE DATA

```
procedure SET_CHOICE_DEVICE_DATA(
   INPUT_DEVICE_INDEX                : in INDEX_TYPE ;
   MAX_NUM_OF_CHOICE_ALTERNATIVES    : in INTEGER_TYPE ;
   DATA_RECORD                       : in DATA_RECORD_TYPE ) ;
```

Function PICK DEVICE DATA

```
procedure SET_PICK_DEVICE_DATA(
   INPUT_DEVICE_INDEX            : in INDEX_TYPE ;
   PICK_APERTURE                 : in PICK_APERTURE_TYPE ;
   MAX_NUM_OF_SEGMENT_PICKS      : in INTEGER_TYPE ;
   INPUT_EXTENT                  : in INPUT_EXTENT_TYPE ;
   INPUT_VIEWPORT                : in INPUT_VIEWPORT_TYPE ;
   ECHO_VIEWPORT                 : in ECHO_AREA_TYPE ;
   DATA_RECORD                   : in DATA_RECORD_TYPE ) ;
```

Function STRING DEVICE DATA

```
procedure SET_STRING_DEVICE_DATA(
   INPUT_DEVICE_INDEX                : in INDEX_TYPE ;
   MAX_STRING_SIZE                   : in INTEGER_TYPE ;
   INPUT_CHAR_SET_INDEX              : in INDEX_TYPE ;
   ALTERNATE_INPUT_CHAR_SET_INDEX    : in INDEX_TYPE ;
   INPUT_CHAR_CODING_ANNOUNCER       : in CODING_TECHNIQUE_TYPE ;
   DATA_RECORD                       : in DATA_RECORD_TYPE ) ;
```

Function RASTER DEVICE DATA

```
procedure SET_RASTER_DEVICE_DATA(
   INPUT_DEVICE_INDEX                : in INDEX_TYPE ;
   SPOT_CENTRE_SEPARATIONS           : in SPOT_CENTRE_SEPARATION_TYPE ;
   COLOUR_CAPABILITY                 : in YES_NO_FLAG_TYPE;
   THRESHOLD_LEVEL                   : in INTEGER_TYPE ;
   BITS_PER_COLOUR                   : in INTEGER_TYPE ;
   SOURCE_WINDOW_FIRST_CORNER        : in RASTER_WINDOW_CORNER_TYPE ;
   SOURCE_WINDOW_SECOND_CORNER       : in RASTER_WINDOW_CORNER_TYPE ;
   DATA_RECORD                       : in DATA_RECORD_TYPE ) ;
```

Function GENERAL DEVICE DATA

```
procedure SET_GENERAL_DEVICE_DATA(
   INPUT_DEVICE_INDEX      : in INDEX_TYPE ;
   MAX_DATA_RECORD_SIZE    : in INTEGER_TYPE ;
   MEASURE_FORMAT_ID       : in INTEGER_TYPE ;
   DATA_RECORD             : in DATA_RECORD_TYPE ) ;
```

Function ASSOCIATE TRIGGERS

```
procedure ASSOCIATE_TRIGGERS(
   INPUT_CLASS          : in INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX   : in INDEX_TYPE ;
```

```
    LIST_OF_TRIGGERS      : in INDEX_ARRAY_TYPE ) ;
```

Function GET ADDITIONAL STROKE DATA

```
procedure GET_ADDITIONAL_STROKE_DATA(
   NUM_OF_REQUESTED_POINTS : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY       : out VALIDITY_FLAG_TYPE ;
   LIST_OF_POINTS          : out POINT_LIST_TYPE ) ;
```

Function GET ADDITIONAL PICK DATA

```
procedure GET_ADDITIONAL_PICK_DATA(
   NUM_OF_REQUESTED_PICK_VALUES : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
   LIST_OF_PICK_VALUES          : out PICK_VALUE_ARRAY_TYPE ) ;
```

Function GET ADDITIONAL STRING DATA

```
procedure GET_ADDITIONAL_STRING_DATA(
   NUM_OF_REQUESTED_CHARS : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY      : out VALIDITY_FLAG_TYPE ;
   STRING_DATA            : out STRING_TYPE ) ;
```

Function GET ADDITIONAL RASTER DATA

```
procedure GET_ADDITIONAL_RASTER_DATA(
   NUM_OF_REQUESTED_COLOUR_VALUES : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY              : out VALIDITY_FLAG_TYPE ;
   LIST_OF_INPUT_COLOUR_VALUES    : out INPUT_COLOUR_ARRAY_TYPE ) ;
```

Function REQUEST LOCATOR

```
procedure REQUEST_LOCATOR(
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   TIMEOUT            : in  REAL_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS     : out REQUEST_STATUS_TYPE ;
   TRIGGER            : out INDEX_TYPE ;
   MEASURE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
   POSITION           : out VDC_POINT_TYPE ) ;
```

Function REQUEST STROKE

```
procedure REQUEST_STROKE(
   INPUT_DEVICE_INDEX          : in  INDEX_TYPE ;
   TIMEOUT                     : in  REAL_TYPE ;
   NUM_OF_REQUESTED_POINTS     : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS              : out REQUEST_STATUS_TYPE ;
   TRIGGER                     : out INDEX_TYPE ;
   MEASURE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_POINTS         : out INTEGER_TYPE ;
   LIST_OF_POINTS              : out POINT_LIST_TYPE ) ;
```

Function REQUEST VALUATOR

```
procedure REQUEST_VALUATOR(
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   TIMEOUT            : in  REAL_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS     : out REQUEST_STATUS_TYPE ;
   TRIGGER            : out INDEX_TYPE ;
   MEASURE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
   VALUE              : out REAL_TYPE ) ;
```

Function REQUEST CHOICE

```
procedure REQUEST_CHOICE(
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   TIMEOUT            : in  REAL_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS     : out REQUEST_STATUS_TYPE ;
   TRIGGER            : out INDEX_TYPE ;
   MEASURE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
   CHOICE_NUMBER      : out INTEGER_TYPE ) ;
```

Function REQUEST PICK

```
procedure REQUEST_PICK(
   INPUT_DEVICE_INDEX          : in  INDEX_TYPE ;
   TIMEOUT                     : in  REAL_TYPE ;
   NUM_OF_REQUESTED_PICK_VALUES : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS              : out REQUEST_STATUS_TYPE ;
   TRIGGER                     : out INDEX_TYPE ;
   MEASURE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_PICK_VALUES    : out INTEGER_TYPE ;
   LIST_OF_PICK_VALUES         : out PICK_VALUE_ARRAY_TYPE ) ;
```

Function REQUEST STRING

```
procedure REQUEST_STRING(
   INPUT_DEVICE_INDEX          : in  INDEX_TYPE ;
   TIMEOUT                     : in  REAL_TYPE ;
   NUM_OF_REQUESTED_CHARS      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS              : out REQUEST_STATUS_TYPE ;
   TRIGGER                     : out INDEX_TYPE ;
   MEASURE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_CHARS          : out INTEGER_TYPE ;
   STRING_MEASURE              : out STRING_TYPE ) ;
```

Function REQUEST RASTER

```
procedure REQUEST_RASTER(
   INPUT_DEVICE_INDEX                  : in  INDEX_TYPE ;
   TIMEOUT                             : in  REAL_TYPE ;
   NUM_OF_REQUESTED_INPUT_COLOUR_VALUES : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS                      : out REQUEST_STATUS_TYPE ;
   TRIGGER                             : out INDEX_TYPE ;
   MEASURE_VALIDITY                    : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_INPUT_COLOUR_VALUES    : out INTEGER_TYPE ;
   XCOUNT                              : out INTEGER_TYPE ;
   YCOUNT                              : out INTEGER_TYPE ;
   LIST_OF_INPUT_COLOUR_VALUES         : out INPUT_COLOUR_ARRAY_TYPE ) ;
```

Function REQUEST GENERAL

```
procedure REQUEST_GENERAL(
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   TIMEOUT            : in  REAL_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS     : out REQUEST_STATUS_TYPE ;
   TRIGGER            : out INDEX_TYPE ;
   MEASURE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
   DATA_RECORD        : out DATA_RECORD_TYPE ) ;
```

Function SAMPLING STATE

```
procedure SET_SAMPLING_STATE(
   INPUT_CLASS        : in INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX : in INDEX_TYPE ;
   SAMPLE_STATE       : in DISABLED_ENABLED_FLAG_TYPE ) ;
```

Function SAMPLE LOCATOR

```
procedure SAMPLE_LOCATOR(
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
   POSITION           : out VDC_POINT_TYPE ) ;
```

Function SAMPLE STROKE

```
procedure SAMPLE_STROKE(
   INPUT_DEVICE_INDEX          : in  INDEX_TYPE ;
   NUM_OF_REQUESTED_POINTS     : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_POINTS         : out INTEGER_TYPE ;
   LIST_OF_POINTS              : out POINT_LIST_TYPE ) ;
```

Function SAMPLE VALUATOR

```
procedure SAMPLE_VALUATOR(
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
   VALUE              : out REAL_TYPE ) ;
```

Function SAMPLE CHOICE

```
procedure SAMPLE_CHOICE(
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
   CHOICE_NUMBER      : out INTEGER_TYPE ) ;
```

Function SAMPLE PICK

```
procedure SAMPLE_PICK(
   INPUT_DEVICE_INDEX              : in  INDEX_TYPE ;
   NUM_OF_REQUESTED_PICK_VALUES    : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY               : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY                : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_PICK_VALUES        : out INTEGER_TYPE ;
   LIST_OF_PICK_VALUES             : out PICK_VALUE_ARRAY_TYPE ) ;
```

Function SAMPLE STRING

```
procedure SAMPLE_STRING(
   INPUT_DEVICE_INDEX          : in  INDEX_TYPE ;
   NUM_OF_REQUESTED_CHARS      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_CHARS          : out INTEGER_TYPE ;
   STRING_MEASURE              : out STRING_TYPE ) ;
```

Function SAMPLE RASTER

```
procedure SAMPLE_RASTER(
   INPUT_DEVICE_INDEX                          : in  INDEX_TYPE ;
   NUM_OF_REQUESTED_INPUT_COLOUR_VALUES        : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                           : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY                            : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_INPUT_COLOUR_VALUES            : out INTEGER_TYPE ;
   XCOUNT                                      : out INTEGER_TYPE ;
   YCOUNT                                      : out INTEGER_TYPE ;
   LIST_OF_INPUT_COLOUR_VALUES                 : out INPUT_COLOUR_ARRAY_TYPE ) ;
```

Function SAMPLE GENERAL

```
procedure SAMPLE_GENERAL(
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
   DATA_RECORD        : out DATA_RECORD_TYPE ) ;
```

Function INITIALIZE ECHO REQUEST

```
procedure INITIALIZE_ECHO_REQUEST(
   INPUT_CLASS        : in INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX : in INDEX_TYPE ;
   TIMEOUT            : in REAL_TYPE ) ;
```

Function ECHO REQUEST LOCATOR

```
procedure ECHO_REQUEST_LOCATOR(
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS     : out REQUEST_STATUS_TYPE ;
```

```
    TRIGGER                 : out INDEX_TYPE ;
    MEASURE_VALIDITY        : out VALIDITY_FLAG_TYPE ;
    POSITION                : out VDC_POINT_TYPE ) ;
```

Function ECHO REQUEST STROKE

```
procedure ECHO_REQUEST_STROKE(
    INPUT_DEVICE_INDEX          : in  INDEX_TYPE ;
    NUM_OF_REQUESTED_POINTS     : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS              : out REQUEST_STATUS_TYPE ;
    TRIGGER                     : out INDEX_TYPE ;
    MEASURE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_POINTS         : out INTEGER_TYPE ;
    LIST_OF_POINTS              : out POINT_LIST_TYPE ) ;
```

Function ECHO REQUEST VALUATOR

```
procedure ECHO_REQUEST_VALUATOR(
    INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
    RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS     : out REQUEST_STATUS_TYPE ;
    TRIGGER            : out INDEX_TYPE ;
    MEASURE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
    VALUE              : out REAL_TYPE ) ;
```

Function ECHO REQUEST CHOICE

```
procedure ECHO_REQUEST_CHOICE(
    INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
    RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS     : out REQUEST_STATUS_TYPE ;
    TRIGGER            : out INDEX_TYPE ;
    MEASURE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
    CHOICE_NUMBER      : out INTEGER_TYPE ) ;
```

Function ECHO REQUEST PICK

```
procedure ECHO_REQUEST_PICK(
    INPUT_DEVICE_INDEX          : in  INDEX_TYPE ;
    NUM_OF_REQUESTED_PICK_VALUES : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS              : out REQUEST_STATUS_TYPE ;
    TRIGGER                     : out INDEX_TYPE ;
    MEASURE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
```

```
   TOTAL_NUM_OF_PICK_VALUES      : out INTEGER_TYPE ;
   LIST_OF_PICK_VALUES           : out PICK_VALUE_ARRAY_TYPE ) ;
```

Function ECHO REQUEST STRING

```
procedure ECHO_REQUEST_STRING(
   INPUT_DEVICE_INDEX          : in  INDEX_TYPE ;
   NUM_OF_REQUESTED_CHARS      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS              : out REQUEST_STATUS_TYPE ;
   TRIGGER                     : out INDEX_TYPE ;
   MEASURE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_CHARS          : out INTEGER_TYPE ;
   STRING_MEASURE              : out STRING_TYPE ) ;
```

Function ECHO REQUEST RASTER

```
procedure ECHO_REQUEST_RASTER(
   INPUT_DEVICE_INDEX                        : in  INDEX_TYPE ;
   NUM_OF_REQUESTED_INPUT_COLOUR_VALUES      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                         : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS                            : out REQUEST_STATUS_TYPE ;
   TRIGGER                                   : out INDEX_TYPE ;
   MEASURE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_INPUT_COLOUR_VALUES          : out INTEGER_TYPE ;
   XCOUNT                                    : out INTEGER_TYPE ;
   YCOUNT                                    : out INTEGER_TYPE ;
   LIST_OF_INPUT_COLOUR_VALUES               : out INPUT_COLOUR_ARRAY_TYPE ) ;
```

Function ECHO REQUEST GENERAL

```
procedure ECHO_REQUEST_GENERAL(
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   REQUEST_STATUS     : out REQUEST_STATUS_TYPE ;
   TRIGGER            : out INDEX_TYPE ;
   MEASURE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
   DATA_RECORD        : out DATA_RECORD_TYPE ) ;
```

Function INITIALIZE EVENT QUEUE

```
procedure INITIALIZE_EVENT_QUEUE(
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   INITIAL_TIME_STAMP : out REAL_TYPE ) ;
```

Function RELEASE EVENT QUEUE

```
procedure RELEASE_EVENT_QUEUE ;
```

Function ENABLE EVENTS

```
procedure ENABLE_EVENTS(
   INPUT_CLASS          : in INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX : in INDEX_TYPE ) ;
```

Function DISABLE EVENTS

```
procedure DISABLE_EVENTS(
   INPUT_CLASS          : in INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX : in INDEX_TYPE ) ;
```

Function EVENT QUEUE BLOCK CONTROL

```
procedure EVENT_QUEUE_BLOCK_CONTROL(
   CONTROL : in BLOCK_CONTROL_TYPE ) ;
```

Function FLUSH EVENTS

```
procedure FLUSH_EVENTS ;
```

Function FLUSH DEVICE EVENTS

```
procedure FLUSH_DEVICE_EVENTS(
   INPUT_CLASS          : in INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX : in INDEX_TYPE ) ;
```

Function AWAIT EVENT

```
procedure AWAIT_EVENT(
   TIMEOUT                 : in  REAL_TYPE ;
   RESPONSE_VALIDITY       : out VALIDITY_FLAG_TYPE ;
   EVENT_STATUS            : out EVENT_STATUS_TYPE ;
   INPUT_CLASS             : out INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX      : out INDEX_TYPE ;
   INPUT_TRIGGER_INDEX : out INDEX_TYPE ;
   TIMESTAMP               : out REAL_TYPE ) ;
```

Function DEQUEUE LOCATOR EVENT

```
procedure DEQUEUE_LOCATOR_EVENT(
   RESPONSE_VALIDITY : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   POSITION          : out VDC_POINT_TYPE ) ;
```

Function DEQUEUE STROKE EVENT

```
procedure DEQUEUE_STROKE_EVENT(
   NUM_OF_REQUESTED_POINTS : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY       : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY        : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_POINTS     : out INTEGER_TYPE ;
   LIST_OF_POINTS          : out POINT_LIST_TYPE ) ;
```

Function DEQUEUE VALUATOR EVENT

```
procedure DEQUEUE_VALUATOR_EVENT(
   RESPONSE_VALIDITY : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   VALUE             : out REAL_TYPE ) ;
```

Function DEQUEUE CHOICE EVENT

```
procedure DEQUEUE_CHOICE_EVENT(
   RESPONSE_VALIDITY : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   CHOICE_NUMBER     : out INTEGER_TYPE ) ;
```

Function DEQUEUE PICK EVENT

```
procedure DEQUEUE_PICK_EVENT(
   NUM_OF_REQUESTED_PICK_VALUES : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY            : out VALIDITY_FLAG_TYPE ;
   MEASURE_VALIDITY             : out VALIDITY_FLAG_TYPE ;
   TOTAL_NUM_OF_PICK_VALUES     : out INTEGER_TYPE ;
   LIST_OF_PICK_VALUES          : out PICK_VALUE_ARRAY_TYPE ) ;
```

Function DEQUEUE STRING EVENT

```
procedure DEQUEUE_STRING_EVENT(
   NUM_OF_REQUESTED_CHARS : in  INTEGER_TYPE ;
```

```
    RESPONSE_VALIDITY          : out VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY           : out VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_CHARS         : out INTEGER_TYPE ;
    STRING_MEASURE             : out STRING_TYPE ) ;
```

Function DEQUEUE RASTER EVENT

```
procedure DEQUEUE_RASTER_EVENT(
    NUM_OF_REQUESTED_INPUT_COLOUR_VALUES : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_INPUT_COLOUR_VALUES     : out INTEGER_TYPE ;
    XCOUNT                               : out INTEGER_TYPE ;
    YCOUNT                               : out INTEGER_TYPE ;
    LIST_OF_INPUT_COLOUR_VALUES          : out INPUT_COLOUR_ARRAY_TYPE ) ;
```

Function DEQUEUE GENERAL EVENT

```
procedure DEQUEUE_GENERAL_EVENT(
    RESPONSE_VALIDITY : out VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
    DATA_RECORD       : out DATA_RECORD_TYPE ) ;
```

Function EVENT QUEUE TRANSFER

```
procedure EVENT_QUEUE_TRANSFER(
    RESPONSE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
    EVENT_REPORTS_LIST  : out EVENT_REPORTS_LIST_TYPE ) ;
```

Function INITIALIZE ECHO OUTPUT

```
procedure INITIALIZE_ECHO_OUTPUT(
    INPUT_CLASS       : in INPUT_CLASS_TYPE ;
    ECHO_ENTITY_INDEX : in INDEX_TYPE ) ;
```

Function RELEASE ECHO OUTPUT

```
procedure RELEASE_ECHO_OUTPUT(
    INPUT_CLASS       : in INPUT_CLASS_TYPE ;
    ECHO_ENTITY_INDEX : in INDEX_TYPE ) ;
```

Function ECHO OUTPUT CONTROLS

```
procedure SET_ECHO_OUTPUT_CONTROLS(
  INPUT_CLASS          : in INPUT_CLASS_TYPE ;
  ECHO_ENTITY_INDEX    : in INDEX_TYPE ;
  PROMPT_CONTROL       : in ECHO_ENTITY_PROMPT_CONTROL_TYPE ;
  ECHO_CONTROL         : in ECHO_ENTITY_ECHO_CONTROL_TYPE ) ;
```

Function PERFORM ACKNOWLEDGEMENT

```
procedure PERFORM_ACK(
  INPUT_CLASS          : in INPUT_CLASS_TYPE ;
  ECHO_ENTITY_INDEX    : in INDEX_TYPE ) ;
```

Function UPDATE LOCATOR ECHO OUTPUT

```
procedure UPDATE_LOCATOR_ECHO_OUTPUT(
  ECHO_ENTITY_INDEX    : in INDEX_TYPE ;
  POSITION             : in VDC_POINT_TYPE ) ;
```

Function UPDATE STROKE ECHO OUTPUT

```
procedure UPDATE_STROKE_ECHO_OUTPUT(
  ECHO_ENTITY_INDEX    : in INDEX_TYPE ;
  LIST_OF_POINTS       : in POINT_LIST_TYPE ) ;
```

Function UPDATE VALUATOR ECHO OUTPUT

```
procedure UPDATE_VALUATOR_ECHO_OUTPUT(
  ECHO_ENTITY_INDEX    : in INDEX_TYPE ;
  VALUE                : in REAL_TYPE ) ;
```

Function UPDATE CHOICE ECHO OUTPUT

```
procedure UPDATE_CHOICE_ECHO_OUTPUT(
  ECHO_ENTITY_INDEX    : in INDEX_TYPE ;
  CHOICE_NUMBER        : in INTEGER_TYPE ) ;
```

Function UPDATE PICK ECHO OUTPUT

```
procedure UPDATE_PICK_ECHO_OUTPUT(
  ECHO_ENTITY_INDEX    : in INDEX_TYPE ;
  LIST_OF_PICK_VALUES  : in PICK_VALUE_ARRAY_TYPE ) ;
```

Function UPDATE STRING ECHO OUTPUT

```
procedure UPDATE_STRING_ECHO_OUTPUT(
   ECHO_ENTITY_INDEX : in INDEX_TYPE ;
   STRING_ECHO       : in STRING_TYPE ) ;
```

Function UPDATE RASTER ECHO OUTPUT

```
procedure UPDATE_RASTER_ECHO_OUTPUT(
   ECHO_ENTITY_INDEX                : in INDEX_TYPE ;
   XCOUNT                           : in INTEGER_TYPE ;
   YCOUNT                           : in INTEGER_TYPE ;
   LIST_OF_INPUT_COLOUR_VALUES : in INPUT_COLOUR_ARRAY_TYPE ) ;
```

Function UPDATE GENERAL ECHO OUTPUT

```
procedure UPDATE_GENERAL_ECHO_OUTPUT(
   ECHO_ENTITY_INDEX  : in INDEX_TYPE ;
   MEASURE_FORMAT_ID : in INTEGER_TYPE ;
   DATA_RECORD        : in DATA_RECORD_TYPE ) ;
```

Function ECHO OUTPUT DATA

```
procedure SET_ECHO_OUTPUT_DATA(
   INPUT_CLASS                      : in INPUT_CLASS_TYPE ;
   ECHO_ENTITY_INDEX                : in INDEX_TYPE ;
   PROMPT_TYPE                      : in INDEX_TYPE ;
   ECHO_TYPE                        : in INDEX_TYPE ;
   ACK_TYPE                         : in INDEX_TYPE ;
   ECHO_AREA                        : in ECHO_AREA_TYPE ;
   ECHO_OUTPUT_ECHO_DATA_RECORD : in DATA_RECORD_TYPE ) ;
```

Function INQUIRE INPUT CAPABILITY

```
procedure INQ_INPUT_CAPABILITY(
   RESPONSE_VALIDITY          : out VALIDITY_FLAG_TYPE ;
   TIMEOUT_CAP                : out TIMEOUT_CAP_TYPE ;
   TIMESTAMP_IMPLEMENTATION : out TIMESTAMP_SUPPORT_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE INPUT DEVICES

```
procedure INQ_AVAIL_INPUT_DEVICES(
   INPUT_CLASS                      : in  INPUT_CLASS_TYPE ;
```

```
   NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE   : out INTEGER_TYPE ;
   LIST_OF_AVAIL_INPUT_DEVICE_INDICES    : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE COMMON INPUT DEVICE PROPERTIES

```
procedure INQ_COMMON_INPUT_DEVICE_PROPERTIES(
   INPUT_CLASS                           : in  INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX                    : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
   PHYSICAL_DEVICE_ID                    : out INTEGER_TYPE ;
   NUM_OF_NON_DISSOCIABLE_TRIGGERS       : out INTEGER_TYPE ;
   REQUEST_SUPPORT                       : out YES_NO_FLAG_TYPE ;
   ECHO_REQUEST_SUPPORT                  : out YES_NO_FLAG_TYPE ;
   SAMPLE_SUPPORT                        : out YES_NO_FLAG_TYPE ;
   EVENT_SUPPORT                         : out YES_NO_FLAG_TYPE ;
   PUT_CURRENT_MEASURE_EFFECTIVE         : out YES_NO_FLAG_TYPE ;
   ECHO_CHANGE_SUPPORT                   : out YES_NO_FLAG_TYPE ) ;
```

Function INQUIRE LIST OF SUPPORTED ECHO TYPES

```
procedure INQ_SUPPORTED_ECHO_TYPES(
   INPUT_CLASS                           : in  INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX                    : in  INDEX_TYPE ;
   NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE   : out INTEGER_TYPE ;
   LIST_OF_ECHO_TYPES                    : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF SUPPORTED PROMPT TYPES

```
procedure INQ_SUPPORTED_PROMPT_TYPES(
   INPUT_CLASS                           : in  INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX                    : in  INDEX_TYPE ;
   NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE   : out INTEGER_TYPE ;
   LIST_OF_PROMPT_TYPES                  : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF SUPPORTED ACKNOWLEDGEMENT TYPES

```
procedure INQ_SUPPORTED_ACK_TYPES(
   INPUT_CLASS                              : in  INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX                       : in  INDEX_TYPE ;
   NUM_OF_LIST_ELEMENTS_REQUESTED           : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN         : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                        : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
   LIST_OF_ACK_TYPES                        : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF ASSOCIABLE TRIGGERS

```
procedure INQ_ASSOCIABLE_TRIGGERS(
   INPUT_CLASS                              : in  INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX                       : in  INDEX_TYPE ;
   NUM_OF_LIST_ELEMENTS_REQUESTED           : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN         : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                        : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
   LIST_OF_TRIGGERS                         : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LOCATOR CAPABILITIES

```
procedure INQ_LOCATOR_CAPABILITIES(
   INPUT_DEVICE_INDEX               : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE;
   LOWER_LEFT_CORNER                : out INPUT_SURFACE_POINT_TYPE ;
   UPPER_RIGHT_CORNER               : out INPUT_SURFACE_POINT_TYPE ;
   PHYSICAL_INPUT_SURFACE_SIZE      : out PHYSICAL_INPUT_SURFACE_SIZE_TYPE;
   INPUT_SURFACE_INTERPRETATION     : out SURFACE_INTERPRETATION_TYPE ;
   NUM_DISTINGUISHABLE_XY_STEPS     : out XY_STEP_TYPE ) ;
```

Function INQUIRE STROKE CAPABILITIES

```
procedure INQ_STROKE_CAPABILITIES(
   INPUT_DEVICE_INDEX               : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE;
   MAX_NUM_OF_POINTS                : out INTEGER_TYPE ;
   LOWER_LEFT_CORNER                : out INPUT_SURFACE_POINT_TYPE ;
   UPPER_RIGHT_CORNER               : out INPUT_SURFACE_POINT_TYPE ;
   PHYSICAL_INPUT_SURFACE_SIZE      : out PHYSICAL_INPUT_SURFACE_SIZE_TYPE;
   INPUT_SURFACE_INTERPRETATION     : out SURFACE_INTERPRETATION_TYPE ;
   NUM_DISTINGUISHABLE_XY_STEPS     : out XY_STEP_TYPE ) ;
```

Function INQUIRE CHOICE CAPABILITIES

```
procedure INQ_CHOICE_CAPABILITIES(
  INPUT_DEVICE_INDEX                  : in  INDEX_TYPE ;
  RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
  MAX_NUM_OF_CHOICE_ALTERNATIVES : out INTEGER_TYPE ) ;
```

Function INQUIRE PICK CAPABILITIES

```
procedure INQ_PICK_CAPABILITIES(
  INPUT_DEVICE_INDEX              : in  INDEX_TYPE ;
  RESPONSE_VALIDITY               : out VALIDITY_FLAG_TYPE;
  MAX_NUM_OF_SEGMENT_PICKS        : out INTEGER_TYPE ;
  LOWER_LEFT_CORNER               : out INPUT_SURFACE_POINT_TYPE ;
  UPPER_RIGHT_CORNER              : out INPUT_SURFACE_POINT_TYPE ;
  PHYSICAL_INPUT_SURFACE_SIZE     : out PHYSICAL_INPUT_SURFACE_SIZE_TYPE;
  INPUT_SURFACE_INTERPRETATION    : out SURFACE_INTERPRETATION_TYPE;
  NUM_DISTINGUISHABLE_XY_STEPS    : out XY_STEP_TYPE ) ;
```

Function INQUIRE STRING CAPABILITIES

```
procedure INQ_STRING_CAPABILITIES(
  INPUT_DEVICE_INDEX              : in  INDEX_TYPE ;
  RESPONSE_VALIDITY               : out VALIDITY_FLAG_TYPE ;
  MAX_STRING_BUFFER_SIZE          : out INTEGER_TYPE ;
  VALID_CHAR_CODING_ANNOUNCERS    : out INTEGER_TYPE ;
  AVAIL_CHAR_CODING_ANNOUNCERS    : out CHAR_CODING_ANNOUNCER_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF AVAILABLE INPUT CHARACTER SETS

```
procedure INQ_AVAIL_INPUT_CHAR_SETS(
  INPUT_DEVICE_INDEX                  : in  INDEX_TYPE ;
  NUM_OF_LIST_ELEMENTS_REQUESTED      : in  INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
  LIST_OF_AVAIL_INPUT_CHAR_SETS       : out CHAR_SET_ARRAY_TYPE ) ;
```

Function INQUIRE RASTER INPUT CAPABILITIES

```
procedure INQ_RASTER_INPUT_CAPABILITIES(
  INPUT_DEVICE_INDEX              : in  INDEX_TYPE ;
  RESPONSE_VALIDITY               : out VALIDITY_FLAG_TYPE ;
  PHYSICAL_INPUT_SURFACE_SIZE     : out PHYSICAL_INPUT_SURFACE_SIZE_TYPE ;
  COLOUR_CAPABILITY               : out YES_NO_FLAG_TYPE ;
  NUMBER_OF_LEVELS                : out INTEGER_TYPE ;
  MAX_NUM_OF_BITS_PER_COLOUR      : out INTEGER_TYPE ;
```

```
    MAX_HEIGHT                      : out INTEGER_TYPE ;
    MAX_WIDTH                       : out INTEGER_TYPE ;
    SPOT_CENTRE_INTERPRETATION      : out SPOT_CENTRE_INTERPRETATION_TYPE ) ;
```

Function INQUIRE LIST OF PERMITTED RASTER SPOT CENTRE SEPARATIONS

```
procedure INQ_PERMITTED_RASTER_SPOT_CENTRE_SEPARATIONS(
    INPUT_DEVICE_INDEX                  : in  INDEX_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED      : in  INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
    LIST_OF_SPOT_CENTRE_SEPARATIONS     : out SPOT_CENTRE_SEPARATION_TYPE ) ;
```

Function INQUIRE GENERAL CAPABILITIES

```
procedure INQ_GENERAL_CAPABILITIES(
     INPUT_DEVICE_INDEX     : in  INDEX_TYPE ;
    RESPONSE_VALIDITY       : out VALIDITY_FLAG_TYPE ;
    MAX_DATA_RECORD_SIZE    : out INTEGER_TYPE ) ;
```

Function INQUIRE LIST OF SUPPORTED GENERAL MEASURE FORMATS

```
procedure INQ_SUPPORTED_GENERAL_MEASURE_FORMATS(
    INPUT_DEVICE_INDEX                  : in  INDEX_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED      : in  INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
    LIST_OF_MEASURE_FORMAT_IDS          : out INTEGER_ARRAY_TYPE ) ;
```

Function INQUIRE COMMON LOGICAL INPUT DEVICE STATE

```
procedure INQ_COMMON_LID_STATE(
    INPUT_CLASS                         : in  INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX                  : in  INDEX_TYPE ;
    RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
    INPUT_DEVICE_STATE                  : out INPUT_DEVICE_STATE_TYPE ;
    SAMPLE_STATE                        : out DISABLED_ENABLED_FLAG_TYPE ;
    ECHO_CONTROL                        : out DISABLED_ENABLED_FLAG_TYPE ;
    ECHO_TYPE                           : out INDEX_TYPE ;
    ECHO_AREA_SPECIF_MODE               : out VIEWPORT_SPECIF_MODE_TYPE ;
    ECHO_AREA_METRIC_SCALE_FACTOR       : out REAL_TYPE ;
    ECHO_AREA                           : out ECHO_AREA_TYPE ;
```

```
   PROMPT_CONTROL                    : out DISABLED_ENABLED_FLAG_TYPE ;
   PROMPT_TYPE                       : out INDEX_TYPE ;
   ACK_CONTROL                       : out DISABLED_ENABLED_FLAG_TYPE ;
   ACK_TYPE                          : out INDEX_TYPE ) ;
```

Function INQUIRE LIST OF ASSOCIATED TRIGGERS

```
procedure INQ_ASSOCIATED_TRIGGERS(
   INPUT_CLASS                                : in  INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX                         : in  INDEX_TYPE ;
   NUM_OF_LIST_ELEMENTS_REQUESTED             : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN           : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
   LIST_OF_ASSOCIATED_TRIGGERS                : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE ECHO DATA RECORD

```
procedure INQ_ECHO_DATA_RECORD(
   INPUT_CLASS          : in  INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   ECHO_DATA_RECORD   : out DATA_RECORD_TYPE ) ;
```

Function INQUIRE INPUT DEVICE DATA RECORD

```
procedure INQ_INPUT_DEVICE_DATA_RECORD(
   INPUT_CLASS          : in  INPUT_CLASS_TYPE ;
   INPUT_DEVICE_INDEX : in  INDEX_TYPE ;
   RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
   DATA_RECORD        : out DATA_RECORD_TYPE ) ;
```

Function INQUIRE LOCATOR STATE

```
procedure INQ_LOCATOR_STATE(
   INPUT_DEVICE_INDEX                         : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   INPUT_EXTENT                               : out INPUT_EXTENT_TYPE ;
   INPUT_VIEWPORT                             : out INPUT_VIEWPORT_TYPE ;
   ECHO_VIEWPORT_SPECIF_MODE                  : out VIEWPORT_SPECIF_MODE_TYPE ;
   ECHO_VIEWPORT_METRIC_SCALE_FACTOR : out REAL_TYPE ;
   ECHO_VIEWPORT                              : out ECHO_AREA_TYPE ) ;
```

Function INQUIRE STROKE STATE

```
procedure INQ_STROKE_STATE(
   INPUT_DEVICE_INDEX                  : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
   MAX_NUM_OF_POINTS                   : out INTEGER_TYPE ;
   SAMPLING_INTERVAL_X_DISPLACEMENT    : out VDC_TYPE ;
   SAMPLING_INTERVAL_Y_DISPLACEMENT    : out VDC_TYPE ;
   MIN_TIME_INTERVAL_PER_SAMPLE        : out REAL_TYPE ;
   MAX_TIME_INTERVAL_PER_SAMPLE        : out REAL_TYPE ;
   INPUT_EXTENT                        : out INPUT_EXTENT_TYPE ;
   INPUT_VIEWPORT                      : out INPUT_VIEWPORT_TYPE ;
   ECHO_VIEWPORT_SPECIF_MODE           : out VIEWPORT_SPECIF_MODE_TYPE ;
   ECHO_VIEWPORT_METRIC_SCALE_FACTOR : out REAL_TYPE ;
   ECHO_VIEWPORT                       : out ECHO_AREA_TYPE ) ;
```

Function INQUIRE VALUATOR STATE

```
procedure INQ_VALUATOR_STATE(
   INPUT_DEVICE_INDEX    : in  INDEX_TYPE ;
   RESPONSE_VALIDITY     : out VALIDITY_FLAG_TYPE ;
   MIN_VALUE_OF_RANGE    : out REAL_TYPE ;
   MAX_VALUE_OF_RANGE : out REAL_TYPE ) ;
```

Function INQUIRE CHOICE STATE

```
procedure INQ_CHOICE_STATE(
   INPUT_DEVICE_INDEX                : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                 : out VALIDITY_FLAG_TYPE ;
   MAX_NUM_OF_CHOICE_ALTERNATIVES : out INTEGER_TYPE ) ;
```

Function INQUIRE PICK STATE

```
procedure INQ_PICK_STATE(
   INPUT_DEVICE_INDEX                  : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
   PICK_APERTURE                       : out PICK_APERTURE_TYPE ;
   MAX_NUM_OF_SEGMENT_PICKS            : out INTEGER_TYPE ;
   INPUT_EXTENT                        : out INPUT_EXTENT_TYPE ;
   INPUT_VIEWPORT                      : out INPUT_VIEWPORT_TYPE ;
   ECHO_VIEWPORT_SPECIF_MODE           : out VIEWPORT_SPECIF_MODE_TYPE ;
   ECHO_VIEWPORT_METRIC_SCALE_FACTOR : out REAL_TYPE ;
   ECHO_VIEWPORT                       : out ECHO_AREA_TYPE ) ;
```

Function INQUIRE STRING STATE

```
procedure INQ_STRING_STATE(
   INPUT_DEVICE_INDEX                  : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
   MAX_STRING_SIZE                     : out INTEGER_TYPE ;
   INPUT_CHAR_SET_INDEX                : out INDEX_TYPE ;
   ALTERNATE_INPUT_CHAR_SET_INDEX : out INDEX_TYPE ;
   INPUT_CHAR_CODING_ANNOUNCER    : out CODING_TECHNIQUE_TYPE ) ;
```

Function INQUIRE RASTER INPUT STATE

```
procedure INQ_RASTER_INPUT_STATE(
   INPUT_DEVICE_INDEX                  : in  INDEX_TYPE ;
   RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
   SPOT_CENTRE_SEPARATIONS             : out SPOT_CENTRE_SEPARATION_TYPE ;
   COLOUR_CAPABILITY                   : out YES_NO_FLAG_TYPE ;
   THRESHOLD_LEVEL                     : out INTEGER_TYPE ;
   BITS_PER_COLOUR                     : out INTEGER_TYPE ;
   SOURCE_WINDOW_FIRST_CORNER     : out RASTER_WINDOW_CORNER_TYPE ;
   SOURCE_WINDOW_SECOND_CORNER : out RASTER_WINDOW_CORNER_TYPE ) ;
```

Function INQUIRE GENERAL STATE

```
procedure INQ_GENERAL_STATE(
   INPUT_DEVICE_INDEX      : in  INDEX_TYPE ;
   RESPONSE_VALIDITY       : out VALIDITY_FLAG_TYPE ;
   MAX_DATA_RECORD_SIZE : out INTEGER_TYPE ;
   MEASURE_FORMAT_ID       : out INTEGER_TYPE ) ;
```

Function INQUIRE EVENT INPUT STATE

```
procedure INQ_EVENT_INPUT_STATE(
   RESPONSE_VALIDITY              : out VALIDITY_FLAG_TYPE ;
   EVENT_QUEUE_STATE              : out EVENT_QUEUE_STATE_TYPE ;
   EVENT_QUEUE_BLOCK_STATE        : out BLOCK_CONTROL_TYPE ;
   UNREPORTED_OVERFLOW_STATE : out UNREPORTED_OVERFLOW_STATE_TYPE ;
   OVERFLOW_DEVICE_CLASS          : out INPUT_CLASS_TYPE ;
   OVERFLOW_DEVICE_INDEX          : out INDEX_TYPE ;
   OVERFLOW_TRIGGER_INDEX         : out INDEX_TYPE ;
   OVERFLOW_TIMESTAMP             : out REAL_TYPE ;
   UNREPORTED_BREAK_STATE         : out UNREPORTED_BREAK_STATE_TYPE ;
   BREAK_DEVICE_CLASS             : out INPUT_CLASS_TYPE ;
   BREAK_DEVICE_INDEX             : out INDEX_TYPE ;
   BREAK_TIMESTAMP                : out REAL_TYPE ) ;
```

Function INQUIRE ECHO OUTPUT CAPABILITIES

```
procedure INQ_ECHO_OUTPUT_CAPABILITIES(
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   MAX_NUM_OF_LOCATOR_ECHO_ENTITIES           : out INTEGER_TYPE ;
   MAX_NUM_OF_STROKE_ECHO_ENTITIES            : out INTEGER_TYPE ;
   MAX_NUM_OF_VALUATOR_ECHO_ENTITIES          : out INTEGER_TYPE ;
   MAX_NUM_OF_CHOICE_ECHO_ENTITIES            : out INTEGER_TYPE ;
   MAX_NUM_OF_PICK_ECHO_ENTITIES              : out INTEGER_TYPE ;
   MAX_NUM_OF_STRING_ECHO_ENTITIES            : out INTEGER_TYPE ;
   MAX_NUM_OF_RASTER_ECHO_ENTITIES            : out INTEGER_TYPE ;
   MAX_NUM_OF_GENERAL_ECHO_ENTITIES           : out INTEGER_TYPE ;
   MAX_NUM_OF_STROKE_POINTS                   : out INTEGER_TYPE ;
   MAX_STRING_BUFFER_SIZE                     : out INTEGER_TYPE ;
   MAX_PIXEL_HEIGHT                           : out INTEGER_TYPE ;
   MAX_PIXEL_WIDTH                            : out INTEGER_TYPE ;
   MAX_GENERAL_INPUT_DATA_RECORD_SIZE : out INTEGER_TYPE ) ;
```

Function INQUIRE LIST OF ECHO OUTPUT ECHO TYPES

```
procedure INQ_ECHO_OUTPUT_ECHO_TYPES(
   INPUT_CLASS                                : in  INPUT_CLASS_TYPE ;
   NUM_OF_LIST_ELEMENTS_REQUESTED             : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN           : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
   LIST_OF_ECHO_TYPES_SUPPORTED               : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF ECHO OUTPUT PROMPT TYPES

```
procedure INQ_ECHO_OUTPUT_PROMPT_TYPES(
   INPUT_CLASS                                : in  INPUT_CLASS_TYPE ;
   NUM_OF_LIST_ELEMENTS_REQUESTED             : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN           : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
   LIST_OF_PROMPT_TYPES_SUPPORTED             : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF ECHO OUTPUT ACKNOWLEDGEMENT TYPES

```
procedure INQ_ECHO_OUTPUT_ACK_TYPES(
   INPUT_CLASS                                : in  INPUT_CLASS_TYPE ;
   NUM_OF_LIST_ELEMENTS_REQUESTED             : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN           : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                          : out VALIDITY_FLAG_TYPE ;
```

```
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
    LIST_OF_ACK_TYPES_SUPPORTED         : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF SUPPORTED GENERAL FORMAT IDENTIFIERS

```
procedure INQ_SUPPORTED_GENERAL_FORMAT_IDS(
    NUM_OF_LIST_ELEMENTS_REQUESTED      : in  INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
    LIST_OF_MEASURE_FORMAT_IDS          : out INTEGER_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF CURRENTLY EXISTING ECHO ENTITIES

```
procedure INQ_CURRENTLY_EXISTING_ECHO_ENTITIES(
    INPUT_CLASS                         : in  INPUT_CLASS_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED      : in  INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
    LIST_OF_CURRENT_INPUT_ECHO_ENTITIES : out INDEX_ARRAY_TYPE ) ;
```

Function INQUIRE ECHO ENTITY STATE

```
procedure INQ_ECHO_ENTITY_STATE(
    INPUT_CLASS                         : in  INPUT_CLASS_TYPE ;
    ECHO_ENTITY_INDEX                   : in  INDEX_TYPE ;
    RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
    ECHO_ENTITY_STATE                   : out ECHO_ENTITY_STATE_TYPE ;
    ECHO_CONTROL                        : out ECHO_ENTITY_ECHO_CONTROL_TYPE ;
    ECHO_TYPE                           : out INDEX_TYPE ;
    PROMPT_CONTROL                      : out ECHO_ENTITY_PROMPT_CONTROL_TYPE;
    PROMPT_TYPE                         : out INDEX_TYPE ;
    ACK_TYPE                            : out INDEX_TYPE ;
    ECHO_AREA_SPECIF_MODE               : out VIEWPORT_SPECIF_MODE_TYPE ;
    ECHO_AREA_METRIC_SCALE_FACTOR       : out REAL_TYPE ;
    ECHO_AREA                           : out ECHO_AREA_TYPE ) ;
```

Function INQUIRE ECHO OUTPUT DATA RECORD

```
procedure INQ_ECHO_OUTPUT_DATA_RECORD(
    INPUT_CLASS         : in  INPUT_CLASS_TYPE ;
    ECHO_ENTITY_INDEX   : in  INDEX_TYPE ;
    RESPONSE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
    ECHO_DATA_RECORD    : out DATA_RECORD_TYPE ) ;
```

7.5 Part 6 raster functions

Function GET NEW BITMAP IDENTIFIER

```
procedure GET_NEW_BITMAP_ID(
   RESPONSE_VALIDITY : out VALIDITY_FLAG_TYPE ;
   BITMAP_ID         : out BITMAP_ID_TYPE ) ;
```

Function CREATE BITMAP

```
procedure CREATE_BITMAP(
   BITMAP_ID                : in BITMAP_ID_TYPE ;
   BITMAP_EXTENT            : in BITMAP_EXTENT_TYPE ;
   BITMAP_DEPTH             : in DEPTH_TYPE ;
   BITMAP_DISPLAYABILITY : in DISPLAYABILITY_TYPE ) ;
```

Function DELETE BITMAP

```
procedure DELETE_BITMAP(
   BITMAP_ID : in BITMAP_ID_TYPE ) ;
```

Function DRAWING BITMAP

```
procedure SET_DRAWING_BITMAP(
   BITMAP_ID : in BITMAP_ID_TYPE ) ;
```

Function DISPLAY BITMAP

```
procedure SET_DISPLAY_BITMAP(
   BITMAP_ID : in BITMAP_ID_TYPE ) ;
```

Function MAPPED BITMAP FOREGROUND COLOUR

```
procedure SET_BITMAP_FOREGROUND_COLOUR(
   COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function MAPPED BITMAP BACKGROUND COLOUR

```
procedure SET_BITMAP_BACKGROUND_COLOUR(
    COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function TRANSPARENT COLOUR

```
procedure SET_TRANSPARENT_COLOUR(
  COLOUR_SPECIFIER : in COLOUR_SPECIFIER_TYPE ) ;
```

Function DRAWING MODE

```
procedure SET_DRAWING_MODE(
  DRAWING_MODE : in DRAWING_MODE_SPECIFIER_TYPE ) ;
```

Function FILL BITMAP

```
procedure SET_FILL_BITMAP(
  BITMAP_ID                : in BITMAP_ID_TYPE ;
  PATTERN_BITMAP_REGION : in BITMAP_REGION_TYPE ) ;
```

Function PIXEL ARRAY

```
procedure PIXEL_ARRAY(
  ORIGIN_POINT                          : in VDC_POINT_TYPE ;
  X_DIMENSION                           : in INTEGER_TYPE ;
  Y_DIMENSION                           : in INTEGER_TYPE ;
  X_SCALE                               : in INTEGER_TYPE ;
  Y_SCALE                               : in INTEGER_TYPE ;
  X_DIRECTION                           : in VDC_DIRECTION_TYPE ;
  Y_DIRECTION                           : in VDC_DIRECTION_TYPE ;
  DRAWING_MODE                          : in DRAWING_MODE_SPECIFIER_TYPE ;
  TRANSPARENCY                          : in TRANSPARENCY_TYPE ;
  LOCAL_COLOUR_PREC_REQUIREMENT : in PREC_REQUIREMENT_TYPE ;
  COLOUR_SPECIFIERS                     : in COLOUR_ARRAY_TYPE ;
  FINAL_FLAG                            : in FINAL_FLAG_TYPE := FINAL ) ;
```

Function GET PIXEL ARRAY

```
procedure GET_PIXEL_ARRAY(
  SOURCE_BITMAP_ID     : in  BITMAP_ID_TYPE ;
  ORIGIN_POINT         : in  VDC_POINT_TYPE ;
  X_DIMENSION          : in  INTEGER_TYPE ;
  Y_DIMENSION          : in  INTEGER_TYPE ;
  X_DIRECTION          : in  VDC_DIRECTION_TYPE ;
  Y_DIRECTION          : in  VDC_DIRECTION_TYPE ;
  LOCAL_COLOUR_PREC    : in  PREC_REQUIREMENT_TYPE ;
  RESPONSE_VALIDITY    : out VALIDITY_FLAG_TYPE ;
  PIXEL_VALIDITY_FLAG : out PIXEL_VALIDITY_FLAG_TYPE ;
```

```
    VALID_X_RANGE          : out VALID_PIXEL_RANGE_TYPE ;
    VALID_Y_RANGE          : out VALID_PIXEL_RANGE_TYPE ;
    COLOUR_SPECIFIERS      : out COLOUR_ARRAY_TYPE ) ;
```

Function GET PIXEL ARRAY DIMENSIONS

```
procedure GET_PIXEL_ARRAY_DIMS(
    SOURCE_BITMAP_ID                    : in  BITMAP_ID_TYPE ;
    REGION                              : in  BITMAP_REGION_TYPE ;
    LOCAL_COLOUR_PREC_REQUIREMENT       : in  PREC_REQUIREMENT_TYPE ;
    RESPONSE_VALIDITY                   : out VALIDITY_FLAG_TYPE ;
    LOCAL_COLOUR_PREC                   : out PREC_REQUIREMENT_TYPE ;
    X_DIMENSION                         : out INTEGER_TYPE ;
    Y_DIMENSION                         : out INTEGER_TYPE ) ;
```

Function SOURCE DESTINATION BITBLT

```
procedure SOURCE_DESTINATION_BITBLT(
    SOURCE_BITMAP_ID    : in BITMAP_ID_TYPE ;
    SOURCE_ORIGIN       : in VDC_POINT_TYPE ;
    DESTINATION_ORIGIN  : in VDC_POINT_TYPE ;
    X_OFFSET            : in VDC_TYPE ;
    Y_OFFSET            : in VDC_TYPE ;
    DRAWING_MODE        : in DRAWING_MODE_SPECIFIER_TYPE ;
    TRANSPARENCY        : in TRANSPARENCY_TYPE ) ;
```

Function TILE THREE OPERAND BITBLT

```
procedure TILE_3_OPERAND_BITBLT(
    PATTERN_BITMAP_ID   : in BITMAP_ID_TYPE ;
    PATTERN_REGION      : in BITMAP_REGION_TYPE ;
    REFERENCE_POINT     : in VDC_POINT_TYPE ;
    SOURCE_BITMAP_ID    : in BITMAP_ID_TYPE ;
    SOURCE_ORIGIN       : in VDC_POINT_TYPE ;
    DESTINATION_ORIGIN  : in VDC_POINT_TYPE ;
    X_OFFSET            : in VDC_TYPE ;
    Y_OFFSET            : in VDC_TYPE ;
    DRAWING_MODE_3      : in DRAWING_MODE_SPECIFIER_TYPE ;
    TRANSPARENCY        : in TRANSPARENCY_TYPE ) ;
```

Function INQUIRE RASTER CAPABILITY

```
procedure INQ_RASTER_CAPABILITY(
    RESPONSE_VALIDITY                           : out VALIDITY_FLAG_TYPE ;
    NUM_OF_PREDEFINED_DISPLAYABLE_BITMAPS       : out INTEGER_TYPE ;
```

```
   DISPLAYABLE_BITMAP_CREATION_SUPPORT     : out YES_NO_FLAG_TYPE ;
   BITMAP_FORMATS_SUPPORTED                : out DEPTHS_SUPPORTED_TYPE ;
   NUM_OF_BITS_PER_FULL_DEPTH_PIXEL        : out INTEGER_TYPE ;
   DRAWING_MODE_TRANSPARENCY_SUPPORT       : out DRAWING_MODE_SUPPORT_TYPE ;
   DRAWING_MODE_3_TRANSPARENCY_SUPPORT     : out DRAWING_MODE_SUPPORT_TYPE ;
   NUM_OF_SUPPORTED_BITMAP_MODES           : out INTEGER_TYPE ;
   ARRAY_OF_SUPPORTED_BITMAP_MODES         : out BITMAP_MODE_ARRAY_TYPE ;
   SIZE_OF_PIXEL                           : out PIXEL_DIMENSION_TYPE ;
   PREFERRED_BITBLT_PATTERN_SIZE           : out BITBLT_DIMENSION_TYPE ;
   SOURCE_BITMAP_TRUNCATION_CAPABILITY     : out BITMAP_TRUNCATION_TYPE ;
   PREVIOUS_DISPLAY_BITMAP_DATA            : out DISPLAY_BITMAP_DATA_TYPE ) ;
```

Function INQUIRE LIST OF SUPPORTED DRAWING-MODE/TRANSPARENCY PAIRS

```
procedure INQ_SUPPORTED_DRAWING_MODE_TRANSPARENCY_PAIRS(
   NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
   LIST_OF_SUPPORTED_PAIRS : out DRAWING_MODE_TRANSPARENCY_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF SUPPORTED DRAWING-MODE-3/TRANSPARENCY PAIRS

```
procedure INQ_SUPPORTED_DRAWING_MODE_3_TRANSPARENCY_PAIRS(
   NUM_OF_LIST_ELEMENTS_REQUESTED        : in  INTEGER_TYPE ;
   INDEX_OF_FIRST_ELEMENT_TO_RETURN      : in  INTEGER_TYPE ;
   RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
   TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out INTEGER_TYPE ;
   LIST_OF_SUPPORTED_PAIRS : out DRAWING_MODE_TRANSPARENCY_ARRAY_TYPE ) ;
```

Function INQUIRE RASTER STATE

```
procedure INQ_RASTER_STATE(
   RESPONSE_VALIDITY                     : out VALIDITY_FLAG_TYPE ;
   DISPLAY_BITMAP_ID                     : out BITMAP_ID_TYPE ;
   DRAWING_BITMAP_ID                     : out BITMAP_ID_TYPE ;
   DRAWING_MODE                          : out DRAWING_MODE_SPECIFIER_TYPE ;
   MAPPED_FOREGROUND_COLOUR_SELECTION    : out COLOUR_SELECTION_MODE_TYPE ;
   MAPPED_FOREGROUND_COLOUR              : out COLOUR_SPECIFIER_TYPE ;
   MAPPED_BACKGROUND_COLOUR_SELECTION    : out COLOUR_SELECTION_MODE_TYPE ;
   MAPPED_BACKGROUND_COLOUR              : out COLOUR_SPECIFIER_TYPE ;
   TRANSPARENT_COLOUR_SELECTION          : out COLOUR_SELECTION_MODE_TYPE ;
   TRANSPARENT_FOREGROUND_COLOUR         : out COLOUR_SPECIFIER_TYPE ) ;
```

Function INQUIRE LIST OF NON-DISPLAYABLE BITMAP IDENTIFIERS

```
procedure INQ_NON_DISPLAYABLE_BITMAP_IDS(
  NUM_OF_LIST_ELEMENTS_REQUESTED   : in  INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_STATE_LIST     : out INTEGER_TYPE ;
  LIST_OF_BITMAP_IDS               : out BITMAP_ID_ARRAY_TYPE ) ;
```

Function INQUIRE LIST OF DISPLAYABLE BITMAP IDENTIFIERS

```
procedure INQ_DISPLAYABLE_BITMAP_IDS(
  NUM_OF_LIST_ELEMENTS_REQUESTED   : in  INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_STATE_LIST     : out INTEGER_TYPE ;
  LIST_OF_BITMAP_IDS               : out BITMAP_ID_ARRAY_TYPE ) ;
```

Function INQUIRE BITMAP STATE

```
procedure INQ_BITMAP_STATE(
  BITMAP_ID                        : in  BITMAP_ID_TYPE ;
  RESPONSE_VALIDITY                : out VALIDITY_FLAG_TYPE ;
  DEPTH_TYPE                       : out DEPTH_TYPE ;
  DISPLAYABILITY                   : out DISPLAYABILITY_TYPE ;
  BITMAP_MODE                      : out BITMAP_MODE_TYPE ;
  BOTTOM_LEFT_PIXEL                : out DEVICE_POINT_TYPE ;
  TOP_RIGHT_PIXEL                  : out DEVICE_POINT_TYPE ;
  VDC_EXTENT                       : out VDC_EXTENT_TYPE ;
  ISOTROPY                         : out ISOTROPY_FLAG_TYPE ;
  HOR_ALIGNMENT                    : out HOR_ALIGNMENT_FLAG_TYPE ;
  VERT_ALIGNMENT                   : out VERT_ALIGNMENT_FLAG_TYPE ;
  VIEWPORT_SPECIF_MODE             : out VIEWPORT_SPECIF_MODE_TYPE ;
  VIEWPORT_METRIC_SCALE_FACTOR     : out REAL_TYPE ;
  REQUESTED_DEVICE_VIEWPORT        : out DEVICE_VIEWPORT_TYPE ;
  EFFECTIVE_VIEWPORT               : out DEVICE_VIEWPORT_TYPE ;
  SURFACE_CLIP_INDICATOR           : out DRAWING_SURFACE_CLIP_INDICATOR_TYPE ;
  DSCRECT                          : out DEVICE_VIEWPORT_TYPE ;
  DSCRECT_SPECIF_MODE              : out VIEWPORT_SPECIF_MODE_TYPE ;
  DSCRECT_METRIC_SCALE_FACTOR      : out REAL_TYPE ) ;
```

7.6 Binding defined utility functions

7.6.1 Data record utilities

The ISO/IEC 9636, defines the Data Record type, whose purpose is to group a list(s) of CGI parameters for various CGI functions. It is to the client's advantage to utilize data records without having knowledge of their internal structure. For this reason, utilities will be bound by a CGI implementation for the client's use in constructing and interpreting the parameter data contained within a data record. A data record is an ordered set of sub-sequences of parameters of a given type. Each such sub-sequence consists of an identifier for the data type, a count of the number of parameters in the sub-sequence, and the list of parameters itself. The utilities provided here allow the client to inquire the next sub-sequence of a data record, and to insert and extract data record sub-sequences. Each data record sub-sequence must be extracted in its entirety.

7.6.1.1 Data record utility constants

The constants defined here are necessary in order for a client/implementation pair to communicate information in regard to the content of a data record. These constants identify the allowed parameter data values which may be contained within a sub-sequence of a data record and will be returned to a client upon inquiry of the da.

DATA_RECORD_PARAMETER_DATA : constant INDEX_TYPE := 1 ;

Constant - Data Record Parameter Data
Purpose - Specifies a sub-sequence which contains data record parameter data.

COLOUR_INDEX_PARAMETER_DATA : constant INDEX_TYPE := 2 ;

Constant - Colour Index Parameter Data
Purpose - Specifies a sub-sequence which contains colour index parameter data.

COLOUR_DIRECT_PARAMETER_DATA : constant INDEX_TYPE := 3 ;

Constant - Colour Direct Parameter Data
Purpose - Specifies a sub-sequence which contains colour direct parameter data.

CSN_PARAMETER_DATA : constant INDEX_TYPE := 4 ;

Constant - CSN Parameter Data
Purpose - Specifies a sub-sequence which contains client specified name parameter data.

ENUMERATED_PARAMETER_DATA : constant INDEX_TYPE := 5 ;

Constant - Enumerated Parameter Data
Purpose - Specifies a sub-sequence which contains enumerated parameter data.

INTEGER_PARAMETER_DATA : constant INDEX_TYPE := 6 ;

Constant - Integer Parameter Data
Purpose - Specifies a sub-sequence which contains integer parameter data.

INPUT_COLOUR_PARAMETER_DATA : constant INDEX_TYPE := 7 ;

Constant - Input Colour Parameter Data
Purpose - Specifies a sub-sequence which contains input colour parameter data.

FIXED_INTEGER_8_PARAMETER_DATA : constant INDEX_TYPE := 8 ;

Constant - Fixed Integer 8 Parameter Data
Purpose - Specifies a sub-sequence which contains fixed 8-bit integer parameter data.

FIXED_INTEGER_16_PARAMETER_DATA : constant INDEX_TYPE := 9 ;

Constant - Fixed Integer 16 Parameter Data
Purpose - Specifies a sub-sequence which contains fixed 16-bit integer parameter data.

FIXED_INTEGER_32_PARAMETER_DATA : constant INDEX_TYPE := 10 ;

Constant - Fixed Integer 32 Parameter Data
Purpose - Specifies a sub-sequence which contains fixed 32-bit integer parameter data.

INDEX_PARAMETER_DATA : constant INDEX_TYPE := 11 ;

Constant - Index Parameter Data
Purpose - Specifies a sub-sequence which contains index parameter data.

REAL_PARAMETER_DATA : constant INDEX_TYPE := 12 ;

Constant - Real Parameter Data
Purpose - Specifies a sub-sequence which contains real parameter data.

STRING_PARAMETER_DATA : constant INDEX_TYPE := 13 ;

Constant - String Parameter Data
Purpose - Specifies a sub-sequence which contains string parameter data.

FIXED_STRING_PARAMETER_DATA : constant INDEX_TYPE := 14 ;

Constant - Fixed String Parameter Data
Purpose - Specifies a sub-sequence which contains fixed string parameter data.

VIEWPORT_COORD_PARAMETER_DATA : constant INDEX_TYPE := 15 ;

Constant - Viewport Coordinate Parameter Data
Purpose - Specifies a sub-sequence which contains viewport coordinate parameter data.

VDC_PARAMETER_DATA : constant INDEX_TYPE := 16 ;

Constant - Virtual Device Coordinate Parameter Data
Purpose - Specifies a sub-sequence which contains virtual device coordinate parameter data.

7.6.1.2 Data record utility types

The types defined here are those types necessary in excess of any predefined types, to bind the data record utilities. Reference to types not documented in this subclause have been previously defined.

```
type ADD_STATUS_TYPE is ( SUCCESSFUL, DATA_RECORD_OVERFLOW ) ;
```

Type - Add Status Type

Purpose - Specifies the status incurred when adding data to a data record. All procedures which add data to data records will return a parameter of this type. If the data is successfully added to the data record, then a successful status will be returned. If the storage available within the data record is not sufficient to contain amount of data to be added, then the data record overflow status will be returned.

```
type REMOVE_STATUS_TYPE is ( SUCCESSFUL,
                             INVALID_PARAMETER_TYPE,
                             OUTPUT_ARRAY_OVERFLOW,
                             NULL_DATA_RECORD,
                             ERROR ) ;
```

Type - Remove Status Type

Purpose - Specifies the status incurred when extracting data from a data record. All procedures which remove data from data records will return a parameter of this type. If the data is successfully removed from the data record, then a successful status will be returned. If an extraction procedure is called which does not match the next sub-sequence of data in the data record, then the invalid parameter type status will be returned. If the array allocated to return sub-sequence data is not large enough to contain the data in the sub-sequence, then the output array overflow error will be returned. If an attempt is made to extract data from an empty data record, then the null data record status will be returned. The error status is returned if an implementation error not specified above is encountered while performing a data record data extraction. If a data record sub-sequence can not be extracted in its entirety, then the sub-sequence will remain intact. If any status is returned other than successful, the content of the output data array is not deemed to be valid.

7.6.1.3 Data record utility functions

The following procedures and functions define the bindings for the data record utility functions.

Procedure INQ_NEXT_HEADER

```
procedure INQ_NEXT_HEADER(
  DATA_RECORD        : in  DATA_RECORD_TYPE ;
  RESPONSE_VALIDITY  : out VALIDITY_FLAG_TYPE ;
  PARAMETER_COUNT    : out INTEGER_TYPE ;
  PARAMETER_TYPE     : out INDEX_TYPE ) ;
```

Purpose - This procedure will return to the client the contents of the next sub-sequence header in the input data record. The contents of the data record will remain unaffected by this inquiry. The response validity will indicate the success of the inquiry. If the input data record is null, then a response validity of invalid shall be returned.

Function IS_NULL_DATA_RECORD

```
function IS_NULL_DATA_RECORD(
  DATA_RECORD : in  DATA_RECORD_TYPE ) return BOOLEAN ;
```

Purpose - This function will provide an indication as to whether or not a data record contains any data. If the data record is not empty this function will return FALSE, otherwise it will return TRUE.

Procedure INITIALIZE

```
procedure INITIALIZE(
  DATA_RECORD : in out DATA_RECORD_TYPE ) ;
```

Purpose - This procedure will initialize the input data record to its default state. A client may wish to use this procedure to ensure data record integrity prior to adding parameter data. Once a data record has been initialized, its previous content can not be retrieved.

Procedure ADD DATA RECORD

```
procedure ADD_DATA_RECORD(
  PARAMETER_COUNT    : in     INTEGER_TYPE ;
  DATA_RECORD_ARRAY  : in     DATA_RECORD_ARRAY_TYPE ;
  DATA_RECORD        : in out DATA_RECORD_TYPE ;
  STATUS             :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the data record data value(s) contained in the input Data Record Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the

number of data record values to be encoded.

Procedure ADD COLOUR INDEX

```
procedure ADD_COLOUR_INDEX(
   PARAMETER_COUNT      : in     INTEGER_TYPE ;
   COLOUR_INDEX_ARRAY   : in     COLOUR_INDEX_ARRAY_TYPE ;
   DATA_RECORD          : in out DATA_RECORD_TYPE ;
   STATUS               :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the colour index data value(s) contained in the input Colour Index Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of colour index values to be encoded.

Procedure ADD COLOUR DIRECT

```
procedure ADD_COLOUR_DIRECT(
   PARAMETER_COUNT      : in     INTEGER_TYPE ;
   COLOUR_DIRECT_ARRAY  : in     DIRECT_COLOUR_ARRAY_TYPE ;
   DATA_RECORD          : in out DATA_RECORD_TYPE ;
   STATUS               :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the direct colour data value(s) contained in the input Colour Direct Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of direct colour values to be encoded.

Procedure ADD CSN

```
procedure ADD_CSN(
   PARAMETER_COUNT : in     INTEGER_TYPE ;
   CSN_ARRAY       : in     CSN_ARRAY_TYPE ;
   DATA_RECORD     : in out DATA_RECORD_TYPE ;
   STATUS          :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the client specified name (CSN) data value(s) contained in the input CSN Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of client specified name values to be encoded.

Procedure ADD ENUMERATED

```
procedure ADD_ENUMERATED(
   PARAMETER_COUNT    : in     INTEGER_TYPE ;
   ENUMERATED_ARRAY   : in     FIXED_INTEGER_16_ARRAY_TYPE ;
   DATA_RECORD        : in out DATA_RECORD_TYPE ;
   STATUS             :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the enumerated data value(s) contained in the input Enumerated Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of enumerated values to be encoded.

Procedure ADD INTEGER

```
procedure ADD_INTEGER(
  PARAMETER_COUNT : in     INTEGER_TYPE ;
  INTEGER_ARRAY   : in     INTEGER_ARRAY_TYPE ;
  DATA_RECORD     : in out DATA_RECORD_TYPE ;
  STATUS          :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the integer data value(s) contained in the input Integer Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of integer values to be encoded.

Procedure ADD FIXED INTEGER 8

```
procedure ADD_FIXED_INTEGER_8(
  PARAMETER_COUNT : in     INTEGER_TYPE ;
  FIXED_INT_8_ARRAY : in   FIXED_INTEGER_8_ARRAY_TYPE ;
  DATA_RECORD     : in out DATA_RECORD_TYPE ;
  STATUS          :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the fixed 8-bit integer data value(s) contained in the input Fixed Int 8 Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of fixed 8-bit integer values to be encoded.

Procedure ADD FIXED INTEGER 16

```
procedure ADD_FIXED_INTEGER_16(
  PARAMETER_COUNT   : in     INTEGER_TYPE ;
  FIXED_INT_16_ARRAY : in    FIXED_INTEGER_16_ARRAY_TYPE ;
  DATA_RECORD       : in out DATA_RECORD_TYPE ;
  STATUS            :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the fixed 16-bit integer data value(s) contained in the input Fixed Int 16 Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of fixed 16-bit integer values to be encoded.

Procedure ADD FIXED INTEGER 32

```
procedure ADD_FIXED_INTEGER_32(
  PARAMETER_COUNT   : in     INTEGER_TYPE ;
  FIXED_INT_32_ARRAY : in    FIXED_INTEGER_32_ARRAY_TYPE ;
```

```
  DATA_RECORD      : in out DATA_RECORD_TYPE ;
  STATUS           :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the fixed 32-bit integer data value(s) contained in the input Fixed Int 32 Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of fixed 32-bit integer values to be encoded.

Procedure ADD INDEX

```
procedure ADD_INDEX(
  PARAMETER_COUNT : in     INTEGER_TYPE ;
  INDEX_ARRAY     : in     INDEX_ARRAY_TYPE ;
  DATA_RECORD     : in out DATA_RECORD_TYPE ;
  STATUS          :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the index data value(s) contained in the input Index Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of index values to be encoded.

Procedure ADD REAL

```
procedure ADD_REAL(
  PARAMETER_COUNT : in     INTEGER_TYPE ;
  REAL_ARRAY      : in     REAL_ARRAY_TYPE ;
  DATA_RECORD     : in out DATA_RECORD_TYPE ;
  STATUS          :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the real data value(s) contained in the input Real Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of real values to be encoded.

Procedure ADD STRING

```
procedure ADD_STRING(
  PARAMETER_COUNT : in     INTEGER_TYPE ;
  STRING_ARRAY    : in     STRING_ARRAY_TYPE ;
  DATA_RECORD     : in out DATA_RECORD_TYPE ;
  STATUS          :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the string data value(s) contained in the input String Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of string values to be encoded.

Procedure ADD FIXED STRING

```
procedure ADD_FIXED_STRING(
  PARAMETER_COUNT     : in     INTEGER_TYPE ;
  FIXED_STRING_ARRAY  : in     FIXED_STRING_ARRAY_TYPE ;
  DATA_RECORD         : in out DATA_RECORD_TYPE ;
  STATUS              :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the fixed string data value(s) contained in the input Fixed String Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of fixed string values to be encoded.

Procedure ADD VDC

```
procedure ADD_VDC(
  PARAMETER_COUNT : in     INTEGER_TYPE ;
  VDC_ARRAY       : in     VDC_ARRAY_TYPE ;
  DATA_RECORD     : in out DATA_RECORD_TYPE ;
  STATUS          :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the virtual device coordinate (VDC) data value(s) contained in the input VDC Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of virtual device coordinate values to be encoded.

Procedure ADD VIEWPORT COORD

```
procedure ADD_VIEWPORT_COORD(
  PARAMETER_COUNT       : in     INTEGER_TYPE ;
  VIEWPORT_COORD_ARRAY  : in     VIEWPORT_COORD_ARRAY_TYPE ;
  DATA_RECORD           : in out DATA_RECORD_TYPE ;
  STATUS                :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the viewport coordinate data value(s) contained in the input Viewport Coord Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of viewport coordinate values to be encoded.

Procedure ADD INPUT COLOUR

```
procedure ADD_INPUT_COLOUR(
  PARAMETER_COUNT     : in     INTEGER_TYPE ;
  INPUT_COLOUR_ARRAY  : in     INPUT_COLOUR_ARRAY_TYPE ;
  DATA_RECORD         : in out DATA_RECORD_TYPE ;
  STATUS              :    out ADD_STATUS_TYPE ) ;
```

Purpose - This procedure will encode the input colour data value(s) contained in the input Input Colour Array into the next sub-sequence of the input Data Record. The input Parameter Count will specify the number of input colour values to be encoded.

Procedure REMOVE DATA RECORD

```
procedure REMOVE_DATA_RECORD(
   DATA_RECORD          : in out DATA_RECORD_TYPE ;
   PARAMETER_COUNT      :    out INTEGER_TYPE ;
   DATA_RECORD_ARRAY    :    out DATA_RECORD_ARRAY_TYPE ;
   STATUS               :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the data record data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE COLOUR INDEX

```
procedure REMOVE_COLOUR_INDEX(
   DATA_RECORD          : in out DATA_RECORD_TYPE ;
   PARAMETER_COUNT      :    out INTEGER_TYPE ;
   COLOUR_INDEX_ARRAY   :    out COLOUR_INDEX_ARRAY_TYPE ;
   STATUS               :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the colour index data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE COLOUR DIRECT

```
procedure REMOVE_COLOUR_DIRECT(
   DATA_RECORD          : in out DATA_RECORD_TYPE ;
   PARAMETER_COUNT      :    out INTEGER_TYPE ;
   COLOUR_DIRECT_ARRAY  :    out DIRECT_COLOUR_ARRAY_TYPE ;
   STATUS               :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the colour direct data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE CSN

```
procedure REMOVE_CSN(
   DATA_RECORD          : in out DATA_RECORD_TYPE ;
   PARAMETER_COUNT      :    out INTEGER_TYPE ;
   CSN_ARRAY            :    out CSN_ARRAY_TYPE ;
   STATUS               :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the client specified name (CSN) data value(s) contained in the next

sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE ENUMERATED

```
procedure REMOVE_ENUMERATED(
  DATA_RECORD          : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT      :    out INTEGER_TYPE ;
  ENUMERATED_ARRAY     :    out FIXED_INTEGER_16_ARRAY_TYPE ;
  STATUS               :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the enumerated data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE INTEGER

```
procedure REMOVE_INTEGER(
  DATA_RECORD          : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT      :    out INTEGER_TYPE ;
  INTEGER_ARRAY        :    out INTEGER_ARRAY_TYPE ;
  STATUS               :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the integer data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE FIXED INTEGER 8

```
procedure REMOVE_FIXED_INTEGER_8(
  DATA_RECORD          : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT      :    out INTEGER_TYPE ;
  FIXED_INT_8_ARRAY    :    out FIXED_INTEGER_8_ARRAY_TYPE ;
  STATUS               :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the fixed 8-bit integer data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE FIXED INTEGER 16

```
procedure REMOVE_FIXED_INT_16(
  DATA_RECORD          : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT      :    out INTEGER_TYPE ;
  FIXED_INT_16_ARRAY   :    out FIXED_INTEGER_16_ARRAY_TYPE ;
  STATUS               :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the fixed 16-bit integer data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE FIXED INTEGER 32

```
procedure REMOVE_FIXED_INT_32(
  DATA_RECORD        : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT    :    out INTEGER_TYPE ;
  FIXED_INT_32_ARRAY :    out FIXED_INTEGER_32_ARRAY_TYPE ;
  STATUS             :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the fixed 32-bit integer data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE INDEX

```
procedure REMOVE_INDEX(
  DATA_RECORD        : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT    :    out INTEGER_TYPE ;
  INDEX_ARRAY        :    out INDEX_ARRAY_TYPE ;
  STATUS             :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the index data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE REAL

```
procedure REMOVE_REAL(
  DATA_RECORD        : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT    :    out INTEGER_TYPE ;
  REAL_ARRAY         :    out REAL_ARRAY_TYPE ;
  STATUS             :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the real data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE STRING

```
procedure REMOVE_STRING(
  DATA_RECORD        : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT    :    out INTEGER_TYPE ;
  STRING_ARRAY       :    out STRING_ARRAY_TYPE ;
  STATUS             :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the string data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE FIXED STRING

```
procedure REMOVE_FIXED_STRING(
  DATA_RECORD          : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT      :    out INTEGER_TYPE ;
  FIXED_STRING_ARRAY   :    out FIXED_STRING_ARRAY_TYPE ;
  STATUS               :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the fixed string data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE VDC

```
procedure REMOVE_VDC(
  DATA_RECORD          : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT      :    out INTEGER_TYPE ;
  VDC_ARRAY            :    out VDC_ARRAY_TYPE ;
  STATUS               :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the virtual device coordinate (VDC) data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE VIEWPORT COORD

```
procedure REMOVE_VIEWPORT_COORD(
  DATA_RECORD          : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT      :    out INTEGER_TYPE ;
  VIEWPORT_COORD_ARRAY :    out VIEWPORT_COORD_ARRAY_TYPE ;
  STATUS               :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the viewport coordinate data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE INPUT COLOUR

```
procedure REMOVE_INPUT_COLOUR(
  DATA_RECORD          : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT      :    out INTEGER_TYPE ;
  INPUT_COLOUR_ARRAY   :    out INPUT_COLOUR_ARRAY_TYPE ;
```

STATUS : out REMOVE_STATUS_TYPE) ;

Purpose - This procedure will extract the input colour data value(s) contained in the next sub-sequence of the input Data Record. The output Parameter Count will specify the number of valid decoded data values.

Procedure REMOVE EVENT

```
procedure REMOVE_EVENT(
  EVENT_REPORTS_LIST : in out EVENT_REPORTS_LIST_TYPE;
  EVENT              :    out EVENT_TYPE;
  STATUS             :    out REMOVE_STATUS_TYPE ) ;
```

Purpose - This procedure will extract the next event contained in the input Event Reports List. The returned Status will indicate the success of the extraction.

Procedure DEALLOCATE EVENT REPORT LIST

```
procedure DEALLOCATE_EVENT_REPORT_LIST(
  EVENT_REPORT_LIST : in out ACCESS_DATA_RECORD_TYPE ) ;
```

Purpose - This procedure will deallocate any memory allocated to the event report list pointed to by the input parameter.

Procedure DEALLOCATE EVENT

```
procedure DEALLOCATE_EVENT(
  EVENT : in out EVENT_TYPE ) ;
```

Purpose - This procedure will deallocate any memory allocated to the input event.

7.6.2 String utilities

7.6.2.1 String utility functions

The following procedures define the bindings for the string utility functions.

Procedure ADA_TO_CGI_STRING

```
procedure ADA_TO_CGI_STRING(
  ADA_STRING          : in  STRING ;
  RESPONSE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
  CGI_STRING          : out STRING_TYPE ) ;
```

Purpose - This function will convert the input Ada string to its corresponding CGI string type. The returned CGI string may be used as input to CGI functions and procedures. The response validity will indicate the success of the operation. If the response validity is returned as invalid, then the returned cgi string is not deemed to be valid.

Procedure CGI_TO_ADA_STRING

```
procedure CGI_TO_ADA_STRING(
  CGI_STRING          : in  STRING_TYPE ;
  RESPONSE_VALIDITY   : out VALIDITY_FLAG_TYPE ;
  ADA_STRING          : out STRING ) ;
```

Purpose - This function will convert the input CGI string to its equivalent Ada string representation. The response validity will indicate the success of the operation. If the response validity is returned as invalid, then the returned ada string is not deemed to be valid.

7.6.3 Error handling utilities

7.6.3.1 Error handling utility functions

The following procedures define the bindings for the error utility functions.

Procedure INITIALIZE_ERROR_QUEUE

```
procedure INITIALIZE_ERROR_QUEUE ;
```

Purpose - This procedure will initialize the language binding error queue to its default state. All error reports on the queue at the time an initialize command is received will be discarded (lost).

Procedure ADD REPORT

```
procedure ADD_REPORT(
  ERROR_ID              : in ERROR_ID_TYPE ;
  FUNCTION_FOR_ERROR : in FUNCTION_ID_TYPE ) ;
```

Purpose - This procedure will add the input CGI error report to the next available location in the language binding error queue. Note that this queue will operate just as the CGI Interpreter error queue. If the input report causes the error queue to overflow, an ERROR REPORTS LOST error will be generated and a count of lost errors will be maintained.

Procedure DEQUEUE REPORTS

```
procedure DEQUEUE_REPORTS(
  REPORTS_REQUESTED     : in  INTEGER_TYPE ;
  RESPONSE_VALIDITY     : out VALIDITY_FLAG_TYPE ;
  NUM_REPORTS_RETURNED : out INTEGER_TYPE ;
  REPORT_LIST           : out ERROR_QUEUE_TYPE ;
  REPORTS_LEFT          : out INTEGER_TYPE ) ;
```

Purpose - This procedure will dequeue the specified number of CGI language binding error reports from the error queue.

Procedure INQUIRE STATUS

```
procedure INQ_STATUS(
  RESPONSE_VALIDITY       : out VALIDITY_FLAG_TYPE ;
  NUM_OF_QUEUED_REPORTS : out INTEGER_TYPE ;
  REPORTS_LOST            : out YES_NO_FLAG_TYPE ) ;
```

Purpose - This procedure returns the state of the language binding error queue to the client.

Procedure INQUIRE ERROR HANDLING SUPPORT

```
procedure INQ_ERROR_HANDLING_SUPPORT(
  RESPONSE_VALIDITY : out VALIDITY_FLAG_TYPE ;
  SUPPORT_FLAG      : out YES_NO_FLAG_TYPE ) ;
```

Purpose - This procedure returns information to the client about the level of language binding error handling support provided by an implementation. If the implementation does not support these error utility functions, the returned Support Flag will be NO.

7.6.4 Data packing utilities

7.6.4.1 Data packing utility types

The types defined here are those types necessary in excess of any predefined types, to bind the data packing utilities. Reference to types not documented in this subclause have been previously defined.

```
type PACKED_COLOUR_ARRAY_TYPE is private ;
```

Type - Packed Colour Array Type
Purpose - Provides a packed array of colour data for client input to the CGI data packing utility functions.

```
type PACKING_METHOD_TYPE is ( BINARY_RUN_LENGTH,
                              BINARY_PACKED_LIST,
                              CHARACTER_RUN_LENGTH,
                              CHARACTER_BIT_STREAM,
                              IMPLEMENTATION_PRIVATE ) ;
```

Type - Packing Method Type
Purpose - Specifies the various packing methods which may be supported by an implementation. Binary run length and binary packed list are specified in ISO/IEC 9637-1. Character run length and character bit stream are specified in ISO/IEC 9637-2. If supported, the implementation private packing method shall be specified in an implementation's documentation.

```
type PACKING_METHOD_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range 1..5 ) of PACKING_METHOD_TYPE ;
```

Type - Packing Method Array Type
Purpose - Specifies an array of supported packing methods to be returned to a client upon inquiry.

7.6.4.2 Data packing utility functions

The following procedures define the bindings for the data packing utility functions.

Procedure PACK COLOUR ARRAY

```
procedure PACK_COLOUR_ARRAY(
  PACKING_METHOD          : in  PACKING_METHOD_TYPE ;
  X_DIMENSION             : in  INTEGER_TYPE ;
  Y_DIMENSION             : in  INTEGER_TYPE ;
  LOCAL_COLOUR_PREC       : in  INTEGER_TYPE ;
  COLOUR_ARRAY            : in  COLOUR_ARRAY_TYPE ;
  RESPONSE_VALIDITY       : out RESPONSE_VALIDITY_TYPE ;
  PACKED_COLOUR_ARRAY     : out PACKED_COLOUR_ARRAY_TYPE ) ;
```

Purpose - This procedure will return a packed colour array to a client which is comprised of the input colour data and its specifications. The packing method will indicate the desired format for the packed array and the response validity will indicate the success of the operation.

Procedure UNPACK COLOUR ARRAY

```
procedure UNPACK_COLOUR_ARRAY(
  PACKED_COLOUR_ARRAY     : in  PACKED_COLOUR_ARRAY_TYPE ;
  RESPONSE_VALIDITY       : out RESPONSE_VALIDITY_TYPE ;
  PACKING_METHOD          : out PACKING_METHOD_TYPE ;
  X_DIMENSION             : out INTEGER_TYPE ;
  Y_DIMENSION             : out INTEGER_TYPE ;
  LOCAL_COLOUR_PREC       : out INTEGER_TYPE ;
  COLOUR_ARRAY            : out COLOUR_ARRAY_TYPE ) ;
```

Purpose - This procedure will unpack an array of colour data and return its contents and specifications to the client. The returned response validity will indicate the success of the operation.

Procedure PIXEL ARRAY

```
procedure PIXEL_ARRAY(
  ORIGIN_POINT                : in VDC_POINT_TYPE ;
  X_DIMENSION                 : in INTEGER_TYPE ;
  Y_DIMENSION                 : in INTEGER_TYPE ;
  X_SCALE                     : in INTEGER_TYPE ;
  Y_SCALE                     : in INTEGER_TYPE ;
  X_DIRECTION                 : in VDC_DIRECTION_TYPE ;
  Y_DIRECTION                 : in VDC_DIRECTION_TYPE ;
  DRAWING_MODE                : in DRAWING_MODE_SPECIFIER_TYPE ;
  TRANSPARENCY                : in TRANSPARENCY_TYPE ;
```

```
  LOCAL_COLOUR_PREC_REQUIREMENT : in PREC_REQUIREMENT_TYPE ;
  COLOUR_SPECIFIERS             : in PACKED_COLOUR_ARRAY_TYPE ;
  FINAL_FLAG                    : in FINAL_FLAG_TYPE := FINAL ) ;
```

Purpose - This procedure is the packed version of the CGI pixel array function. This procedure shall be implemented identically to the CGI pixel array function with the exception of the colour specifiers data format.

Function GET PIXEL ARRAY

```
procedure GET_PIXEL_ARRAY(
  SOURCE_BITMAP_ID     : in  BITMAP_ID_TYPE ;
  ORIGIN_POINT         : in  VDC_POINT_TYPE ;
  X_DIMENSION          : in  INTEGER_TYPE ;
  Y_DIMENSION          : in  INTEGER_TYPE ;
  X_DIRECTION          : in  VDC_DIRECTION_TYPE ;
  Y_DIRECTION          : in  VDC_DIRECTION_TYPE ;
  LOCAL_COLOUR_PREC    : in  PREC_REQUIREMENT_TYPE ;
  RESPONSE_VALIDITY    : out VALIDITY_FLAG_TYPE ;
  PIXEL_VALIDITY_FLAG  : out PIXEL_VALIDITY_FLAG_TYPE ;
  VALID_X_RANGE        : out VALID_PIXEL_RANGE_TYPE ;
  VALID_Y_RANGE        : out VALID_PIXEL_RANGE_TYPE ;
  COLOUR_SPECIFIERS    : out PACKED_COLOUR_ARRAY_TYPE ) ;
```

Purpose - This procedure is the packed version of the CGI pixel array function. This procedure shall be implemented identically to the CGI get pixel array function with the exception of the colour specifiers data format.

Function CELL ARRAY

```
procedure CELL_ARRAY(
  CELL_ARRAY_PARALLELOGRAM      : in CELL_ARRAY_PARALLELOGRAM_TYPE ;
  DIMENSIONS                    : in CELL_ARRAY_DIMENSION_TYPE ;
  LOCAL_COLOUR_PREC_REQUIREMENT : in INTEGER_TYPE ;
  CELL_COLOUR_SPECIFIERS        : in PACKED_COLOUR_ARRAY_TYPE ;
  FINAL_FLAG                    : in FINAL_FLAG_TYPE := FINAL ) ;
```

Purpose - This procedure is the packed version of the CGI cell array function. This procedure shall be implemented identically to the CGI cell array function with the exception of the cell colour specifiers data format.

Procedure INQ AVAIL PACKING METHODS

```
procedure INQ_AVAIL_PACKING_METHODS(
  RESPONSE_VALIDITY             : out VALIDITY_FLAG_TYPE ;
```

```
    NUM_OF_VALID_PACKING_METHODS     : out INTEGER_TYPE ;
    ARRAY_OF_AVAIL_PACKING_METHODS : out PACKING_METHOD_ARRAY_TYPE ) ;
```

Purpose - This procedure will return the available packing methods supported by an implementation. If no packing methods are supported, the number of available packing methods will be returned as 0.

Annex A
(informative)

Compilable CGI specification

This annex is intended to show one example of a compilable CGI binding to the Ada programming language. It is not the only possible such example. It is expected that implementations of a CGI 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 part of ISO/IEC 9638. The packages defined within this Annex are specified in compilation order. This order is as follows:

CGI Configuration and Types Packages

CGI_CONFIG
CGI_TYPES
CGI_DATA_RECORD_UTILS *
CGI

Constant Packages

CGI_PROFILE_ID_CONST
CGI_FUNCTION_ID_CONST
CGI_REGISTRATION_CONST
CGI_ERROR_CONST

Utility Packages

CGI_STRING_UTILS
CGI_ERROR_HANDLING_UTILS
CGI_PACKING_UTILS

* Although the data record utility package is a client utility package, it needs to be compiled prior to the specification of the CGI. This is due to the implementation dependent definition of the data record type used by the various CGI functions.

A.1 Package specification CGI_CONFIG

```
package CGI_CONFIG is

   -- Package: CGI_CONFIG
   -- Functional Description:
   --
   -- This package provides the client tailorable portion of the CGI.  By providing the definition of these
   -- constants, the client can direct the implementation to constrain various type definitions or to
   -- supply particular client needs.  A description of each constant's purpose is supplied along with any
   -- relevant information about the constant's units or precision.

   REAL_DIGITS : constant := 6 ;

   -- Specifies the number of digits required by the client in order to define a CGI floating point real value.
   -- This value will be used in the declaration of the CGI floating point real type.

   VDC_REAL_DIGITS : constant := 6 ;

   -- Specifies the number of digits required by the client in order to define a CGI VDC floating point real value.
   -- This value will be used in the declaration of the CGI VDC floating point real type.

   MIN_FIXED_POINT_WHOLE_VALUE : constant := -( 2 ** 31 ) ;

   -- Specifies the minimum value for the whole part of a CGI fixed point real.  This value will be used in the
   -- declaration of the CGI fixed point real type.

   MAX_FIXED_POINT_WHOLE_VALUE : constant := ( 2 ** 31 ) - 1 ;

   -- Specifies the maximum value for the whole part of a CGI fixed point real.  This value will be used in the
   -- declaration of the CGI fixed point real type.

   MAX_FIXED_POINT_FRACTION_VALUE : constant := ( 2 ** 31 ) - 1 ;

   -- Specifies the maximum value for the fractional part of a CGI fixed point real.  This value will be used in
   -- the declaration of the CGI fixed point real type.

   VDC_MIN_FIXED_POINT_WHOLE_VALUE : constant := -( 2 ** 31 ) ;

   -- Specifies the minimum value for the whole part of a CGI fixed point VDC real.  This value will be used in
   --  the declaration of the CGI VDC fixed point real type.

   VDC_MAX_FIXED_POINT_WHOLE_VALUE : constant := ( 2 ** 31 ) - 1 ;

   -- Specifies the maximum value for the whole part of a CGI fixed point VDC real.  This value will be used in
   -- the declaration of the CGI VDC fixed point real type.

   VDC_MAX_FIXED_POINT_FRACTION_VALUE : constant := ( 2 ** 31 ) - 1 ;
```

```
-- Specifies the maximum value for the fractional part of a CGI VDC fixed point real.  This value will be
-- used in the declaration of the CGI fixed point real type.

MIN_INTEGER : constant := -( 2 ** 31 ) ;

-- Specifies the minimum value of a CGI signed integer.  The CGI integer range will be used for various CGI
-- basic and abstract data type declarations.  For an implementation which supports varying precisions, this
-- integer range should represent the largest integer precision achievable on the host machine.

MAX_INTEGER : constant := ( 2 ** 31 ) - 1 ;

-- Specifies the maximum value of a CGI signed integer.  The CGI integer range will be used for various CGI
-- basic and abstract data type declarations.  For an implementation which supports varying precisions, this
-- integer range should represent the largest integer precision achievable on the host machine.

VDC_MIN_INTEGER : constant := -( 2 ** 31 ) ;

-- Specifies the minimum value of a CGI signed VDC integer.  The CGI VDC integer range will be used for
-- various CGI basic and abstract data type declarations.  For an implementation which supports varying
-- precisions, this integer range should represent the largest integer precision achievable on the host machine.

VDC_MAX_INTEGER : constant := ( 2 ** 31 ) - 1 ;

-- Specifies the maximum value of a CGI signed VDC integer.  The CGI VDC integer range will be used for
-- various CGI basic and abstract data type declarations.  For an implementation which supports varying
-- precisions, this integer range should represent the largest integer precision achievable on the host machine.

MIN_INDEX : constant := -( 2 ** 31 ) ;

-- Specifies the minimum value of a CGI index.  The CGI index range will be used  to constrain the CGI
-- basic index data type.  For an implementation which supports varying index precision, this range should
-- represent the largest index precision achievable on the host machine.

MAX_INDEX : constant := ( 2 ** 31 ) - 1 ;

-- Specifies the maximum value of a CGI index.  The CGI index range will be used to constrain the CGI
-- basic index data type.  For an implementation which supports varying index precision, this range should
-- represent the largest index precision achievable on the host machine.

MIN_CSN : constant := -( 2 ** 31 ) ;

-- Specifies the minimum value of a CGI client specified name.  The CGI CSN range will be used to constrain
-- the CGI basic client specified name data type.  For an implementation which supports varying CSN
-- precision,  this range should represent the largest CSN precision achievable on the host machine.

MAX_CSN : constant := ( 2 ** 31 ) - 1 ;

-- Specifies the maximum value of a CGI client specified name.  The CGI CSN range will be used to
-- constrain the CGI basic client specified name data type.  For an implementation which supports varying
-- CSN precision, this range should represent the largest CSN precision achievable on the host machine.
```

```
MAX_COLOUR_INDEX : constant := ( 2 ** 31 ) - 1 ;

-- Specifies the maximum value of a CGI colour index. The CGI colour index range will be used to constrain
-- the CGI basic colour index data type. For an implementation which supports varying colour index
-- precision, this range should represent the largest colour index precision achievable on the host machine.

MIN_DIRECT_COLOUR : constant := -( 2 ** 31 ) ;

-- Specifies the minimum value of a CGI direct colour specifier. The CGI direct colour value range will be
-- used to constrain the CGI basic direct colour value data type. For an implementation which supports
-- varying direct colour value precision, this range should represent the largest direct colour value precision
-- achievable on the host machine.

MAX_DIRECT_COLOUR : constant := ( 2 ** 31 ) - 1 ;

-- Specifies the maximum value of a CGI direct colour specifier. The CGI direct colour value range will be
-- used to constrain the CGI basic direct colour value data type. For an implementation which supports
-- varying direct colour value precision, this range should represent the largest direct colour value precision
-- achievable on the host machine.

MAX_INPUT_COLOUR_INDEX : constant := ( 2 ** 31 ) - 1 ;

-- Specifies the maximum value of a CGI input colour index. The CGI input colour index range will be used
-- to constrain the CGI abstract input colour data type. For an implementation which supports raster input,
-- this range should represent the maximum bits per colour in the Raster Class Specific Logical Input Device
-- State List.

MIN_DIRECT_INPUT_COLOUR : constant := -( 2 ** 31 ) ;

-- Specifies the minimum value of a CGI direct input colour value. The CGI direct input colour value range
-- will  be used to constrain the CGI abstract input colour data type. For an implementation which supports
-- raster input, this range should represent the maximum bits per colour in the Raster Class Specific Logical
-- Input Device State List.

MAX_DIRECT_INPUT_COLOUR : constant := ( 2 ** 31 ) - 1 ;

-- Specifies the maximum value of a CGI direct input colour value. The CGI direct input colour value range
-- will be used to constrain the CGI abstract input colour data type. For an implementation which supports
-- raster input, this range should represent the maximum bits per colour in the Raster Class Specific Logical
-- Input Device State List.

MIN_DEVICE_COORD : constant := -( 2 ** 15 ) ;

-- Specifies the minimum value of the CGI abstract device coordinate data type. This value will be used in
-- the declaration of the CGI device coordinate data type.

MAX_DEVICE_COORD : constant := ( 2 ** 15 ) - 1 ;

-- Specifies the maximum value of the CGI abstract device coordinate data type. This value will be used in
-- the declaration of the CGI device coordinate data type.
```

```
MIN_INTRINSIC_NAME : constant := -( 2 ** 15 ) ;

-- Specifies the minimum value of the CGI abstract intrinsic name data type.  This value will be used in the
-- declaration of the CGI intrinsic name data type.

MAX_INTRINSIC_NAME : constant := ( 2 ** 15 ) - 1 ;

-- Specifies the maximum value of the CGI abstract intrinsic name data type.  This value will be used in the
-- declaration of the CGI intrinsic name data type.

MIN_INPUT_SURFACE_COORD : constant := -( 2 ** 15 ) ;

-- Specifies the minimum value of the CGI abstract input surface coordinate data type.  This value will be
--  used in the declaration of the CGI input surface coordinate data type.

MAX_INPUT_SURFACE_COORD : constant := ( 2 ** 15 ) - 1 ;

-- Specifies the maximum value of the CGI abstract input surface coordinate data type.  This value will be
-- used in the declaration of the CGI input surface coordinate data type.

MAX_LOST_ERROR_COUNT : constant := ( 2 ** 15 ) - 1 ;

-- Specifies the maximum value of the Lost Error Count part of the error report data type.  This value will
-- be used in the declaration of the CGI error report data type.

MAX_CHAR_SET_LENGTH : constant := 512 ;

-- Specifies the maximum number of octets within a CGI character set.  This value will be used in the
-- declaration of the CGI character set type.  This may be useful for applications of CGI (client programs)
-- which know the maximum size of CGI character set strings or which want to limit the character set string
-- size due to memory constraints.

MAX_FIXED_STRING_LENGTH : constant := 512 ;

-- Specifies the maximum number of octets within a CGI fixed string.  This value will be used in the
-- declaration of the CGI fixed string type.  This may be useful for applications of CGI (client programs)
-- which know the maximum size of CGI fixed strings or which want to limit the string size due to memory
-- constraints.

MAX_STRING_LENGTH : constant := 1024 ;

-- Specifies the maximum number of octets within a CGI string.  This value will be used in the declaration
-- of the CGI string type.  This may be useful for applications of CGI (client programs) which know the
-- maximum size of CGI strings or which want to limit the string size due to memory constraints.

CGI_ERROR_NOTIFICATION : BOOLEAN := FALSE ;

-- Specifies whether or not an application wishes to be notified by the implementation when errors occur.
-- Normally CGI errors will be stored in the CGI error queue until the client performs a DEQUEUE ERROR
-- REPORTS.  This boolean allows the client to obtain knowledge of errors (only the errors detected by the
```

```
    -- CGI generator/language binding) as they are detected.  When an error is detected by the CGI generator and
    -- the error handling state is set to REPORTING ON for the associated error class, the implementation will
    -- raise the CGI ERROR defined in the CGI Exceptions clause of this standard.
    --
    --  Possible Values:
    --
    --    TRUE  - If this constant is set to true, any run-time errors which arise in the CGI implementation
    --            will raise the CGI_ERROR exception defined in the CGI package specification.  The
    --            exception will be raised after the error has been added to the generator error queue.
    --
    --    FALSE - If this constant is set to false, any run-time errors which arise in the CGI implementation
    --            will not interrupt the client's session.  The error will be added to the generator error queue
    --            and the appropriate error reaction will be taken as dictated by the error class.

    SIZE_OF_GENERATOR_ERROR_QUEUE : constant := 64 ;

    -- Specifies the maximum number of error reports which may be held in the CGI generator/language binding
    -- error queue before error reports are lost.  This value will be used in the declaration of the CGI generator
    -- error queue type.

    SIZE_OF_INTERPRETER_ERROR_QUEUE : constant := 64 ;

    -- Specifies the maximum number of error reports which may be held in the CGI Interpreter error queue
    -- before error reports are lost.  This value will be used in the declaration of the CGI error queue type.

    LAST_ARRAY_INDEX : constant := ( 2 ** 15 ) - 1 ;

    -- Specifies the last index in the array index type.  The array index type will be used to declare a common
    -- array index range for the various CGI array definitions.

    FIRST_ENTRY_IN_COLOUR_TABLE : constant := 1 ;

    -- Specifies the first entry of the colour table on the graphics device.

    LAST_ENTRY_IN_COLOUR_TABLE : constant := 64 ;

    -- Specifies the last entry of the colour table on the graphics device.


                    ------------------  Application Defined Constants Section  ----------------------


    MAX_DATA_RECORD_SIZE : constant := 1024 ;

    -- Specifies the maximum number of octets within a CGI data record.  This implementation of CGI defines
    -- the CGI data record in terms of the binary encoding protocol definition.  This value will be used in the
    -- declaration of the CGI data record type in order to constrain the maximum data record size.  This size is
    -- specified in bytes.

  end CGI_CONFIG ;
```

A.2 Package specification CGI_TYPES

```
with CGI_CONFIG ;

package CGI_TYPES is

   -- Package: CGI_TYPES
   -- Functional Description:
   --
   -- This package provides all of the CGI type definitions as specified in the ISO/IEC 9636.

   -- Define the types which are basic to the CGI and will be used throughout.

   -- Define the integer and real types for normal and VDC operations.

   type INTEGER_TYPE is range
      CGI_CONFIG.MIN_INTEGER..CGI_CONFIG.MAX_INTEGER ;

   type VDC_INTEGER_TYPE is range
      CGI_CONFIG.VDC_MIN_INTEGER..CGI_CONFIG.VDC_MAX_INTEGER ;

   type FLOATING_POINT_REAL_TYPE is digits CGI_CONFIG.REAL_DIGITS ;

   type VDC_FLOATING_POINT_REAL_TYPE is digits CGI_CONFIG.VDC_REAL_DIGITS ;

   type FIXED_POINT_WHOLE_PART_TYPE is range
      CGI_CONFIG.MIN_FIXED_POINT_WHOLE_VALUE..
      CGI_CONFIG.MAX_FIXED_POINT_WHOLE_VALUE ;

   type FIXED_POINT_FRACTION_PART_TYPE is range
      0..CGI_CONFIG.MAX_FIXED_POINT_FRACTION_VALUE ;

   type FIXED_POINT_REAL_TYPE is
      record
         WHOLE_PART    : FIXED_POINT_WHOLE_PART_TYPE ;
         FRACTION_PART : FIXED_POINT_FRACTION_PART_TYPE ;
      end record ;

   type VDC_FIXED_POINT_WHOLE_PART_TYPE is range
      CGI_CONFIG.VDC_MIN_FIXED_POINT_WHOLE_VALUE..
      CGI_CONFIG.VDC_MAX_FIXED_POINT_WHOLE_VALUE ;

   type VDC_FIXED_POINT_FRACTION_PART_TYPE is range
      0..CGI_CONFIG.VDC_MAX_FIXED_POINT_FRACTION_VALUE ;

   type VDC_FIXED_POINT_REAL_TYPE is
      record
         WHOLE_PART    : VDC_FIXED_POINT_WHOLE_PART_TYPE ;
```

```
      FRACTION_PART : VDC_FIXED_POINT_FRACTION_PART_TYPE ;
    end record ;

  -- Define pairs of integers and reals for real and point data.


  type INTEGER_PAIR_TYPE is
    record
      X : INTEGER_TYPE ;
      Y : INTEGER_TYPE ;
    end record ;

  type FLOAT_PAIR_TYPE is
    record
      X : FLOATING_POINT_REAL_TYPE ;
      Y : FLOATING_POINT_REAL_TYPE ;
    end record ;

  type FIXED_PAIR_TYPE is
    record
      X : FIXED_POINT_REAL_TYPE ;
      Y : FIXED_POINT_REAL_TYPE ;
    end record ;

  type VDC_INTEGER_PAIR_TYPE is
    record
      X : VDC_INTEGER_TYPE ;
      Y : VDC_INTEGER_TYPE ;
    end record ;

  type VDC_FLOAT_PAIR_TYPE is
    record
      X : VDC_FLOATING_POINT_REAL_TYPE ;
      Y : VDC_FLOATING_POINT_REAL_TYPE ;
    end record ;

  type VDC_FIXED_PAIR_TYPE is
    record
      X : VDC_FIXED_POINT_REAL_TYPE ;
      Y : VDC_FIXED_POINT_REAL_TYPE ;
    end record ;

  -- Define the array index range and the Yes/No flag since these will also used throughout the CGI.

  type ARRAY_INDEX_RANGE_TYPE is range 0..CGI_CONFIG.LAST_ARRAY_INDEX ;

  subtype ARRAY_INDEX_TYPE is ARRAY_INDEX_RANGE_TYPE
    range 1..ARRAY_INDEX_RANGE_TYPE'LAST ;

  type YES_NO_FLAG_TYPE is ( NO, YES ) ;
```

```
---------- CGI Basic Data Type Definitions ----------

type REAL_MODE_TYPE is ( FIXED_POINT, FLOATING_POINT ) ;

type VDC_MODE_TYPE is
  ( INTEGER_VDC, FIXED_POINT_VDC, FLOATING_POINT_VDC ) ;

type COLOUR_INDEX_TYPE is range 0..CGI_CONFIG.MAX_COLOUR_INDEX ;


type CSN_TYPE is range CGI_CONFIG.MIN_CSN..CGI_CONFIG.MAX_CSN ;

type DIRECT_COLOUR_VALUE_RANGE_TYPE is range
  CGI_CONFIG.MIN_DIRECT_COLOUR..CGI_CONFIG.MAX_DIRECT_COLOUR ;

type DIRECT_COLOUR_VALUE_TYPE is
  record
    RED   : DIRECT_COLOUR_VALUE_RANGE_TYPE ;
    GREEN : DIRECT_COLOUR_VALUE_RANGE_TYPE ;
    BLUE  : DIRECT_COLOUR_VALUE_RANGE_TYPE ;
  end record ;

type DIRECT_INPUT_COLOUR_VALUE_RANGE_TYPE is range
  CGI_CONFIG.MIN_DIRECT_INPUT_COLOUR..CGI_CONFIG.MAX_DIRECT_INPUT_COLOUR ;

type DIRECT_INPUT_COLOUR_VALUE_TYPE is
  record
    RED   : DIRECT_INPUT_COLOUR_VALUE_RANGE_TYPE ;
    GREEN : DIRECT_INPUT_COLOUR_VALUE_RANGE_TYPE ;
    BLUE  : DIRECT_INPUT_COLOUR_VALUE_RANGE_TYPE ;
 end record ;

type FIXED_INTEGER_8_TYPE is range -( 2 ** 7 )..( 2 ** 7 ) - 1 ;

type FIXED_INTEGER_16_TYPE is range -( 2 ** 15 )..( 2 ** 15 ) - 1 ;

type FIXED_INTEGER_32_TYPE is range -( 2 ** 31 )..( 2 ** 31 ) - 1 ;

type INDEX_TYPE is range CGI_CONFIG.MIN_INDEX..CGI_CONFIG.MAX_INDEX ;

type INPUT_COLOUR_INDEX_TYPE is range 0..CGI_CONFIG.MAX_INPUT_COLOUR_INDEX ;

type INPUT_COLOUR_TYPE( COLOUR : YES_NO_FLAG_TYPE := NO ) is
  record
    case COLOUR is
      when NO  => INDEXED_REALIZATION : INPUT_COLOUR_INDEX_TYPE ;
      when YES => DIRECT_REALIZATION : DIRECT_INPUT_COLOUR_VALUE_TYPE ;
    end case ;
  end record ;
```

```
type REAL_TYPE( REAL_MODE : REAL_MODE_TYPE := FLOATING_POINT ) is
  record
    case REAL_MODE is
      when FLOATING_POINT => FLOAT_DATA : FLOATING_POINT_REAL_TYPE ;
      when FIXED_POINT    => FIXED_DATA  : FIXED_POINT_REAL_TYPE ;
    end case ;
  end record ;

type VDC_TYPE( VDC_TYPE : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    case VDC_TYPE is
      when INTEGER_VDC          => INTEGER_VDC : VDC_INTEGER_TYPE ;
      when FIXED_POINT_VDC      => FIXED_VDC   : VDC_FIXED_POINT_REAL_TYPE ;
      when FLOATING_POINT_VDC => FLOAT_VDC   : VDC_FLOATING_POINT_REAL_TYPE ;
    end case ;
  end record ;

                    ------------------- String Type Definition --------------------

type OCTET_TYPE is range 0..( 2 ** 8 ) - 1 ;
for OCTET_TYPE'SIZE use 8 ;

type OCTET_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of OCTET_TYPE ;
pragma pack( OCTET_ARRAY_TYPE ) ;

subtype STRING_LENGTH_TYPE is ARRAY_INDEX_RANGE_TYPE
  range 0..CGI_CONFIG.MAX_STRING_LENGTH ;

subtype CHAR_SET_LENGTH_TYPE is ARRAY_INDEX_RANGE_TYPE
  range 0..CGI_CONFIG.MAX_CHAR_SET_LENGTH ;

subtype FIXED_STRING_LENGTH_TYPE is ARRAY_INDEX_RANGE_TYPE
  range 0..CGI_CONFIG.MAX_FIXED_STRING_LENGTH ;

type STRING_TYPE
  ( LENGTH : STRING_LENGTH_TYPE := CGI_CONFIG.MAX_STRING_LENGTH ) is
  record
    CHARS_IN_STRING : STRING_LENGTH_TYPE := 0 ;
    STRING_DATA     : OCTET_ARRAY_TYPE( 1..LENGTH ) ;
  end record ;

type CHAR_SET_STRING_TYPE
  ( LENGTH : CHAR_SET_LENGTH_TYPE := CGI_CONFIG.MAX_CHAR_SET_LENGTH ) is
  record
    CHARS_IN_STRING : CHAR_SET_LENGTH_TYPE := 0 ;
    STRING_DATA     : OCTET_ARRAY_TYPE( 1..LENGTH ) ;
  end record ;

type FIXED_STRING_TYPE
  ( LENGTH : FIXED_STRING_LENGTH_TYPE := CGI_CONFIG.MAX_FIXED_STRING_LENGTH ) is
```

```
    record
      CHARS_IN_STRING : FIXED_STRING_LENGTH_TYPE := 0 ;
      STRING_DATA     : OCTET_ARRAY_TYPE( 1..LENGTH ) ;
    end record ;


  ---------- CGI Derived Types Package ----------

  -- This internal package has been provided for the implementation of the CGI derived abstract data
  -- type definitions.  Since Ada does not allow the type derivation from a type definition within the
  -- same package, derived types will be derived from this internal package.
  --
  -- ** NOTE **
  --      This package is not intended for the client's usage.

  package DERIVED_TYPES is

    -- The only CGI basic data types which will be used for derivation are the Client Specified
    -- Name and Intrinsic Name types.

    type CSN_TYPE is range CGI_CONFIG.MIN_CSN..CGI_CONFIG.MAX_CSN ;

    type INTRINSIC_NAME_TYPE is
       range CGI_CONFIG.MIN_INTRINSIC_NAME..CGI_CONFIG.MAX_INTRINSIC_NAME ;

  end DERIVED_TYPES ;

                  ---------- CGI Abstract Data Type Definitions ----------

  type COLOUR_SELECTION_MODE_TYPE is ( INDEXED, DIRECT ) ;

  type SPECIF_MODE_TYPE is ( VDC, SCALED ) ;

  type VIEWPORT_MODE_TYPE is
    ( INTEGER_VIEWPORT, FIXED_POINT_VIEWPORT, FLOATING_POINT_VIEWPORT ) ;

  type ASN_TYPE is new DERIVED_TYPES.CSN_TYPE ;

  type BITMAP_ID_TYPE is new DERIVED_TYPES.CSN_TYPE ;

  type CHAR_SET_ID_TYPE is ( CHAR_SET_94,
                             CHAR_MULTIBYTE_SET_94,
                             CHAR_SET_96,
                             CHAR_MULTIBYTE_SET_96,
                             COMPLETE_CODE ) ;

  type CHAR_SET_TYPE
    ( LENGTH : CHAR_SET_LENGTH_TYPE := CGI_CONFIG.MAX_CHAR_SET_LENGTH ) is
    record
      CHAR_SET_ID     : CHAR_SET_ID_TYPE ;
```

```
      CHAR_SET_STRING : CHAR_SET_STRING_TYPE( LENGTH ) ;
    end record ;

  type COLOUR_SPECIFIER_TYPE
    ( COLOUR_SELECTION_MODE : COLOUR_SELECTION_MODE_TYPE := INDEXED ) is
    record
      case COLOUR_SELECTION_MODE is
        when DIRECT   => DIRECT_COLOUR : DIRECT_COLOUR_VALUE_TYPE ;
        when INDEXED => INDEX_COLOUR  : COLOUR_INDEX_TYPE ;
      end case ;
    end record ;

  type DEVICE_COORD_TYPE is range
    CGI_CONFIG.MIN_DEVICE_COORD..CGI_CONFIG.MAX_DEVICE_COORD ;

  type DEVICE_POINT_TYPE is
    record
      X : DEVICE_COORD_TYPE ;
      Y : DEVICE_COORD_TYPE ;
    end record ;

  type INTRINSIC_NAME_TYPE is range
    CGI_CONFIG.MIN_INTRINSIC_NAME..CGI_CONFIG.MAX_INTRINSIC_NAME ;

  subtype ERROR_CLASS_TYPE is INTRINSIC_NAME_TYPE range 1..9 ;

  type FUNCTION_ID_TYPE is new DERIVED_TYPES.INTRINSIC_NAME_TYPE ;

  type ERROR_ID_TYPE is
    record
      ERROR_CLASS : ERROR_CLASS_TYPE ;
      ERROR_NUM   : INTRINSIC_NAME_TYPE ;
    end record ;

  type LOST_ERROR_COUNT_TYPE is range 2..CGI_CONFIG.MAX_LOST_ERROR_COUNT ;

  type ERROR_REPORT_TYPE( REPORTS_LOST : BOOLEAN := FALSE ) is
    record
      ERROR_ID : ERROR_ID_TYPE ;
      case REPORTS_LOST is
        when TRUE  => LOST_ERROR_COUNT    : LOST_ERROR_COUNT_TYPE ;
        when FALSE => FUNCTION_FOR_ERROR : FUNCTION_ID_TYPE ;
      end case ;
    end record ;

  type INPUT_SURFACE_COORD_TYPE is range
    CGI_CONFIG.MIN_INPUT_SURFACE_COORD..CGI_CONFIG.MAX_INPUT_SURFACE_COORD ;

  type INPUT_SURFACE_POINT_TYPE is
    record
```

```
      X : INPUT_SURFACE_COORD_TYPE ;
      Y : INPUT_SURFACE_COORD_TYPE ;
    end record ;

  type PICK_ID_TYPE is new DERIVED_TYPES.CSN_TYPE ;

  type SEGMENT_ID_TYPE is new DERIVED_TYPES.CSN_TYPE ;

  type PICK_VALUE_TYPE is
    record
      SEGMENT_ID : SEGMENT_ID_TYPE ;
      PICK_ID    : PICK_ID_TYPE ;
    end record ;

  type PROFILE_ID_TYPE is new DERIVED_TYPES.INTRINSIC_NAME_TYPE;

  type SIZE_SPECIF_TYPE( SPECIF_MODE : SPECIF_MODE_TYPE := VDC ) is
    record
      case SPECIF_MODE is
        when VDC    => VDC_SIZE    : VDC_TYPE ;
        when SCALED => SCALED_SIZE : REAL_TYPE ;
      end case ;
    end record ;

  type VDC_POINT_TYPE( VDC_TYPE : VDC_MODE_TYPE := INTEGER_VDC ) is
    record
      case VDC_TYPE is
        when INTEGER_VDC          => INTEGER_DATA : VDC_INTEGER_PAIR_TYPE ;
        when FIXED_POINT_VDC      => FIXED_DATA   : VDC_FIXED_PAIR_TYPE ;
        when FLOATING_POINT_VDC => FLOAT_DATA     : VDC_FLOAT_PAIR_TYPE ;
      end case ;
    end record ;

  type VIEWPORT_COORD_TYPE
    ( VIEWPORT_TYPE : VIEWPORT_MODE_TYPE := INTEGER_VIEWPORT ) is
    record
      case VIEWPORT_TYPE is
        when INTEGER_VIEWPORT =>
          INTEGER_VIEWPORT : INTEGER_TYPE ;
        when FIXED_POINT_VIEWPORT =>
          FIXED_VIEWPORT : FIXED_POINT_REAL_TYPE ;
        when FLOATING_POINT_VIEWPORT =>
          FLOAT_VIEWPORT : FLOATING_POINT_REAL_TYPE;
      end case ;
    end record ;

  type VIEWPORT_POINT_TYPE
    ( VIEWPORT_TYPE : VIEWPORT_MODE_TYPE := INTEGER_VIEWPORT ) is
    record
      case VIEWPORT_TYPE is
```

```
        when INTEGER_VIEWPORT          => INTEGER_DATA : INTEGER_PAIR_TYPE ;
        when FIXED_POINT_VIEWPORT      => FIXED_DATA   : FIXED_PAIR_TYPE ;
        when FLOATING_POINT_VIEWPORT => FLOAT_DATA     : FLOAT_PAIR_TYPE ;
      end case ;
    end record ;

                    ---------- CGI Enumerated Type Definitions ----------

type ACTION_REQUIRED_FLAG_TYPE is ( NO_ACTION_REQUIRED, ACTION_REQUIRED ) ;

type ASF_TYPE is ( LINE_TYPE_ASF,
                   LINE_WIDTH_ASF,
                   LINE_COLOUR_ASF,
                   MARKER_TYPE_ASF,
                   MARKER_SIZE_ASF,
                   MARKER_COLOUR_ASF,
                   TEXT_FONT_INDEX_ASF,
                   TEXT_PREC_ASF,
                   CHAR_EXPAN_FACTOR_ASF,
                   CHAR_SPACING_ASF,
                   TEXT_COLOUR_ASF,
                   INTERIOR_STYLE_ASF,
                   FILL_COLOUR_ASF,
                   HATCH_INDEX_ASF,
                   PATTERN_INDEX_ASF,
                   EDGE_TYPE_ASF,
                   EDGE_WIDTH_ASF,
                   EDGE_COLOUR_ASF,
                   EDGE_VISIBILITY_ASF,
                   FILL_BITMAP_ASF ) ;

type ASF_VALUE_TYPE is ( INDIVIDUAL, BUNDLED ) ;

type BACKGROUND_COLOUR_CAP_TYPE is ( NONE, INDEX_0, INDEXED, FULL ) ;

type BLOCK_CONTROL_TYPE is ( NOT_BLOCKED, BLOCKED ) ;

type BITMAP_MODE_TYPE is ( INDEXED, DIRECT, MIXED ) ;

type BITMAP_TRUNCATION_TYPE is ( NOT_TRUNCATED, TRUNCATED ) ;

type BUNDLE_TABLE_TYPE is ( LINE, MARKER, TEXT, FILL, EDGE ) ;

type CELL_ARRAY_ALIGNMENT_CAP_TYPE is ( AXIS_ALIGNED, SKEWED ) ;

type CELL_ARRAY_FILL_CAP_TYPE is ( OUTLINED, FILLED ) ;

type CELL_ARRAY_TECHNIQUE_TYPE is ( OTHER, EXACT ) ;

type CLEAR_DRAWING_SURFACE_TYPE is ( UNCONDITIONAL, CONDITIONAL ) ;
```

```
type CLIP_EFFECTIVENESS_TYPE is ( NONE, CIRCUMSCRIBED, FULL ) ;

type CLIP_INHERITANCE_FILTER_TYPE is ( STATE_LIST, INTERSECTION ) ;

type CLIP_MODE_TYPE is ( LOCUS, SHAPE, LOCUS_THEN_SHAPE ) ;

type CLOSE_TYPE is ( PIE, CHORD ) ;

type CODING_TECHNIQUE_TYPE is ( BASIC_7_BIT,
                                BASIC_8_BIT,
                                EXTENDED_7_BIT,
                                EXTENDED_8_BIT,
                                PRIVATE_CODING ) ;

type COLOUR_REALIZATION_TYPE is ( ADDITIVE, SUBTRACTIVE ) ;

type COLOUR_SELECTION_MODE_AVAIL_TYPE is ( INDEXED_ONLY,
                                           INDEXED_AND_DIRECT ) ;

type COMPOUND_OBJECT_CAP_TYPE is ( NONE, GLOBAL, LOCAL ) ;

type DEFERRAL_MODE_TYPE is ( ASTI, BNI, ASAP ) ;

type DEPTH_TYPE is ( MAPPED, FULL_DEPTH ) ;

type DEPTHS_SUPPORTED_TYPE is ( FULL_DEPTH, MAPPED_AND_FULL_DEPTH ) ;

type DETECTABILITY_TYPE is ( UNDETECTABLE, DETECTABLE ) ;

type DEVICE_CLASS_TYPE is ( OUTPUT, INPUT, OUTIN ) ;

type DISABLED_ENABLED_FLAG_TYPE is ( DISABLED, ENABLED ) ;

type DISPLAYABILITY_TYPE is ( NON_DISPLAYABLE, DISPLAYABLE ) ;

type DISPLAY_BITMAP_DATA_TYPE is ( CLEARED, PRESERVED ) ;

type DISPLAY_TYPE is ( VECTOR, RASTER, OTHER ) ;

type DRAWING_MODE_SUPPORT_TYPE is ( NONE, SOME, ALL_MODES ) ;

type DRAWING_SURFACE_CLIP_INDICATOR_TYPE is ( DSCRECT, VIEWPORT, OFF ) ;

type DRAWING_SURFACE_STATE_TYPE is ( DIRTY, CLEAN ) ;

type DYN_MOD_FLAG_TYPE is ( IRG, CBS, IMM ) ;

type ECHO_ENTITY_ECHO_CONTROL_TYPE is ( ECHO_OFF, ECHO_ON ) ;

type ECHO_ENTITY_PROMPT_CONTROL_TYPE is ( PROMPT_OFF, PROMPT_ON ) ;
```

```
type ECHO_ENTITY_STATE_TYPE is ( READY, ACTIVE ) ;

type EDGE_OUT_FLAG_TYPE is ( INVISIBLE,
                             VISIBLE,
                             CLOSE_INVISIBLE,
                             CLOSE_VISIBLE ) ;

type EDGE_VISIBILITY_TYPE is ( VISIBLE, INVISIBLE ) ;

type EDGE_WIDTH_REALIZATION_TYPE is ( INTERIOR, CENTRED ) ;

type EFFECTIVE_PRIORITY_TYPE is ( OTHER,
                                  TIME_ORDER_REOPEN,
                                  TIME_ORDER ) ;

type ERROR_HANDLING_FLAG_TYPE is ( ON, REPORTING_OFF, DETECTION_OFF ) ;

type EVENT_QUEUE_STATE_TYPE is ( RELEASED,
                                 EMPTY_NO_LID_ENABLED,
                                 NOT_EMPTY_NO_LID_ENABLED,
                                 EMPTY_LID_ENABLED,
                                 NOT_EMPTY_LID_ENABLED ) ;

type EVENT_STATUS_TYPE is ( TIMEOUT, EVENT, OVERFLOW, BREAK ) ;

type FILTER_SELECTION_TYPE is ( LINE_TYPE_ASF,
                                LINE_WIDTH_ASF,
                                LINE_COLOUR_ASF,
                                MARKER_TYPE_ASF,
                                MARKER_SIZE_ASF,
                                MARKER_COLOUR_ASF,
                                TEXT_FONT_INDEX_ASF,
                                TEXT_PREC_ASF,
                                CHAR_EXPANSION_FACTOR_ASF,
                                CHAR_SPACING_ASF,
                                TEXT_COLOUR_ASF,
                                INTERIOR_STYLE_ASF,
                                FILL_COLOUR_ASF,
                                HATCH_INDEX_ASF,
                                PATTERN_INDEX_ASF,
                                FILL_BITMAP_ASF,
                                EDGE_TYPE_ASF,
                                EDGE_WIDTH_ASF,
                                EDGE_COLOUR_ASF,
                                EDGE_VISIBILITY_ASF,
                                LINE_BUNDLE_INDEX,
                                LINE_TYPE,
                                LINE_WIDTH,
                                LINE_COLOUR,
                                LINE_CLIPPING_MODE,
```

```
MARKER_BUNDLE_INDEX,
MARKER_TYPE,
MARKER_SIZE,
MARKER_COLOUR,
MARKER_CLIPPING_MODE,
TEXT_BUNDLE_INDEX,
TEXT_FONT_INDEX,
TEXT_COLOUR,
CHAR_EXPANSION_FACTOR,
CHAR_SPACING,
CHAR_HEIGHT,
CHAR_ORIENTATION,
TEXT_PREC,
TEXT_PATH,
TEXT_ALIGNMENT,
FILL_BUNDLE_INDEX,
INTERIOR_STYLE,
FILL_COLOUR,
HATCH_INDEX,
PATTERN_INDEX,
FILL_BITMAP,
EDGE_BUNDLE_INDEX,
EDGE_TYPE,
EDGE_WIDTH,
EDGE_COLOUR,
EDGE_VISIBILITY,
EDGE_CLIPPING_MODE,
FILL_REFERENCE_POINT,
PATTERN_SIZE,
AUXILIARY_COLOUR,
TRANSPARENCY,
DRAWING_MODE,
PICK_ID,
LINE_ASFS,
MARKER_ASFS,
TEXT_ASFS,
FILL_ASFS,
EDGE_ASFS,
ALL_ASFS,
LINE_ATTRIBUTES,
MARKER_ATTRIBUTES,
LOCAL_TEXT_ATTRIBUTES,
GLOBAL_TEXT_ATTRIBUTES,
FILL_ATTRIBUTES,
EDGE_ATTRIBUTES,
PATTERN_ATTRIBUTES,
OUTPUT_CONTROL,
PICK_ATTRIBUTES,
ALL_ATTRIBUTES,
ALL_ATTRIBUTES_AND_ASFS ) ;
```

```
type FINAL_FLAG_TYPE is ( NOT_FINAL, FINAL ) ;

type HARD_SOFT_COPY_TYPE is ( HARD, SOFT ) ;

type HIGHLIGHTING_TYPE is ( NORMAL, HIGHLIGHTED ) ;

type HOR_ALIGNMENT_FLAG_TYPE is ( LEFT, CENTRE, RIGHT ) ;

type HOR_ALIGNMENT_TYPE is ( NORMAL_HOR,
                             LEFT,
                             CENTRE,
                             RIGHT,
                             CONTINUOUS_HOR ) ;

type IMPLICIT_SEGMENT_REGEN_MODE_TYPE is ( SUPPRESSED, UQUM, ALLOWED ) ;

type INHERITANCE_FILTER_TYPE is ( STATE_LIST, SEGMENT ) ;

type INPUT_CLASS_TYPE is ( LOCATOR_LID,
                           STROKE_LID,
                           VALUATOR_LID,
                           CHOICE_LID,
                           PICK_LID,
                           STRING_LID,
                           RASTER_LID,
                           GENERAL_LID ) ;

type INPUT_DEVICE_STATE_TYPE is ( RELEASED,
                                  READY,
                                  REQUEST_PENDING,
                                  ECHO_REQUEST_ENABLED,
                                  ECHO_REQUEST_PENDING,
                                  ECHO_REQUEST_COMPLETED,
                                  EVENTS_ENABLED ) ;

type INQUIRY_SOURCE_FLAG_TYPE is ( CURRENT, DEFAULT ) ;

type INTERIOR_STYLE_TYPE is ( HOLLOW,
                              SOLID,
                              PATTERN,
                              HATCH,
                              EMPTY,
                              BITMAP ) ;

type ISOTROPY_FLAG_TYPE is ( NOT_FORCED, FORCED ) ;

type LINE_EDGE_CONTINUITY_CAP_TYPE is ( RESTART, CONTINUOUS, OTHER ) ;

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

```
type OUTPUT_STATE_TYPE is ( ACTIVE, TEXT_OPEN, FIGURE_OPEN ) ;

type PATTERN_FILL_FALLBACK_TYPE is ( CONDENSE, TRUNCATE ) ;

type PATTERN_TRAN_SUPPORT_TYPE is ( NONE, UNSKEWED, FULL ) ;

type PICK_APERTURE_SHAPE_TYPE is ( RECTANGLE, CIRCLE, ELLIPSE ) ;

type PIXEL_RELATIVE_LOCATION_TYPE is ( ON, BETWEEN ) ;

type PIXEL_VALIDITY_FLAG_TYPE is ( NONE, SOME, ALL_VALID ) ;

type PROFILE_SUPPORT_FLAG_TYPE is ( NO, YES, UNRECOGNIZED ) ;

type REQUEST_STATUS_TYPE is ( TRIGGER_FIRED, BREAK, TIMEOUT ) ;

type SEGMENT_OPEN_STATE_TYPE is ( NO, YES, OVERFLOW ) ;

type SPOT_CENTRE_INTERPRETATION_TYPE is ( NOMINAL, ACTUAL ) ;

type SURFACE_INTERPRETATION_TYPE is ( NOMINAL, ACTUAL, UNLIMITED ) ;

type TEXT_PATH_TYPE is ( RIGHT, LEFT, UP, DOWN ) ;

type TEXT_PREC_TYPE is ( STRING_PREC, CHAR_PREC, STROKE_PREC ) ;

type TIMEOUT_CAP_TYPE is ( LIMITED_CAP, FULL_CAP ) ;

type TIMESTAMP_SUPPORT_TYPE is ( NA, TRIGGER_COUNT, CLOCK ) ;

type TRANSPARENCY_TYPE is ( OPAQUE, TRANSPARENT ) ;

type UNREPORTED_BREAK_STATE_TYPE is ( NO_BREAK, BREAK ) ;

type UNREPORTED_OVERFLOW_STATE_TYPE is ( NO_OVERFLOW, OVERFLOW ) ;

type VALIDITY_FLAG_TYPE is ( INVALID, VALID ) ;

type VDC_DIRECTION_TYPE is ( INCREASING_VDC, DECREASING_VDC ) ;

type VDC_SELECTION_TYPE is ( INTEGER_VDC, REAL_VDC ) ;

type VDC_SUPPORT_TYPE is ( INTEGER_VDC, REAL_VDC, BOTH ) ;

type VERT_ALIGNMENT_FLAG_TYPE is ( BOTTOM, CENTRE, TOP ) ;

type VERT_ALIGNMENT_TYPE is ( NORMAL_VERT,
                              TOP,
                              CAP,
                              HALF,
```

```
                                        BASE,
                                        BOTTOM,
                                        CONTINUOUS_VERT ) ;

type VISIBILITY_TYPE is ( INVISIBLE, VISIBLE ) ;

type VIEWPORT_SPECIF_MODE_TYPE is ( FRACTION_OF_DRAWING_SURFACE,
                                    MILLIMETRES_WITH_SCALE_FACTOR,
                                    PHYSICAL_DEVICE_COORDS ) ;

                    ---------- CGI Record Type Definitions ----------

type ARC_OFFSET_POINT_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    X_START : VDC_TYPE( SELECTOR ) ;
    Y_START : VDC_TYPE( SELECTOR ) ;
    X_END   : VDC_TYPE( SELECTOR ) ;
    Y_END   : VDC_TYPE( SELECTOR ) ;
  end record ;

type ASF_RECORD_TYPE is
  record
    ASF       : ASF_TYPE ;
    ASF_VALUE : ASF_VALUE_TYPE ;
  end record ;

type BITBLT_DIMENSION_TYPE is
  record
    LENGTH : INTEGER_TYPE ;
    WIDTH  : INTEGER_TYPE ;
  end record ;

type BITMAP_EXTENT_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    CORNER          : VDC_POINT_TYPE( SELECTOR ) ;
    OPPOSITE_CORNER : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;

type BITMAP_REGION_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    POINT_1 : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_2 : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;

type CELL_ARRAY_DIMENSION_TYPE is
  record
    X_DIMENSION : INTEGER_TYPE ;
    Y_DIMENSION : INTEGER_TYPE ;
  end record ;
```

```
type CELL_ARRAY_PARALLELOGRAM_TYPE
  ( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    POINT_P : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_Q : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_R : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;

type CHAR_VECTOR_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    X_COMPONENT : VDC_TYPE( SELECTOR ) ;
    Y_COMPONENT : VDC_TYPE( SELECTOR ) ;
  end record ;

type CHAR_ORIENTATION_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    CHAR_UP_VECTOR    : CHAR_VECTOR_TYPE( SELECTOR ) ;
    CHAR_BASE_VECTOR : CHAR_VECTOR_TYPE( SELECTOR ) ;
  end record ;

type CIE_CHROMATICITY_TYPE( SELECTOR : REAL_MODE_TYPE := FLOATING_POINT ) is
  record
    U : REAL_TYPE( SELECTOR ) ;
    V : REAL_TYPE( SELECTOR ) ;
    Y : REAL_TYPE( SELECTOR ) ;
  end record ;

type CIE_TRISTIMULUS_VALUE_TYPE( SELECTOR : REAL_MODE_TYPE := FLOATING_POINT ) is
  record
    X : REAL_TYPE( SELECTOR ) ;
    Y : REAL_TYPE( SELECTOR ) ;
    Z : REAL_TYPE( SELECTOR ) ;
  end record ;

type CLIP_RECTANGLE_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    CORNER          : VDC_POINT_TYPE( SELECTOR ) ;
    OPPOSITE_CORNER : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;

type COLOUR_VALUE_EXTENT_TYPE is
  record
    MIN_COLOUR_VALUE  : DIRECT_COLOUR_VALUE_TYPE ;
    MAX_COLOUR_VALUE : DIRECT_COLOUR_VALUE_TYPE ;
  end record ;

type DEVICE_VIEWPORT_TYPE( SELECTOR : VIEWPORT_MODE_TYPE := INTEGER_VIEWPORT ) is
  record
    CORNER          : VIEWPORT_POINT_TYPE( SELECTOR ) ;
    OPPOSITE_CORNER : VIEWPORT_POINT_TYPE( SELECTOR ) ;
```

```
    end record ;

  type DRAWING_MODE_SPECIFIER_TYPE is
    record
      DRAWING_MODE_CLASS : INDEX_TYPE ;
      DRAWING_OPERATION  : INDEX_TYPE ;
    end record ;

  type DRAWING_MODE_TRANSPARENCY_TYPE is
    record
      DRAWING_MODE : DRAWING_MODE_SPECIFIER_TYPE ;
      TRANSPARENCY : TRANSPARENCY_TYPE ;
    end record ;

  type DSCRECT_TYPE( SELECTOR : VIEWPORT_MODE_TYPE := INTEGER_VIEWPORT ) is
    record
      CORNER          : VIEWPORT_POINT_TYPE( SELECTOR ) ;
      OPPOSITE_CORNER : VIEWPORT_POINT_TYPE( SELECTOR ) ;
    end record ;

  type ECHO_AREA_TYPE( SELECTOR : VIEWPORT_MODE_TYPE := INTEGER_VIEWPORT ) is
    record
      POINT_1 : VIEWPORT_POINT_TYPE( SELECTOR ) ;
      POINT_2 : VIEWPORT_POINT_TYPE( SELECTOR ) ;
    end record ;

  type ERROR_CONTROL_TYPE is
    record
      ERROR_CLASS         : ERROR_CLASS_TYPE ;
      ERROR_HANDLING_FLAG : ERROR_HANDLING_FLAG_TYPE ;
    end record ;

  type EXTENT_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
    record
      DELTA_WIDTH  : VDC_TYPE( SELECTOR ) ;
      DELTA_HEIGHT : VDC_TYPE( SELECTOR ) ;
    end record ;

  type INHERITANCE_VALUE_TYPE is
    record
      FILTER_SELECTION  : FILTER_SELECTION_TYPE ;
      SELECTION_SETTING : INHERITANCE_FILTER_TYPE ;
    end record ;

  type INPUT_EXTENT_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
    record
      FIRST_CORNER  : VDC_POINT_TYPE( SELECTOR ) ;
      SECOND_CORNER : VDC_POINT_TYPE( SELECTOR ) ;
    end record ;
```

```
type INPUT_VIEWPORT_TYPE is
  record
    POINT_1 : INPUT_SURFACE_POINT_TYPE ;
    POINT_2 : INPUT_SURFACE_POINT_TYPE ;
    POINT_3 : INPUT_SURFACE_POINT_TYPE ;
  end record ;

type PATTERN_DIMENSION_TYPE is
  record
    ROWS    : INTEGER_TYPE ;
    COLUMNS : INTEGER_TYPE ;
  end record ;

type PATTERN_VECTOR_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    X_COMPONENT : VDC_TYPE( SELECTOR ) ;
    Y_COMPONENT : VDC_TYPE( SELECTOR ) ;
  end record ;

type PATTERN_SIZE_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    HEIGHT_VECTOR : PATTERN_VECTOR_TYPE( SELECTOR ) ;
    WIDTH_VECTOR  : PATTERN_VECTOR_TYPE( SELECTOR ) ;
  end record ;

type PHYSICAL_INPUT_SURFACE_SIZE_TYPE
  ( SELECTOR : REAL_MODE_TYPE := FLOATING_POINT ) is
  record
    POINT_1 : REAL_TYPE( SELECTOR ) ;
    POINT_2 : REAL_TYPE( SELECTOR ) ;
  end record ;

type PICK_APERTURE_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    HEIGHT : VDC_TYPE( SELECTOR ) ;
    WIDTH  : VDC_TYPE( SELECTOR ) ;
  end record ;

type PIXEL_DIMENSION_TYPE( SELECTOR : REAL_MODE_TYPE := FLOATING_POINT ) is
  record
    HEIGHT : REAL_TYPE( SELECTOR ) ;
    WIDTH  : REAL_TYPE( SELECTOR ) ;
  end record ;

type PROFILE_SUPPORT_TYPE is
  record
    PROFILE  : PROFILE_ID_TYPE ;
    SUPPORT : YES_NO_FLAG_TYPE ;
   end record ;
```

```
type RASTER_WINDOW_CORNER_TYPE is
  record
    X : INTEGER_TYPE ;
    Y : INTEGER_TYPE ;
  end record ;

type RECTANGLE_DESCRIPTOR_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    CORNER          : VDC_POINT_TYPE( SELECTOR ) ;
    OPPOSITE_CORNER : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;

type SCALING_ROTATION_MATRIX_TYPE is array
  ( ARRAY_INDEX_TYPE range 1..2, ARRAY_INDEX_TYPE range 1..2 ) of REAL_TYPE ;

type TRANSLATION_MATRIX_TYPE is array ( ARRAY_INDEX_TYPE range 1..2 ) of VDC_TYPE ;

type SEGMENT_TRANSFORMATION_TYPE is
  record
    SCALING_ROTATION_PORTION : SCALING_ROTATION_MATRIX_TYPE ;
    TRANSLATION_PORTION      : TRANSLATION_MATRIX_TYPE ;
  end record ;

type SPOT_CENTRE_SEPARATION_TYPE
  ( SELECTOR : REAL_MODE_TYPE := FLOATING_POINT ) is
  record
    X_SEPARATION : REAL_TYPE( SELECTOR ) ;
    Y_SEPARATION : REAL_TYPE( SELECTOR ) ;
  end record ;

type STROKE_SAMPLING_INTERVAL_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    X_DISPLACEMENT_INTERVAL : VDC_TYPE( SELECTOR ) ;
    Y_DISPLACEMENT_INTERVAL : VDC_TYPE( SELECTOR ) ;
  end record ;

type TEXT_EXTENT_PARALLELOGRAM_TYPE
  ( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    POINT_1 : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_2 : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_3 : VDC_POINT_TYPE( SELECTOR ) ;
    POINT_4 : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;

type VALID_PIXEL_RANGE_TYPE is
  record
    STARTING_INDEX : INTEGER_TYPE ;
    ENDING_INDEX   : INTEGER_TYPE ;
  end record ;
```

```
type VDC_EXTENT_TYPE( SELECTOR : VDC_MODE_TYPE := INTEGER_VDC ) is
  record
    CORNER          : VDC_POINT_TYPE( SELECTOR ) ;
    OPPOSITE_CORNER : VDC_POINT_TYPE( SELECTOR ) ;
  end record ;

type XY_STEP_TYPE is
  record
    STEPS_IN_X : INTEGER_TYPE ;
    STEPS_IN_Y : INTEGER_TYPE ;
  end record ;

                    ---------- CGI Array Type Definitions ----------

-- Define the VDC coordinate and point arrays.

type VDC_INTEGER_COORD_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_INTEGER_TYPE ;

type VDC_FIXED_COORD_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_FIXED_POINT_REAL_TYPE ;

type VDC_FLOAT_COORD_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_FLOATING_POINT_REAL_TYPE ;

type VDC_INTEGER_PAIR_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_INTEGER_PAIR_TYPE ;

type VDC_FIXED_PAIR_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_FIXED_PAIR_TYPE ;

type VDC_FLOAT_PAIR_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of VDC_FLOAT_PAIR_TYPE ;

-- Define the rest of the arrays used throughout the CGI.

subtype ASF_INDEX_TYPE is ARRAY_INDEX_TYPE range 1..20 ;

type ASF_ARRAY_TYPE is array ( ASF_INDEX_TYPE range <> ) of ASF_TYPE ;

type ASF_RECORD_ARRAY_TYPE is array ( ASF_INDEX_TYPE range <> )
  of ASF_RECORD_TYPE ;

type ASF_VALUE_ARRAY_TYPE is array ( ASF_INDEX_TYPE range <> )
  of ASF_VALUE_TYPE ;

type ASN_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of ASN_TYPE ;

type BITMAP_ID_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of BITMAP_ID_TYPE ;
```

```
type BITMAP_MODE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..3 )
  of BITMAP_MODE_TYPE ;

type BUNDLE_TABLE_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range 1..5 ) of BUNDLE_TABLE_TYPE ;

type CHAR_CODING_ANNOUNCER_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range 1..5 ) of CODING_TECHNIQUE_TYPE ;

type CHAR_SET_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of CHAR_SET_TYPE ;

type CLIP_MODE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..3 ) of CLIP_MODE_TYPE ;

type INDEX_COLOUR_MATRIX_TYPE is array
  ( ARRAY_INDEX_TYPE range <>, ARRAY_INDEX_TYPE range <> ) of COLOUR_INDEX_TYPE ;

type DIRECT_COLOUR_MATRIX_TYPE is array
  ( ARRAY_INDEX_TYPE range <>, ARRAY_INDEX_TYPE range <> )
  of DIRECT_COLOUR_VALUE_TYPE ;

type COLOUR_ARRAY_TYPE(
  COLOUR_SELECTION_MODE : COLOUR_SELECTION_MODE_TYPE := INDEXED ;
  NUM_OF_ROWS           : ARRAY_INDEX_TYPE := ARRAY_INDEX_TYPE'LAST ;
  COLOURS_PER_ROW       : ARRAY_INDEX_TYPE := ARRAY_INDEX_TYPE'LAST ) is
  record
    case COLOUR_SELECTION_MODE is
      when INDEXED => INDEX_COLOUR_DATA : INDEX_COLOUR_MATRIX_TYPE
        ( 1..NUM_OF_ROWS, 1..COLOURS_PER_ROW ) ;
      when DIRECT  => DIRECT_COLOUR_DATA : DIRECT_COLOUR_MATRIX_TYPE
        ( 1..NUM_OF_ROWS, 1..COLOURS_PER_ROW ) ;
    end case ;
  end record ;

type COLOUR_INDEX_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of COLOUR_INDEX_TYPE ;

type DIRECT_COLOUR_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of DIRECT_COLOUR_VALUE_TYPE ;

type COLOUR_SPECIFIER_ARRAY_TYPE
  ( COLOUR_SELECTION_MODE : COLOUR_SELECTION_MODE_TYPE := INDEXED ;
    LENGTH : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
  record
    case COLOUR_SELECTION_MODE is
      when INDEXED =>
        INDEX_COLOUR_DATA : COLOUR_INDEX_ARRAY_TYPE( 1..LENGTH ) ;
      when DIRECT =>
        DIRECT_COLOUR_DATA : DIRECT_COLOUR_ARRAY_TYPE( 1..LENGTH ) ;
    end case ;
  end record ;
```

```
subtype COLOUR_TABLE_RANGE_TYPE is COLOUR_INDEX_TYPE range
  CGI_CONFIG.FIRST_ENTRY_IN_COLOUR_TABLE..
  CGI_CONFIG.LAST_ENTRY_IN_COLOUR_TABLE ;

type COLOUR_TABLE_TYPE is array ( COLOUR_TABLE_RANGE_TYPE range <> )
  of DIRECT_COLOUR_VALUE_TYPE ;

type CSN_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of CSN_TYPE ;

type DEFERRAL_MODE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..3 )
  of DEFERRAL_MODE_TYPE ;

type DEVICE_COORD_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of DEVICE_COORD_TYPE ;

type DRAWING_MODE_TRANSPARENCY_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of DRAWING_MODE_TRANSPARENCY_TYPE ;

type EDGE_OUT_FLAG_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of EDGE_OUT_FLAG_TYPE ;

type ERROR_HANDLING_ARRAY_TYPE is array ( ERROR_CLASS_TYPE range <> )
  of ERROR_CONTROL_TYPE ;

type ERROR_HANDLING_FLAG_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range 1..9 ) of ERROR_HANDLING_FLAG_TYPE ;

type ERROR_QUEUE_TYPE is array ( ARRAY_INDEX_TYPE range 1..CGI_CONFIG.
  SIZE_OF_INTERPRETER_ERROR_QUEUE ) of ERROR_REPORT_TYPE ;

subtype FILTER_SELECTION_INDEX_TYPE is ARRAY_INDEX_TYPE range 1 .. 75;

type FILTER_SELECTION_LIST_TYPE is array
  ( FILTER_SELECTION_INDEX_TYPE range <> ) of FILTER_SELECTION_TYPE ;

type FIXED_INTEGER_8_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FIXED_INTEGER_8_TYPE ;

type FIXED_INTEGER_16_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FIXED_INTEGER_16_TYPE ;

type FIXED_INTEGER_32_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FIXED_INTEGER_32_TYPE ;

type FIXED_POINT_REAL_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FIXED_POINT_REAL_TYPE ;

type FIXED_STRING_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FIXED_STRING_TYPE ;
```

```
type FLOATING_POINT_REAL_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FLOATING_POINT_REAL_TYPE ;

type FUNCTION_ID_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of FUNCTION_ID_TYPE ;

type HOR_ALIGNMENT_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..5 )
  of HOR_ALIGNMENT_TYPE ;

type INDEX_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of INDEX_TYPE ;

type INHERITANCE_VALUE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of INHERITANCE_VALUE_TYPE ;

type INPUT_COLOUR_INDEX_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of INPUT_COLOUR_INDEX_TYPE ;

type DIRECT_INPUT_COLOUR_ARRAY_TYPE is array
  ( ARRAY_INDEX_TYPE range <> ) of DIRECT_INPUT_COLOUR_VALUE_TYPE ;

type INPUT_COLOUR_ARRAY_TYPE
  ( COLOUR : YES_NO_FLAG_TYPE          := NO ;
    LENGTH : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
  record
    case COLOUR is
      when NO  =>
        INDEX_COLOUR_DATA  : INPUT_COLOUR_INDEX_ARRAY_TYPE( 1..LENGTH ) ;
      when YES =>
        DIRECT_COLOUR_DATA : DIRECT_INPUT_COLOUR_ARRAY_TYPE( 1..LENGTH ) ;
    end case ;
  end record ;

type INTEGER_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of INTEGER_TYPE ;

subtype INTENSITY_ARRAY_TYPE is INTEGER_ARRAY_TYPE( 1..3 ) ;

type INTERIOR_STYLE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..6 )
  of INTERIOR_STYLE_TYPE ;

type ORIENTATION_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of CHAR_ORIENTATION_TYPE ;

type PICK_VALUE_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
  of PICK_VALUE_TYPE ;

type POINT_LIST_TYPE
  ( VDC_TYPE : VDC_MODE_TYPE          := INTEGER_VDC ;
    LENGTH   : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
  record
    case VDC_TYPE is
```

```
        when INTEGER_VDC =>
          INTEGER_DATA : VDC_INTEGER_PAIR_ARRAY_TYPE( 1..LENGTH ) ;
        when FIXED_POINT_VDC =>
          FIXED_DATA   : VDC_FIXED_PAIR_ARRAY_TYPE( 1..LENGTH ) ;
        when FLOATING_POINT_VDC =>
          FLOAT_DATA   : VDC_FLOAT_PAIR_ARRAY_TYPE( 1..LENGTH ) ;
      end case ;
    end record ;

  type FLAGGED_POINT_LIST_TYPE
    ( VDC_TYPE : VDC_MODE_TYPE                := INTEGER_VDC ;
      LENGTH   : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
    record
      POINT_LIST : POINT_LIST_TYPE( VDC_TYPE, LENGTH ) ;
      FLAG_LIST  : EDGE_OUT_FLAG_ARRAY_TYPE( 1..LENGTH ) ;
    end record ;

  type PREC_REQUIREMENT_TYPE is array ( ARRAY_INDEX_TYPE range 1..3 ) of INTEGER_TYPE ;

  type PROFILE_ID_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of PROFILE_ID_TYPE ;

  type PROFILE_SUPPORT_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
    of PROFILE_SUPPORT_TYPE ;

  type PROFILE_SUPPORT_FLAG_ARRAY_TYPE is array
    ( ARRAY_INDEX_TYPE range <> ) of PROFILE_SUPPORT_FLAG_TYPE ;

  type REAL_ARRAY_TYPE
    ( REAL_MODE : REAL_MODE_TYPE    := FLOATING_POINT ;
      LENGTH    : ARRAY_INDEX_TYPE := ARRAY_INDEX_TYPE'LAST ) is
    record
      case REAL_MODE is
        when FLOATING_POINT =>
          FLOAT_DATA : FLOATING_POINT_REAL_ARRAY_TYPE( 1..LENGTH ) ;
        when FIXED_POINT =>
          FIXED_DATA  : FIXED_POINT_REAL_ARRAY_TYPE( 1..LENGTH ) ;
      end case ;
    end record ;

  type SEGMENT_ID_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
    of SEGMENT_ID_TYPE ;

  type STRING_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> ) of STRING_TYPE ;

  type TEXT_PATH_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..4 )
    of TEXT_PATH_TYPE ;

  type VDC_ARRAY_TYPE
    ( VDC_TYPE : VDC_MODE_TYPE                := INTEGER_VDC ;
      LENGTH   : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
```

```
    record
      case VDC_TYPE is
        when INTEGER_VDC =>
          INTEGER_DATA : VDC_INTEGER_COORD_ARRAY_TYPE( 1..LENGTH ) ;
        when FIXED_POINT_VDC =>
          FIXED_DATA   : VDC_FIXED_COORD_ARRAY_TYPE( 1..LENGTH ) ;
        when FLOATING_POINT_VDC =>
          FLOAT_DATA   : VDC_FLOAT_COORD_ARRAY_TYPE( 1..LENGTH ) ;
      end case ;
    end record ;

  type VERT_ALIGNMENT_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range 1..7 )
    of VERT_ALIGNMENT_TYPE ;

  type VIEWPORT_COORD_ARRAY_TYPE
    ( VIEWPORT_TYPE : VIEWPORT_MODE_TYPE       := INTEGER_VIEWPORT ;
      LENGTH        : ARRAY_INDEX_RANGE_TYPE := ARRAY_INDEX_TYPE'LAST ) is
    record
      case VIEWPORT_TYPE is
        when INTEGER_VIEWPORT =>
          INTEGER_DATA : INTEGER_ARRAY_TYPE( 1..LENGTH ) ;
        when FIXED_POINT_VIEWPORT =>
          FIXED_DATA   : FIXED_POINT_REAL_ARRAY_TYPE( 1..LENGTH ) ;
        when FLOATING_POINT_VIEWPORT =>
          FLOAT_DATA   : FLOATING_POINT_REAL_ARRAY_TYPE( 1..LENGTH ) ;
      end case ;
    end record ;

  type VIEWPORT_SPECIF_MODE_ARRAY_TYPE is array
    ( ARRAY_INDEX_TYPE range 1..3 ) of VIEWPORT_SPECIF_MODE_TYPE ;

  type YES_NO_FLAG_ARRAY_TYPE is array ( ARRAY_INDEX_TYPE range <> )
    of YES_NO_FLAG_TYPE ;

                    ---------- CGI Access Declarations ----------

  type ACCESS_POINT_LIST_TYPE is access POINT_LIST_TYPE ;

  type ACCESS_INPUT_COLOUR_ARRAY_TYPE is access INPUT_COLOUR_ARRAY_TYPE ;

  type ACCESS_PICK_VALUE_ARRAY_TYPE is access PICK_VALUE_ARRAY_TYPE ;

  type ACCESS_STRING_TYPE is access STRING_TYPE ;

                    ---------- CGI Exception Declarations ----------

  CGI_ERROR : exception ;

                    ---------- Private Part ----------
```

```
private

  for YES_NO_FLAG_TYPE use ( NO => 0, YES => 1 ) ;

  for REAL_MODE_TYPE use ( FIXED_POINT => 0, FLOATING_POINT => 1 ) ;

  for VDC_MODE_TYPE use
    ( INTEGER_VDC => 0, FIXED_POINT_VDC => 1, FLOATING_POINT_VDC => 2 ) ;

  for COLOUR_SELECTION_MODE_TYPE use ( INDEXED => 0, DIRECT => 1 ) ;


  for SPECIF_MODE_TYPE use ( VDC => 0, SCALED => 1 ) ;

  for VIEWPORT_MODE_TYPE use ( INTEGER_VIEWPORT          => 0,
                               FIXED_POINT_VIEWPORT      => 1,
                               FLOATING_POINT_VIEWPORT => 2 ) ;

  for CHAR_SET_ID_TYPE use ( CHAR_SET_94                => 0,
                             CHAR_MULTIBYTE_SET_94 => 1,
                             CHAR_SET_96                => 2,
                             CHAR_MULTIBYTE_SET_96 => 3,
                             COMPLETE_CODE              => 4 ) ;

  for ACTION_REQUIRED_FLAG_TYPE use
    ( NO_ACTION_REQUIRED => 0, ACTION_REQUIRED => 1 ) ;

  for ASF_TYPE use ( LINE_TYPE_ASF          => 0,
                     LINE_WIDTH_ASF         => 1,
                     LINE_COLOUR_ASF        => 2,
                     MARKER_TYPE_ASF        => 3,
                     MARKER_SIZE_ASF        => 4,
                     MARKER_COLOUR_ASF      => 5,
                     TEXT_FONT_INDEX_ASF    => 6,
                     TEXT_PREC_ASF          => 7,
                     CHAR_EXPAN_FACTOR_ASF => 8,
                     CHAR_SPACING_ASF       => 9,
                     TEXT_COLOUR_ASF        => 10,
                     INTERIOR_STYLE_ASF     => 11,
                     FILL_COLOUR_ASF        => 12,
                     HATCH_INDEX_ASF        => 13,
                     PATTERN_INDEX_ASF      => 14,
                     EDGE_TYPE_ASF          => 15,
                     EDGE_WIDTH_ASF         => 16,
                     EDGE_COLOUR_ASF        => 17,
                     EDGE_VISIBILITY_ASF    => 18,
                     FILL_BITMAP_ASF        => 19 ) ;

  for ASF_VALUE_TYPE use ( INDIVIDUAL => 0, BUNDLED => 1 ) ;
```

```
for BACKGROUND_COLOUR_CAP_TYPE use
  ( NONE => 0, INDEX_0 => 1, INDEXED => 2, FULL => 3 ) ;

for BLOCK_CONTROL_TYPE use ( NOT_BLOCKED => 0, BLOCKED => 1 ) ;

for BITMAP_MODE_TYPE use ( INDEXED => 0, DIRECT => 1, MIXED => 2 ) ;

for BITMAP_TRUNCATION_TYPE use ( NOT_TRUNCATED => 0, TRUNCATED => 1 ) ;

for BUNDLE_TABLE_TYPE use ( LINE      => 0,
                            MARKER    => 1,
                            TEXT      => 2,
                            FILL      => 3,
                            EDGE      => 4 ) ;

for CELL_ARRAY_ALIGNMENT_CAP_TYPE use ( AXIS_ALIGNED => 0, SKEWED => 1 ) ;

for CELL_ARRAY_FILL_CAP_TYPE use ( OUTLINED => 0, FILLED => 1 ) ;

for CELL_ARRAY_TECHNIQUE_TYPE use ( OTHER => 0, EXACT => 1 ) ;

for CLEAR_DRAWING_SURFACE_TYPE use ( UNCONDITIONAL => 0,  CONDITIONAL => 1 ) ;

for CLIP_EFFECTIVENESS_TYPE use ( NONE => 0, CIRCUMSCRIBED => 1, FULL => 2) ;

for CLIP_INHERITANCE_FILTER_TYPE use ( STATE_LIST => 0,  INTERSECTION => 1 ) ;

for CLIP_MODE_TYPE use ( LOCUS => 0, SHAPE => 1, LOCUS_THEN_SHAPE => 2 ) ;

for CLOSE_TYPE use ( PIE => 0, CHORD => 1 ) ;

for CODING_TECHNIQUE_TYPE use ( BASIC_7_BIT       => 0,
                                BASIC_8_BIT       => 1,
                                EXTENDED_7_BIT    => 2,
                                EXTENDED_8_BIT    => 3,
                                PRIVATE_CODING    => 4 ) ;

for COLOUR_REALIZATION_TYPE use ( ADDITIVE => 0, SUBTRACTIVE => 1 ) ;

for COLOUR_SELECTION_MODE_AVAIL_TYPE use
  ( INDEXED_ONLY => 0,  INDEXED_AND_DIRECT => 1 ) ;

for COMPOUND_OBJECT_CAP_TYPE use ( NONE => 0, GLOBAL => 1, LOCAL => 2 ) ;

for DEFERRAL_MODE_TYPE use ( ASTI => 0, BNI => 1, ASAP => 2 ) ;

for DEPTH_TYPE use ( MAPPED => 0, FULL_DEPTH => 1 ) ;

for DEPTHS_SUPPORTED_TYPE use ( FULL_DEPTH => 0, MAPPED_AND_FULL_DEPTH => 1 ) ;
```

```
for DETECTABILITY_TYPE use ( UNDETECTABLE => 0, DETECTABLE => 1 ) ;

for DEVICE_CLASS_TYPE use ( OUTPUT => 0, INPUT => 1, OUTIN => 2 ) ;

for DISABLED_ENABLED_FLAG_TYPE use ( DISABLED => 0, ENABLED => 1 ) ;

for DISPLAYABILITY_TYPE use ( NON_DISPLAYABLE  => 0, DISPLAYABLE => 1 ) ;

for DISPLAY_BITMAP_DATA_TYPE use ( CLEARED => 0, PRESERVED => 1 ) ;

for DISPLAY_TYPE use ( VECTOR => 0, RASTER => 1, OTHER => 2 ) ;

for DRAWING_MODE_SUPPORT_TYPE use ( NONE => 0, SOME => 1, ALL_MODES => 2 ) ;

for DRAWING_SURFACE_CLIP_INDICATOR_TYPE use
  ( DSCRECT => 0,  VIEWPORT  => 1,  OFF => 2 ) ;

for DRAWING_SURFACE_STATE_TYPE use ( DIRTY => 0, CLEAN => 1 ) ;

for DYN_MOD_FLAG_TYPE use ( IRG => 0, CBS => 1, IMM => 2 ) ;

for ECHO_ENTITY_ECHO_CONTROL_TYPE use ( ECHO_OFF => 0, ECHO_ON => 1 ) ;

for ECHO_ENTITY_PROMPT_CONTROL_TYPE use ( PROMPT_OFF => 0, PROMPT_ON => 1 ) ;

for ECHO_ENTITY_STATE_TYPE use ( READY => 0, ACTIVE => 1 ) ;

for EDGE_OUT_FLAG_TYPE use
  ( INVISIBLE => 0, VISIBLE => 1, CLOSE_INVISIBLE => 2, CLOSE_VISIBLE  => 3 ) ;

for EDGE_VISIBILITY_TYPE use ( VISIBLE => 0, INVISIBLE => 1 ) ;

for EDGE_WIDTH_REALIZATION_TYPE use ( INTERIOR => 0, CENTRED => 1 ) ;

for EFFECTIVE_PRIORITY_TYPE use
  ( OTHER => 0, TIME_ORDER_REOPEN => 1, TIME_ORDER => 2 ) ;

for ERROR_HANDLING_FLAG_TYPE use
  ( ON => 0, REPORTING_OFF => 1, DETECTION_OFF  => 2 ) ;

for EVENT_QUEUE_STATE_TYPE use ( RELEASED                    => 0,
                                 EMPTY_NO_LID_ENABLED        => 1,
                                 NOT_EMPTY_NO_LID_ENABLED    => 2,
                                 EMPTY_LID_ENABLED           => 3,
                                 NOT_EMPTY_LID_ENABLED       => 4 ) ;

for EVENT_STATUS_TYPE use ( TIMEOUT => 0, EVENT => 1, OVERFLOW => 2,  BREAK => 3 ) ;

for FILTER_SELECTION_TYPE use ( LINE_TYPE_ASF                => 0,
                                LINE_WIDTH_ASF               => 1,
```

```
LINE_COLOUR_ASF                => 2,
MARKER_TYPE_ASF                => 3,
MARKER_SIZE_ASF                => 4,
MARKER_COLOUR_ASF              => 5,
TEXT_FONT_INDEX_ASF            => 6,
TEXT_PREC_ASF                  => 7,
CHAR_EXPANSION_FACTOR_ASF      => 8,
CHAR_SPACING_ASF               => 9,
TEXT_COLOUR_ASF                => 10,
INTERIOR_STYLE_ASF             => 11,
FILL_COLOUR_ASF                => 12,
HATCH_INDEX_ASF                => 13,
PATTERN_INDEX_ASF              => 14,
FILL_BITMAP_ASF                => 15,
EDGE_TYPE_ASF                  => 16,
EDGE_WIDTH_ASF                 => 17,
EDGE_COLOUR_ASF                => 18,
EDGE_VISIBILITY_ASF            => 19,
LINE_BUNDLE_INDEX              => 20,
LINE_TYPE                      => 21,
LINE_WIDTH                     => 22,
LINE_COLOUR                    => 23,
LINE_CLIPPING_MODE             => 24,
MARKER_BUNDLE_INDEX            => 25,
MARKER_TYPE                    => 26,
MARKER_SIZE                    => 27,
MARKER_COLOUR                  => 28,
MARKER_CLIPPING_MODE           => 29,
TEXT_BUNDLE_INDEX              => 30,
TEXT_FONT_INDEX                => 31,
TEXT_COLOUR                    => 32,
CHAR_EXPANSION_FACTOR          => 33,
CHAR_SPACING                   => 34,
CHAR_HEIGHT                    => 35,
CHAR_ORIENTATION               => 36,
TEXT_PREC                      => 37,
TEXT_PATH                      => 38,
TEXT_ALIGNMENT                 => 39,
FILL_BUNDLE_INDEX              => 40,
INTERIOR_STYLE                 => 41,
FILL_COLOUR                    => 42,
HATCH_INDEX                    => 43,
PATTERN_INDEX                  => 44,
FILL_BITMAP                    => 45,
EDGE_BUNDLE_INDEX              => 46,
EDGE_TYPE                      => 47,
EDGE_WIDTH                     => 48,
EDGE_COLOUR                    => 49,
EDGE_VISIBILITY                => 50,
EDGE_CLIPPING_MODE             => 51,
```

```
                                    FILL_REFERENCE_POINT        => 52,
                                    PATTERN_SIZE                => 53,
                                    AUXILIARY_COLOUR            => 54,
                                    TRANSPARENCY                => 55,
                                    DRAWING_MODE                => 56,
                                    PICK_ID                     => 57,
                                    LINE_ASFS                   => 58,
                                    MARKER_ASFS                 => 59,
                                    TEXT_ASFS                   => 60,
                                    FILL_ASFS                   => 61,
                                    EDGE_ASFS                   => 62,
                                    ALL_ASFS                    => 63,
                                    LINE_ATTRIBUTES             => 64,
                                    MARKER_ATTRIBUTES           => 65,
                                    LOCAL_TEXT_ATTRIBUTES       => 66,
                                    GLOBAL_TEXT_ATTRIBUTES      => 67,
                                    FILL_ATTRIBUTES             => 68,
                                    EDGE_ATTRIBUTES             => 69,
                                    PATTERN_ATTRIBUTES          => 70,
                                    OUTPUT_CONTROL              => 71,
                                    PICK_ATTRIBUTES             => 72,
                                    ALL_ATTRIBUTES              => 73,
                                    ALL_ATTRIBUTES_AND_ASFS     => 74 ) ;

for FINAL_FLAG_TYPE use ( NOT_FINAL => 0, FINAL => 1 ) ;

for HARD_SOFT_COPY_TYPE use ( HARD => 0, SOFT => 1 ) ;


for HIGHLIGHTING_TYPE use ( NORMAL => 0, HIGHLIGHTED => 1 ) ;

for HOR_ALIGNMENT_FLAG_TYPE use ( LEFT => 0, CENTRE => 1, RIGHT => 2 ) ;

for HOR_ALIGNMENT_TYPE use
  ( NORMAL_HOR => 0, LEFT => 1, CENTRE => 2, RIGHT => 3, CONTINUOUS_HOR  => 4 ) ;

for IMPLICIT_SEGMENT_REGEN_MODE_TYPE use
  ( SUPPRESSED => 0, UQUM => 1, ALLOWED => 2 ) ;

for INHERITANCE_FILTER_TYPE use ( STATE_LIST => 0, SEGMENT => 1 ) ;

for INPUT_CLASS_TYPE use ( LOCATOR_LID   => 0,
                           STROKE_LID    => 1,
                           VALUATOR_LID  => 2,
                           CHOICE_LID    => 3,
                           PICK_LID      => 4,
                           STRING_LID    => 5,
                           RASTER_LID    => 6,
                           GENERAL_LID   => 7 ) ;
```

```
for INPUT_DEVICE_STATE_TYPE use ( RELEASED                => 0,
                                  READY                   => 1,
                                  REQUEST_PENDING         => 2,
                                  ECHO_REQUEST_ENABLED    => 3,
                                  ECHO_REQUEST_PENDING    => 4,
                                  ECHO_REQUEST_COMPLETED  => 5,
                                  EVENTS_ENABLED          => 6 ) ;

for INQUIRY_SOURCE_FLAG_TYPE use ( CURRENT => 0, DEFAULT => 1 ) ;

for INTERIOR_STYLE_TYPE use ( HOLLOW  => 0,
                              SOLID   => 1,
                              PATTERN => 2,
                              HATCH   => 3,
                              EMPTY   => 4,
                              BITMAP  => 5 ) ;

for ISOTROPY_FLAG_TYPE use ( NOT_FORCED => 0, FORCED => 1 ) ;

for LINE_EDGE_CONTINUITY_CAP_TYPE use
  ( RESTART => 0, CONTINUOUS  => 1, OTHER => 2 ) ;

for ON_OFF_FLAG_TYPE use ( OFF => 0, ON => 1 ) ;

for OUTPUT_STATE_TYPE use ( ACTIVE => 0, TEXT_OPEN => 1, FIGURE_OPEN => 2 ) ;

for PATTERN_FILL_FALLBACK_TYPE use ( CONDENSE => 0, TRUNCATE => 1 ) ;

for PATTERN_TRAN_SUPPORT_TYPE use ( NONE => 0, UNSKEWED => 1, FULL => 2 ) ;

for PICK_APERTURE_SHAPE_TYPE use ( RECTANGLE => 0, CIRCLE => 1, ELLIPSE => 2 ) ;

for PIXEL_RELATIVE_LOCATION_TYPE use ( ON => 0, BETWEEN => 1 ) ;

for PIXEL_VALIDITY_FLAG_TYPE use ( NONE => 0, SOME => 1, ALL_VALID => 2 ) ;

for PROFILE_SUPPORT_FLAG_TYPE use ( NO => 0, YES => 1, UNRECOGNIZED => 2 ) ;

for REQUEST_STATUS_TYPE use ( TRIGGER_FIRED => 0, BREAK => 1, TIMEOUT => 2 ) ;

for SEGMENT_OPEN_STATE_TYPE use ( NO => 0, YES => 1, OVERFLOW => 2 ) ;

for SPOT_CENTRE_INTERPRETATION_TYPE use ( NOMINAL => 0, ACTUAL => 1 ) ;

for SURFACE_INTERPRETATION_TYPE use ( NOMINAL => 0,  ACTUAL => 1, UNLIMITED => 2 ) ;

for TEXT_PATH_TYPE use ( RIGHT => 0, LEFT => 1, UP => 2, DOWN => 3 ) ;

for TEXT_PREC_TYPE use ( STRING_PREC => 0, CHAR_PREC => 1, STROKE_PREC => 2 ) ;
```

```
        for TIMEOUT_CAP_TYPE use ( LIMITED_CAP => 0, FULL_CAP => 1 ) ;

        for TIMESTAMP_SUPPORT_TYPE use ( NA => 0, TRIGGER_COUNT => 1, CLOCK => 2 ) ;

        for TRANSPARENCY_TYPE use ( OPAQUE => 0, TRANSPARENT => 1 ) ;

        for UNREPORTED_BREAK_STATE_TYPE use ( NO_BREAK => 0, BREAK => 1 ) ;

        for UNREPORTED_OVERFLOW_STATE_TYPE use ( NO_OVERFLOW => 0, OVERFLOW => 1 ) ;

        for VALIDITY_FLAG_TYPE use ( INVALID => 0, VALID => 1 ) ;

        for VDC_DIRECTION_TYPE use ( INCREASING_VDC => 0, DECREASING_VDC => 1 ) ;

        for VDC_SELECTION_TYPE use ( INTEGER_VDC => 0, REAL_VDC => 1 ) ;

        for VDC_SUPPORT_TYPE use ( INTEGER_VDC => 0, REAL_VDC => 1, BOTH => 2 ) ;

        for VERT_ALIGNMENT_FLAG_TYPE use ( BOTTOM => 0, CENTRE => 1, TOP => 2 ) ;

        for VERT_ALIGNMENT_TYPE use ( NORMAL_VERT    => 0,
                                      TOP            => 1,
                                      CAP            => 2,
                                      HALF           => 3,
                                      BASE           => 4,
                                      BOTTOM         => 5,
                                      CONTINUOUS_VERT => 6 ) ;

        for VISIBILITY_TYPE use ( INVISIBLE => 0, VISIBLE => 1 ) ;

        for VIEWPORT_SPECIF_MODE_TYPE use ( FRACTION_OF_DRAWING_SURFACE    => 0,
                                            MILLIMETRES_WITH_SCALE_FACTOR => 1,
                                            PHYSICAL_DEVICE_COORDS        => 2 ) ;
end CGI_TYPES ;
```

A.3 Package specification CGI_DATA_RECORD_UTILS

```
with CGI_TYPES ;
with CGI_CONFIG ;

package CGI_DATA_RECORD_UTILS is

   -- Package: CGI_DATA_RECORD_UTILS
   -- Functional Description:
   --
   -- This package provides the necessary utilities to construct and interpret CGI data records.

   -- Data record type definition.

   subtype DATA_RECORD_LENGTH_TYPE is CGI_TYPES.ARRAY_INDEX_RANGE_TYPE
     range 0..CGI_CONFIG.MAX_DATA_RECORD_SIZE ;

   type DATA_RECORD_TYPE
     ( LENGTH : DATA_RECORD_LENGTH_TYPE := CGI_CONFIG.MAX_DATA_RECORD_SIZE )
     is private ;

   type ACCESS_DATA_RECORD_TYPE is access DATA_RECORD_TYPE ;

   -- Data record subtypes used throughout CGI.

   subtype EVENT_REPORTS_LIST_TYPE is ACCESS_DATA_RECORD_TYPE ;

   type EVENT_TYPE(
     INPUT_CLASS : CGI_TYPES.INPUT_CLASS_TYPE := CGI_TYPES.LOCATOR_LID ) is
     record
       LID_INDEX          : CGI_TYPES.INDEX_TYPE ;
       TIMESTAMP          : CGI_TYPES.REAL_TYPE ;
       TRIGGER            : CGI_TYPES.INDEX_TYPE ;
       MEASURE_VALIDITY   : CGI_TYPES.VALIDITY_FLAG_TYPE ;

       case INPUT_CLASS is
         when CGI_TYPES.LOCATOR_LID =>
           LOCATOR_MEASURE   : CGI_TYPES.VDC_POINT_TYPE ;

         when CGI_TYPES.STROKE_LID =>
           STROKE_MEASURE    : CGI_TYPES.ACCESS_POINT_LIST_TYPE ;

         when CGI_TYPES.VALUATOR_LID =>
           VALUATOR_MEASURE : CGI_TYPES.REAL_TYPE ;

         when CGI_TYPES.CHOICE_LID =>
           CHOICE_MEASURE    : CGI_TYPES.INTEGER_TYPE ;
```

```
        when CGI_TYPES.PICK_LID =>
          PICK_MEASURE        : CGI_TYPES.ACCESS_PICK_VALUE_ARRAY_TYPE ;

        when CGI_TYPES.STRING_LID =>
          STRING_MEASURE      : CGI_TYPES.ACCESS_STRING_TYPE ;

        when CGI_TYPES.RASTER_LID =>
          XCOUNT              : CGI_TYPES.INTEGER_TYPE ;
          YCOUNT              : CGI_TYPES.INTEGER_TYPE ;
          RASTER_MEASURE      : CGI_TYPES.ACCESS_INPUT_COLOUR_ARRAY_TYPE ;

        when CGI_TYPES.GENERAL_LID =>
          GENERAL_MEASURE     : ACCESS_DATA_RECORD_TYPE ;
      end case ;
    end record ;

  type DATA_RECORD_ARRAY_TYPE is array
    ( CGI_TYPES.ARRAY_INDEX_TYPE range <> ) of DATA_RECORD_TYPE ;

  -- Data record sub-sequence header.  This is the header for data record sub-sequence insertion and
  -- removals.  This header is derived from the sub-sequence header description in the ISO/IEC 9637
  -- Data Stream Binding - Binary Encoding.

  type DATA_RECORD_SUB_SEQUENCE_HEADER_TYPE is
    record
      PARAMETER_TYPE    : CGI_TYPES.INDEX_TYPE ;
      PARAMETER_COUNT : CGI_TYPES.INTEGER_TYPE ;
    end record ;

  -- Data record parameter identifiers

  DATA_RECORD_PARAMETER_DATA      : constant CGI_TYPES.INDEX_TYPE := 1 ;
  COLOUR_INDEX_PARAMETER_DATA     : constant CGI_TYPES.INDEX_TYPE := 2 ;
  COLOUR_DIRECT_PARAMETER_DATA    : constant CGI_TYPES.INDEX_TYPE := 3 ;
  CSN_PARAMETER_DATA              : constant CGI_TYPES.INDEX_TYPE := 4 ;
  ENUMERATED_PARAMETER_DATA       : constant CGI_TYPES.INDEX_TYPE := 5 ;
  INTEGER_PARAMETER_DATA          : constant CGI_TYPES.INDEX_TYPE := 6 ;
  INPUT_COLOUR_PARAMETER_DATA     : constant CGI_TYPES.INDEX_TYPE := 7 ;
  FIXED_INTEGER_8_PARAMETER_DATA  : constant CGI_TYPES.INDEX_TYPE := 8 ;
  FIXED_INTEGER_16_PARAMETER_DATA : constant CGI_TYPES.INDEX_TYPE := 9 ;
  FIXED_INTEGER_32_PARAMETER_DATA : constant CGI_TYPES.INDEX_TYPE := 10 ;
  INDEX_PARAMETER_DATA            : constant CGI_TYPES.INDEX_TYPE := 11 ;
  REAL_PARAMETER_DATA             : constant CGI_TYPES.INDEX_TYPE := 12 ;
  STRING_PARAMETER_DATA           : constant CGI_TYPES.INDEX_TYPE := 13 ;
  FIXED_STRING_PARAMETER_DATA     : constant CGI_TYPES.INDEX_TYPE := 14 ;
  VIEWPORT_COORD_PARAMETER_DATA   : constant CGI_TYPES.INDEX_TYPE := 15 ;
  VDC_PARAMETER_DATA              : constant CGI_TYPES.INDEX_TYPE := 16 ;

  -- Define the Add and Remove status types.  These types will indicate the status for inserting and
  -- removing data record sub-sequences to and from CGI data records.
```

```
type ADD_STATUS_TYPE is ( SUCCESSFUL, DATA_RECORD_OVERFLOW ) ;

type REMOVE_STATUS_TYPE is ( SUCCESSFUL,
                             INVALID_PARAMETER_TYPE,
                             OUTPUT_ARRAY_OVERFLOW,
                             NULL_DATA_RECORD,
                             ERROR ) ;

-- Provide a procedure to return the next data record header to the client.

procedure INQ_NEXT_HEADER(
  DATA_RECORD          : in  DATA_RECORD_TYPE ;
  RESPONSE_VALIDITY    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  PARAMETER_COUNT      : out CGI_TYPES.INTEGER_TYPE ;
  PARAMETER_TYPE       : out CGI_TYPES.INDEX_TYPE ) ;

-- Provide a function to see if a data record is empty.  This function returns true if the data record is empty.

function IS_NULL_DATA_RECORD
  ( DATA_RECORD : in DATA_RECORD_TYPE ) return BOOLEAN ;

-- Provide a function which will initialize a data record to a known default state.

procedure INITIALIZE(
  DATA_RECORD : in out DATA_RECORD_TYPE ) ;

-- Provide the add/remove sequence procedures for the data records.

procedure ADD_DATA_RECORD(
  PARAMETER_COUNT     : in     CGI_TYPES.INTEGER_TYPE ;
  DATA_RECORD_ARRAY   : in     DATA_RECORD_ARRAY_TYPE ;
  DATA_RECORD         : in out DATA_RECORD_TYPE ;
  STATUS              :    out ADD_STATUS_TYPE ) ;

procedure ADD_COLOUR_INDEX(
  PARAMETER_COUNT     : in     CGI_TYPES.INTEGER_TYPE ;
  COLOUR_INDEX_ARRAY  : in     CGI_TYPES.COLOUR_INDEX_ARRAY_TYPE ;
  DATA_RECORD         : in out DATA_RECORD_TYPE ;
  STATUS              :    out ADD_STATUS_TYPE ) ;

procedure ADD_COLOUR_DIRECT(
  PARAMETER_COUNT     : in     CGI_TYPES.INTEGER_TYPE ;
  COLOUR_DIRECT_ARRAY : in     CGI_TYPES.DIRECT_COLOUR_ARRAY_TYPE ;
  DATA_RECORD         : in out DATA_RECORD_TYPE ;
  STATUS              :    out ADD_STATUS_TYPE ) ;

procedure ADD_CSN(
  PARAMETER_COUNT : in     CGI_TYPES.INTEGER_TYPE ;
  CSN_ARRAY       : in     CGI_TYPES.CSN_ARRAY_TYPE ;
  DATA_RECORD     : in out DATA_RECORD_TYPE ;
```

```
    STATUS              :    out ADD_STATUS_TYPE ) ;

  procedure ADD_ENUMERATED(
    PARAMETER_COUNT    : in     CGI_TYPES.INTEGER_TYPE ;
    ENUMERATED_ARRAY : in     CGI_TYPES.FIXED_INTEGER_16_ARRAY_TYPE ;
    DATA_RECORD         : in out DATA_RECORD_TYPE ;
    STATUS              :    out ADD_STATUS_TYPE ) ;

  procedure ADD_INTEGER(
    PARAMETER_COUNT : in     CGI_TYPES.INTEGER_TYPE ;
    INTEGER_ARRAY     : in     CGI_TYPES.INTEGER_ARRAY_TYPE ;
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    STATUS            :    out ADD_STATUS_TYPE ) ;

  procedure ADD_FIXED_INTEGER_8(
    PARAMETER_COUNT : in     CGI_TYPES.INTEGER_TYPE ;
    FIXED_INT_8_ARRAY : in     CGI_TYPES.FIXED_INTEGER_8_ARRAY_TYPE ;
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    STATUS            :    out ADD_STATUS_TYPE ) ;

  procedure ADD_FIXED_INTEGER_16(
    PARAMETER_COUNT    : in     CGI_TYPES.INTEGER_TYPE ;
    FIXED_INT_16_ARRAY : in     CGI_TYPES.FIXED_INTEGER_16_ARRAY_TYPE ;
    DATA_RECORD        : in out DATA_RECORD_TYPE ;
    STATUS             :    out ADD_STATUS_TYPE ) ;

  procedure ADD_FIXED_INTEGER_32(
    PARAMETER_COUNT    : in     CGI_TYPES.INTEGER_TYPE ;
    FIXED_INT_32_ARRAY : in     CGI_TYPES.FIXED_INTEGER_32_ARRAY_TYPE ;
    DATA_RECORD        : in out DATA_RECORD_TYPE ;
    STATUS             :    out ADD_STATUS_TYPE ) ;

  procedure ADD_INDEX(
    PARAMETER_COUNT : in     CGI_TYPES.INTEGER_TYPE ;
    INDEX_ARRAY       : in     CGI_TYPES.INDEX_ARRAY_TYPE ;
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    STATUS            :    out ADD_STATUS_TYPE ) ;

  procedure ADD_REAL(
    PARAMETER_COUNT : in     CGI_TYPES.INTEGER_TYPE ;
    REAL_ARRAY        : in     CGI_TYPES.REAL_ARRAY_TYPE ;
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    STATUS            :    out ADD_STATUS_TYPE ) ;

  procedure ADD_STRING(
    PARAMETER_COUNT : in     CGI_TYPES.INTEGER_TYPE ;
    STRING_ARRAY      : in     CGI_TYPES.STRING_ARRAY_TYPE ;
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    STATUS            :    out ADD_STATUS_TYPE ) ;
```

```
procedure ADD_FIXED_STRING(
  PARAMETER_COUNT    : in     CGI_TYPES.INTEGER_TYPE ;
  FIXED_STRING_ARRAY : in     CGI_TYPES.FIXED_STRING_ARRAY_TYPE ;
  DATA_RECORD        : in out DATA_RECORD_TYPE ;
  STATUS             :    out ADD_STATUS_TYPE ) ;

procedure ADD_VDC(
  PARAMETER_COUNT : in     CGI_TYPES.INTEGER_TYPE ;
  VDC_ARRAY       : in     CGI_TYPES.VDC_ARRAY_TYPE ;
  DATA_RECORD     : in out DATA_RECORD_TYPE ;
  STATUS          :    out ADD_STATUS_TYPE ) ;

procedure ADD_VIEWPORT_COORD(
  PARAMETER_COUNT        : in     CGI_TYPES.INTEGER_TYPE ;
  VIEWPORT_COORD_ARRAY : in     CGI_TYPES.VIEWPORT_COORD_ARRAY_TYPE ;
  DATA_RECORD            : in out DATA_RECORD_TYPE ;
  STATUS                 :    out ADD_STATUS_TYPE ) ;

procedure ADD_INPUT_COLOUR(
  PARAMETER_COUNT     : in     CGI_TYPES.INTEGER_TYPE ;
  INPUT_COLOUR_ARRAY : in     CGI_TYPES.INPUT_COLOUR_ARRAY_TYPE ;
  DATA_RECORD         : in out DATA_RECORD_TYPE ;
  STATUS              :    out ADD_STATUS_TYPE ) ;

procedure REMOVE_DATA_RECORD(
  DATA_RECORD         : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT     :    out CGI_TYPES.INTEGER_TYPE ;
  DATA_RECORD_ARRAY :    out DATA_RECORD_ARRAY_TYPE ;
  STATUS              :    out REMOVE_STATUS_TYPE ) ;

procedure REMOVE_COLOUR_INDEX(
  DATA_RECORD         : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT     :    out CGI_TYPES.INTEGER_TYPE ;
  COLOUR_INDEX_ARRAY :    out CGI_TYPES.COLOUR_INDEX_ARRAY_TYPE ;
  STATUS              :    out REMOVE_STATUS_TYPE ) ;

procedure REMOVE_COLOUR_DIRECT(
  DATA_RECORD         : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT     :    out CGI_TYPES.INTEGER_TYPE ;
  COLOUR_DIRECT_ARRAY :    out CGI_TYPES.DIRECT_COLOUR_ARRAY_TYPE ;
  STATUS              :    out REMOVE_STATUS_TYPE ) ;

procedure REMOVE_CSN(
  DATA_RECORD     : in out DATA_RECORD_TYPE ;
  PARAMETER_COUNT :    out CGI_TYPES.INTEGER_TYPE ;
  CSN_ARRAY       :    out CGI_TYPES.CSN_ARRAY_TYPE ;
  STATUS          :    out REMOVE_STATUS_TYPE ) ;

procedure REMOVE_ENUMERATED(
  DATA_RECORD         : in out DATA_RECORD_TYPE ;
```

```
    PARAMETER_COUNT   :   out CGI_TYPES.INTEGER_TYPE ;
    ENUMERATED_ARRAY  :   out CGI_TYPES.FIXED_INTEGER_16_ARRAY_TYPE ;
    STATUS            :   out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_INTEGER(
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    PARAMETER_COUNT   :   out CGI_TYPES.INTEGER_TYPE ;
    INTEGER_ARRAY     :   out CGI_TYPES.INTEGER_ARRAY_TYPE ;
    STATUS            :   out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_FIXED_INTEGER_8(
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    PARAMETER_COUNT   :   out CGI_TYPES.INTEGER_TYPE ;
    FIXED_INT_8_ARRAY :   out CGI_TYPES.FIXED_INTEGER_8_ARRAY_TYPE ;
    STATUS            :   out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_FIXED_INT_16(
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    PARAMETER_COUNT   :   out CGI_TYPES.INTEGER_TYPE ;
    FIXED_INT_16_ARRAY :  out CGI_TYPES.FIXED_INTEGER_16_ARRAY_TYPE ;
    STATUS            :   out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_FIXED_INT_32(
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    PARAMETER_COUNT   :   out CGI_TYPES.INTEGER_TYPE ;
    FIXED_INT_32_ARRAY :  out CGI_TYPES.FIXED_INTEGER_32_ARRAY_TYPE ;
    STATUS            :   out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_INDEX(
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    PARAMETER_COUNT   :   out CGI_TYPES.INTEGER_TYPE ;
    INDEX_ARRAY       :   out CGI_TYPES.INDEX_ARRAY_TYPE ;
    STATUS            :   out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_REAL(
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    PARAMETER_COUNT   :   out CGI_TYPES.INTEGER_TYPE ;
    REAL_ARRAY        :   out CGI_TYPES.REAL_ARRAY_TYPE ;
    STATUS            :   out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_STRING(
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    PARAMETER_COUNT   :   out CGI_TYPES.INTEGER_TYPE ;
    STRING_ARRAY      :   out CGI_TYPES.STRING_ARRAY_TYPE ;
    STATUS            :   out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_FIXED_STRING(
    DATA_RECORD       : in out DATA_RECORD_TYPE ;
    PARAMETER_COUNT   :   out CGI_TYPES.INTEGER_TYPE ;
    FIXED_STRING_ARRAY :  out CGI_TYPES.FIXED_STRING_ARRAY_TYPE ;
```

```
    STATUS                  :    out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_VDC(
    DATA_RECORD      : in out DATA_RECORD_TYPE ;
    PARAMETER_COUNT :    out CGI_TYPES.INTEGER_TYPE ;
    VDC_ARRAY        :    out CGI_TYPES.VDC_ARRAY_TYPE ;
    STATUS           :    out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_VIEWPORT_COORD(
    DATA_RECORD              : in out DATA_RECORD_TYPE ;
    PARAMETER_COUNT          :    out CGI_TYPES.INTEGER_TYPE ;
    VIEWPORT_COORD_ARRAY :    out CGI_TYPES.VIEWPORT_COORD_ARRAY_TYPE ;
    STATUS                   :    out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_INPUT_COLOUR(
    DATA_RECORD          : in out DATA_RECORD_TYPE ;
    PARAMETER_COUNT      :    out CGI_TYPES.INTEGER_TYPE ;
    INPUT_COLOUR_ARRAY :    out CGI_TYPES.INPUT_COLOUR_ARRAY_TYPE ;
    STATUS               :    out REMOVE_STATUS_TYPE ) ;

  procedure REMOVE_EVENT(
    EVENT_REPORTS_LIST : in out EVENT_REPORTS_LIST_TYPE ;
    EVENT              :    out EVENT_TYPE ;
    STATUS             :    out REMOVE_STATUS_TYPE ) ;

  -- Deallocation procedures.

  procedure DEALLOCATE_EVENT_REPORT_LIST(
    EVENT_REPORT_LIST : in out ACCESS_DATA_RECORD_TYPE ) ;

  procedure DEALLOCATE_EVENT(
    EVENT : in out EVENT_TYPE ) ;

  -- Private declaration of the CGI data record.

  private

    type DATA_RECORD_TYPE
      ( LENGTH : DATA_RECORD_LENGTH_TYPE := CGI_CONFIG.MAX_DATA_RECORD_SIZE ) is
      record
        CHARS_IN_STRING : DATA_RECORD_LENGTH_TYPE := 0 ;
        STRING_DATA     : CGI_TYPES.OCTET_ARRAY_TYPE( 1..LENGTH ) ;
      end record ;

end CGI_DATA_RECORD_UTILS ;
```

A.4 Package specification CGI

```
with CGI_TYPES ;
with CGI_DATA_RECORD_UTILS ;

package CGI is

   -- Package: CGI
   -- Functional Description:
   --
   -- This package provides the specifications for all of the functions denoted in the ISO/IEC 9636.

                              ------------------- CGI Control Functions -------------------

   procedure INITIALIZE_SESSION ;

   procedure TERMINATE_SESSION ;

   procedure EXECUTE_DEFERRED_ACTIONS ;

   procedure SET_DEFERRAL_MODE(
     DEFERRAL_MODE : in CGI_TYPES.DEFERRAL_MODE_TYPE ) ;

   procedure PREPARE_DRAWING_SURFACE(
     CLEAR_DRAWING_SURFACE : in CGI_TYPES.CLEAR_DRAWING_SURFACE_TYPE ) ;

   procedure END_PAGE ;

   procedure SET_VDC_TYPE(
     VDC_TYPE_SELECTOR : CGI_TYPES.VDC_SELECTION_TYPE ) ;

   procedure SET_VDC_INTEGER_PREC_REQUIREMENT(
     LOG_2_MAGNITUDE_OF_UPPER_BOUND : in CGI_TYPES.INTEGER_TYPE ) ;

   procedure SET_VDC_REAL_PREC_REQUIREMENTS(
     LOG_2_MAGNITUDE_OF_UPPER_BOUND          : in CGI_TYPES.INTEGER_TYPE ;
     LOG_2_NON_ZERO_MAGNITUDE_MIN            : in CGI_TYPES.INTEGER_TYPE ;
     LOG_2_RELATIVE_PREC_REQUIREMENT         : in CGI_TYPES.INTEGER_TYPE ;
     LOG_2_MIN_MAGNITUDE_FOR_RELATIVE_PREC : in CGI_TYPES.INTEGER_TYPE ;
     LOG_2_TYPICAL_MAGNITUDE                 : in CGI_TYPES.INTEGER_TYPE ) ;

   procedure SET_VDC_EXTENT(
     VDC_EXTENT : in CGI_TYPES.VDC_EXTENT_TYPE ) ;

   procedure SET_DEVICE_VIEWPORT(
     VIEWPORT : in CGI_TYPES.DEVICE_VIEWPORT_TYPE ) ;

   procedure SET_DEVICE_VIEWPORT_SPECIF_MODE(
```

```
    VC_MODE                  : in CGI_TYPES.VIEWPORT_SPECIF_MODE_TYPE ;
    METRIC_SCALE_FACTOR : in CGI_TYPES.REAL_TYPE ) ;

  procedure SET_DEVICE_VIEWPORT_MAPPING(
    ISOTROPY_FLAG          : in CGI_TYPES.ISOTROPY_FLAG_TYPE ;
    HOR_ALIGNMENT_FLAG  : in CGI_TYPES.HOR_ALIGNMENT_FLAG_TYPE ;
    VERT_ALIGNMENT_FLAG : in CGI_TYPES.VERT_ALIGNMENT_FLAG_TYPE ) ;

  procedure SET_DRAWING_SURFACE_CLIP_RECTANGLE(
    DSCRECT : in CGI_TYPES.DSCRECT_TYPE ) ;

  procedure SET_DRAWING_SURFACE_CLIP_INDICATOR(
    DRAWING_SURFACE_CLIP_INDICATOR : in CGI_TYPES.
                                          DRAWING_SURFACE_CLIP_INDICATOR_TYPE ) ;

  procedure DEQUEUE_ERROR_REPORTS(
    NUM_OF_REPORTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY          : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    REPORTS_REMAINING_IN_QUEUE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_ERROR_REPORTS      : out CGI_TYPES.ERROR_QUEUE_TYPE ) ;

  procedure SET_ERROR_HANDLING_CONTROL(
    ERROR_CLASS_HANDLING_PAIRS : in CGI_TYPES.ERROR_HANDLING_ARRAY_TYPE ) ;

  procedure SET_INTEGER_PREC_REQUIREMENT(
    LOG_2_MAGNITUDE_OF_UPPER_BOUND : in CGI_TYPES.INTEGER_TYPE ) ;

  procedure SET_REAL_PREC_REQUIREMENTS(
    LOG_2_MAGNITUDE_OF_UPPER_BOUND      : in CGI_TYPES.INTEGER_TYPE ;
    LOG_2_NON_ZERO_MAGNITUDE_MIN        : in CGI_TYPES.INTEGER_TYPE ;
    LOG_2_RELATIVE_PREC_REQUIREMENT     : in CGI_TYPES.INTEGER_TYPE ;
    LOG_2_MIN_MAGNITUDE_FOR_RELATIVE_PREC : in CGI_TYPES.INTEGER_TYPE ;
    LOG_2_TYPICAL_MAGNITUDE             : in CGI_TYPES.INTEGER_TYPE ) ;

  procedure SET_INDEX_PREC_REQUIREMENT(
    LOG_2_MAGNITUDE_OF_UPPER_BOUND : in CGI_TYPES.INTEGER_TYPE ) ;

  procedure SET_COLOUR_PREC_REQUIREMENT(
    LOG_2_MAGNITUDE_OF_UPPER_BOUND : in CGI_TYPES.INTEGER_TYPE ) ;

  procedure SET_COLOUR_INDEX_PREC_REQUIREMENT(
    LOG_2_MAGNITUDE_OF_UPPER_BOUND : in CGI_TYPES.INTEGER_TYPE ) ;

  procedure SET_CSN_PREC_REQUIREMENT(
    LOG_2_MAGNITUDE_OF_UPPER_BOUND : in CGI_TYPES.INTEGER_TYPE ) ;

  procedure MESSAGE(
    ACTION_REQUIRED_FLAG : in CGI_TYPES.ACTION_REQUIRED_FLAG_TYPE ;
    MESSAGE_TEXT         : in CGI_TYPES.STRING_TYPE ) ;
```

```
procedure ESCAPE(
  ESCAPE_FUNCTION_ID : in CGI_TYPES.INTEGER_TYPE ;
  DATA_RECORD        : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

procedure GET_ESCAPE(
  ESCAPE_FUNCTION_ID   : in  CGI_TYPES.INTEGER_TYPE ;
  DATA_RECORD          : in  CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ;
  RESPONSE_VALIDITY    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  OUTPUT_DATA_RECORD : out CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;


procedure SET_STATE_LIST_INQUIRY_SOURCE(
  SOURCE : in CGI_TYPES.INQUIRY_SOURCE_FLAG_TYPE ) ;

procedure INQ_DEVICE_ID(
  MAX_CHARS_PER_STRING : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  DEVICE_CLASS         : out CGI_TYPES.DEVICE_CLASS_TYPE ;
  DEVICE_IDENTIFICATION : out CGI_TYPES.STRING_TYPE ) ;

procedure INQ_DEVICE_DESCRIPTION(
  RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  HARD_SOFT_COPY_CLASS                 : out CGI_TYPES.HARD_SOFT_COPY_TYPE ;
  DISPLAY_TYPE                         : out CGI_TYPES.DISPLAY_TYPE ;
  DYN_MOD_ACCEPTED_FOR_VDC_TO_DEV_MAP : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
  SPONTANEOUS_DISPLAY_CHANGE_POSSIBLE  : out CGI_TYPES.YES_NO_FLAG_TYPE ;
  DISPLAY_SURFACE_BOTTOM_LEFT          : out CGI_TYPES.DEVICE_POINT_TYPE ;
  DISPLAY_SURFACE_UPPER_RIGHT          : out CGI_TYPES.DEVICE_POINT_TYPE ;
  DISPLAY_SURFACE_WIDTH                : out CGI_TYPES.REAL_TYPE ;
  DISPLAY_SURFACE_HEIGHT               : out CGI_TYPES.REAL_TYPE ;
  PIXEL_LOCATION_RELATIVE_TO_COORDS    : out CGI_TYPES.
                                           PIXEL_RELATIVE_LOCATION_TYPE ) ;

procedure LOOKUP_FUNCTION_SUPPORT(
  LIST_OF_FUNCTION_IDS      : in  CGI_TYPES.FUNCTION_ID_ARRAY_TYPE ;
  RESPONSE_VALIDITY         : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  LIST_OF_SUPPORT_INDICATORS : out CGI_TYPES.YES_NO_FLAG_ARRAY_TYPE ) ;

procedure LOOKUP_PROFILE_SUPPORT(
  LIST_OF_PROFILE_IDS       : in  CGI_TYPES.PROFILE_ID_ARRAY_TYPE ;
  RESPONSE_VALIDITY         : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  LIST_OF_SUPPORT_INDICATORS : out CGI_TYPES.PROFILE_SUPPORT_FLAG_ARRAY_TYPE ) ;

procedure INQ_PROFILE_SUPPORT_INDICATORS(
  NUM_OF_LIST_ELEMENTS_REQUESTED      : in  CGI_TYPES.INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY                   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_PROFILE_SUPPORT_INDICATORS  : out CGI_TYPES.
                                          PROFILE_SUPPORT_ARRAY_TYPE ) ;
```

```
procedure INQ_SUPPORTED_VDC_TYPES(
  RESPONSE_VALIDITY : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  VDC_TYPE_SUPPORT  : out CGI_TYPES.VDC_SUPPORT_TYPE ) ;

procedure INQ_DEVICE_CONTROL_CAP(
  RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  VIEWPORT_SPECIF_MODE_ELEMENTS        : out CGI_TYPES.INTEGER_TYPE ;
  VIEWPORT_SPECIF_MODE_ARRAY           : out CGI_TYPES.
                                           VIEWPORT_SPECIF_MODE_ARRAY_TYPE ;
  MESSAGE_ACTION_DETECTION_SUPPORT     : out CGI_TYPES.YES_NO_FLAG_TYPE ;
  MAX_NUM_OF_CHARS_FOR_MESSAGE         : out CGI_TYPES.INTEGER_TYPE ;
  DEVICE_VIEWPORT_MIRRORING_SUPPORT    : out CGI_TYPES.YES_NO_FLAG_TYPE ;
  DEFERRAL_MODE_ELEMENTS               : out CGI_TYPES.INTEGER_TYPE ;
  DEFERRAL_MODE_ARRAY                  : out CGI_TYPES.DEFERRAL_MODE_ARRAY_TYPE ;
  SIZE_OF_ERROR_QUEUE                  : out CGI_TYPES.INTEGER_TYPE ) ;

procedure LOOKUP_ESCAPE_SUPPORT(
  LIST_OF_ESCAPE_FUNCTION_IDS : in  CGI_TYPES.INTEGER_ARRAY_TYPE ;
  RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  LIST_OF_SUPPORT_INDICATORS  : out CGI_TYPES.YES_NO_FLAG_ARRAY_TYPE ) ;

procedure LOOKUP_GET_ESCAPE_SUPPORT(
  LIST_OF_GET_ESCAPE_FUNCTION_IDS : in  CGI_TYPES.INTEGER_ARRAY_TYPE ;
  RESPONSE_VALIDITY               : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  LIST_OF_SUPPORT_INDICATORS      : out CGI_TYPES.YES_NO_FLAG_ARRAY_TYPE ) ;

procedure INQ_CONTROL_STATE(
  RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  DEVICE_DRAWING_SURFACE_STATE     : out CGI_TYPES.DRAWING_SURFACE_STATE_TYPE ;
  DEFERRAL_MODE                    : out CGI_TYPES.DEFERRAL_MODE_TYPE ;
  VDC_TYPE                         : out CGI_TYPES.VDC_MODE_TYPE ;
  VIEWPORT_SPECIF_MODE             : out CGI_TYPES.VIEWPORT_SPECIF_MODE_TYPE ;
  VIEWPORT_METRIC_SCALE_FACTOR     : out CGI_TYPES.REAL_TYPE ;
  NUM_OF_QUEUED_ERROR_REPORTS      : out CGI_TYPES.INTEGER_TYPE ) ;

procedure INQ_CURRENT_PREC_REQUIREMENTS(
  RESPONSE_VALIDITY                                    : out CGI_TYPES.
                                                           VALIDITY_FLAG_TYPE ;
  VDC_INTEGER_LOG2_MAGNITUDE_OF_UPPER_BOUND            : out CGI_TYPES.INTEGER_TYPE ;
  VDC_REAL_LOG2_MAGNITUDE_OF_UPPER_BOUND               : out CGI_TYPES.INTEGER_TYPE ;
  VDC_REAL_LOG2_NON_ZERO_MAGNITUDE_MIN                 : out CGI_TYPES.INTEGER_TYPE ;
  VDC_REAL_LOG2_RELATIVE_PREC_REQUIREMENT              : out CGI_TYPES.INTEGER_TYPE ;
  VDC_REAL_LOG2_MIN_MAGNITUDE_FOR_RELATIVE_PREC        : out CGI_TYPES.INTEGER_TYPE ;
  VDC_REAL_LOG2_TYPICAL_MAGNITUDE                      : out CGI_TYPES.INTEGER_TYPE ;
  INTEGER_LOG2_MAGNITUDE_OF_UPPER_BOUND                : out CGI_TYPES.INTEGER_TYPE ;
  REAL_LOG2_MAGNITUDE_OF_UPPER_BOUND                   : out CGI_TYPES.INTEGER_TYPE ;
  REAL_LOG2_NON_ZERO_MAGNITUDE_MIN                     : out CGI_TYPES.INTEGER_TYPE ;
  REAL_LOG2_RELATIVE_PREC_REQUIREMENT                  : out CGI_TYPES.INTEGER_TYPE ;
  REAL_LOG2_MIN_MAGNITUDE_FOR_RELATIVE_PREC            : out CGI_TYPES.INTEGER_TYPE ;
  REAL_LOG2_TYPICAL_MAGNITUDE                          : out CGI_TYPES.INTEGER_TYPE ;
```

```
    INDEX_LOG2_MAGNITUDE_OF_UPPER_BOUND          : out CGI_TYPES.INTEGER_TYPE ;
    COLOUR_LOG2_MAGNITUDE_OF_UPPER_BOUND         : out CGI_TYPES.INTEGER_TYPE ;
    COLOUR_INDEX_LOG2_MAGNITUDE_OF_UPPER_BOUND   : out CGI_TYPES.INTEGER_TYPE ;
    CSN_LOG2_MAGNITUDE_OF_UPPER_BOUND            : out CGI_TYPES.INTEGER_TYPE
  ) ;

  procedure INQ_MISCELLANEOUS_CONTROL_STATE(
    RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    STATE_LIST_INQUIRY_SOURCE : out CGI_TYPES.INQUIRY_SOURCE_FLAG_TYPE ) ;

  procedure INQ_VDC_TO_DEVICE_MAPPING(
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE;
    VDC_EXTENT                       : out CGI_TYPES.VDC_EXTENT_TYPE ;
    ISOTROPY                         : out CGI_TYPES.ISOTROPY_FLAG_TYPE ;
    HOR_ALIGNMENT                    : out CGI_TYPES.HOR_ALIGNMENT_FLAG_TYPE ;
    VERT_ALIGNMENT                   : out CGI_TYPES.VERT_ALIGNMENT_FLAG_TYPE ;
    VIEWPORT_SPECIF_MODE             : out CGI_TYPES.VIEWPORT_SPECIF_MODE_TYPE ;
    VIEWPORT_METRIC_SCALE_FACTOR : out CGI_TYPES.REAL_TYPE ;
    REQUESTED_DEVICE_VIEWPORT        : out CGI_TYPES.DEVICE_VIEWPORT_TYPE ;
    EFFECTIVE_VIEWPORT               : out CGI_TYPES.DEVICE_VIEWPORT_TYPE ;
    SURFACE_CLIP_INDICATOR           : out CGI_TYPES.
                                         DRAWING_SURFACE_CLIP_INDICATOR_TYPE;
    DSCRECT_SPECIF_MODE              : out CGI_TYPES.VIEWPORT_SPECIF_MODE_TYPE ;
    DSCRECT_METRIC_SCALE_FACTOR  : out CGI_TYPES.REAL_TYPE ;
    DSCRECT                          : out CGI_TYPES.DEVICE_VIEWPORT_TYPE ) ;

  procedure INQ_ERROR_HANDLING(
    RESPONSE_VALIDITY       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    ERROR_HANDLING_FLAGS : out CGI_TYPES.ERROR_HANDLING_FLAG_ARRAY_TYPE ) ;

                        ---------------------- CGI Output Functions ----------------------

  procedure POLYLINE(
    POINT_LIST  : in CGI_TYPES.POINT_LIST_TYPE ;
    FINAL_FLAG : in CGI_TYPES.FINAL_FLAG_TYPE := CGI_TYPES.FINAL) ;

  procedure DISJOINT_POLYLINE(
    POINT_LIST : in CGI_TYPES.POINT_LIST_TYPE ) ;

  procedure CIRCULAR_ARC_3_POINT(
    STARTING_POINT       : in CGI_TYPES.VDC_POINT_TYPE ;
    INTERMEDIATE_POINT : in CGI_TYPES.VDC_POINT_TYPE ;
    ENDING_POINT         : in CGI_TYPES.VDC_POINT_TYPE ) ;

  procedure CIRCULAR_ARC_CENTRE(
    CENTRE_POINT        : in CGI_TYPES.VDC_POINT_TYPE ;
    ARC_OFFSET_POINTS : in CGI_TYPES.ARC_OFFSET_POINT_TYPE ;
    RADIUS                  : in CGI_TYPES.VDC_TYPE ) ;

  procedure CIRCULAR_ARC_CENTRE_REVERSED(
```

```
    CENTRE_POINT        : in CGI_TYPES.VDC_POINT_TYPE ;
    ARC_OFFSET_POINTS   : in CGI_TYPES.ARC_OFFSET_POINT_TYPE ;
    RADIUS              : in CGI_TYPES.VDC_TYPE ) ;

  procedure ELLIPTICAL_ARC(
    CENTRE_POINT                          : in CGI_TYPES.VDC_POINT_TYPE ;
    FIRST_CONJUGATE_RADIUS_ENDPOINT       : in CGI_TYPES.VDC_POINT_TYPE ;
    SECOND_CONJUGATE_RADIUS_ENDPOINT      : in CGI_TYPES.VDC_POINT_TYPE ;
    ARC_OFFSET_POINTS                     : in CGI_TYPES.ARC_OFFSET_POINT_TYPE ) ;

  procedure CONNECTING_EDGE ;

  procedure POLYMARKER(
    POINT_LIST : in CGI_TYPES.POINT_LIST_TYPE ) ;

  procedure TEXT(
    POINT               : in CGI_TYPES.VDC_POINT_TYPE ;
    TEXT_FINAL_FLAG     : in CGI_TYPES.FINAL_FLAG_TYPE ;
    TEXT_STRING         : in CGI_TYPES.STRING_TYPE ) ;

  procedure RESTRICTED_TEXT(
    EXTENT              : in CGI_TYPES.EXTENT_TYPE ;
    POINT               : in CGI_TYPES.VDC_POINT_TYPE ;
    TEXT_FINAL_FLAG     : in CGI_TYPES.FINAL_FLAG_TYPE ;
    TEXT_STRING         : in CGI_TYPES.STRING_TYPE ) ;

  procedure APPEND_TEXT(
    TEXT_FINAL_FLAG     : in CGI_TYPES.FINAL_FLAG_TYPE ;
    TEXT_STRING         : in CGI_TYPES.STRING_TYPE ) ;

  procedure POLYGON(
    POINT_LIST   : in CGI_TYPES.POINT_LIST_TYPE ;
    FINAL_FLAG : in CGI_TYPES.FINAL_FLAG_TYPE := CGI_TYPES.FINAL ) ;

  procedure POLYGON_SET(
    FLAGGED_POINT_LIST : in CGI_TYPES.FLAGGED_POINT_LIST_TYPE ) ;

  procedure RECTANGLE(
    CORNER_POINTS : in CGI_TYPES.RECTANGLE_DESCRIPTOR_TYPE ) ;

  procedure CIRCLE(
    CENTRE_POINT : in CGI_TYPES.VDC_POINT_TYPE ;
    RADIUS       : in CGI_TYPES.VDC_TYPE ) ;

  procedure CIRCULAR_ARC_3_POINT_CLOSE(
    STARTING_POINT      : in CGI_TYPES.VDC_POINT_TYPE ;
    INTERMEDIATE_POINT  : in CGI_TYPES.VDC_POINT_TYPE ;
    ENDING_POINT        : in CGI_TYPES.VDC_POINT_TYPE ;
    CLOSE_TYPE          : in CGI_TYPES.CLOSE_TYPE ) ;
```

```
procedure CIRCULAR_ARC_CENTRE_CLOSE(
  CENTRE_POINT        : in CGI_TYPES.VDC_POINT_TYPE ;
  ARC_OFFSET_POINTS : in CGI_TYPES.ARC_OFFSET_POINT_TYPE ;
  RADIUS              : in CGI_TYPES.VDC_TYPE ;
  CLOSE_TYPE          : in CGI_TYPES.CLOSE_TYPE ) ;

procedure ELLIPSE(
  CENTRE_POINT                              : in CGI_TYPES.VDC_POINT_TYPE ;
  FIRST_CONJUGATE_RADIUS_ENDPOINT     : in CGI_TYPES.VDC_POINT_TYPE ;
  SECOND_CONJUGATE_RADIUS_ENDPOINT : in CGI_TYPES.VDC_POINT_TYPE ) ;

procedure ELLIPTICAL_ARC_CLOSE(
  CENTRE_POINT                              : in CGI_TYPES.VDC_POINT_TYPE ;
  FIRST_CONJUGATE_RADIUS_ENDPOINT     : in CGI_TYPES.VDC_POINT_TYPE ;
  SECOND_CONJUGATE_RADIUS_ENDPOINT : in CGI_TYPES.VDC_POINT_TYPE ;
  ARC_OFFSET_POINTS                         : in CGI_TYPES.ARC_OFFSET_POINT_TYPE ;
  CLOSE_TYPE                                : in CGI_TYPES.CLOSE_TYPE ) ;

procedure CELL_ARRAY(
  CELL_ARRAY_PARALLELOGRAM       : in CGI_TYPES.
                                     CELL_ARRAY_PARALLELOGRAM_TYPE ;
  DIMENSIONS                           : in CGI_TYPES.CELL_ARRAY_DIMENSION_TYPE ;
  LOCAL_COLOUR_PREC_REQUIREMENT : in CGI_TYPES.INTEGER_TYPE ;
  CELL_COLOUR_SPECIFIERS         : in CGI_TYPES.COLOUR_ARRAY_TYPE ;
  FINAL_FLAG : in CGI_TYPES.FINAL_FLAG_TYPE := CGI_TYPES.FINAL ) ;


procedure GENERALIZED_DRAWING_PRIMITIVE(
  GDP_ID          : in CGI_TYPES.INTEGER_TYPE ;
  POINT_LIST      : in CGI_TYPES.POINT_LIST_TYPE ;
  DATA_RECORD : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ;
  FINAL_FLAG      : in CGI_TYPES.FINAL_FLAG_TYPE := CGI_TYPES.FINAL ) ;

procedure SET_LINE_BUNDLE_INDEX(
  LINE_BUNDLE_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_LINE_TYPE(
  LINE_TYPE : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_LINE_WIDTH(
  LINE_WIDTH_SPECIFIER : in CGI_TYPES.SIZE_SPECIF_TYPE ) ;

procedure SET_LINE_COLOUR(
  LINE_COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

procedure SET_LINE_CLIPPING_MODE(
  CLIPPING_MODE : in CGI_TYPES.CLIP_MODE_TYPE ) ;

procedure SET_MARKER_BUNDLE_INDEX(
  MARKER_BUNDLE_INDEX : in CGI_TYPES.INDEX_TYPE ) ;
```

```
procedure SET_MARKER_TYPE(
  MARKER_TYPE : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_MARKER_SIZE(
  MARKER_SIZE_SPECIFIER : in CGI_TYPES.SIZE_SPECIF_TYPE ) ;

procedure SET_MARKER_COLOUR(
  MARKER_COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

procedure SET_MARKER_CLIPPING_MODE(
  CLIPPING_MODE : in CGI_TYPES.CLIP_MODE_TYPE ) ;

procedure SET_TEXT_BUNDLE_INDEX(
  TEXT_BUNDLE_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_TEXT_FONT_INDEX(
  FONT_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_TEXT_PREC(
  TEXT_PREC : in CGI_TYPES.TEXT_PREC_TYPE ) ;

procedure SET_CHAR_EXPAN_FACTOR(
  CHAR_EXPAN_FACTOR : in CGI_TYPES.REAL_TYPE ) ;

procedure SET_CHAR_SPACING(
  CHAR_SPACING : in CGI_TYPES.REAL_TYPE ) ;

procedure SET_TEXT_COLOUR(
  TEXT_COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;


procedure SET_CHAR_HEIGHT(
  CHAR_HEIGHT : in CGI_TYPES.VDC_TYPE ) ;

procedure SET_CHAR_ORIENTATION(
  CHAR_UP_VECTOR   : in CGI_TYPES.CHAR_VECTOR_TYPE ;
  CHAR_BASE_VECTOR : in CGI_TYPES.CHAR_VECTOR_TYPE ) ;

procedure SET_TEXT_PATH(
  TEXT_PATH : in CGI_TYPES.TEXT_PATH_TYPE ) ;

procedure SET_TEXT_ALIGNMENT(
  HOR_ALIGNMENT                : in CGI_TYPES.HOR_ALIGNMENT_TYPE ;
  VERT_ALIGNMENT               : in CGI_TYPES.VERT_ALIGNMENT_TYPE ;
  CONTINUOUS_HOR_ALIGNMENT     : in CGI_TYPES.REAL_TYPE ;
  CONTINUOUS_VERT_ALIGNMENT    : in CGI_TYPES.REAL_TYPE ) ;

procedure SET_CHAR_SET_INDEX(
  CHAR_SET_INDEX : in CGI_TYPES.INDEX_TYPE ) ;
```

```
procedure SET_ALTERNATE_CHAR_SET_INDEX(
  ALTERNATE_CHAR_SET_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_CHAR_CODING_ANNOUNCER(
  CODING_TECHNIQUE : in CGI_TYPES.CODING_TECHNIQUE_TYPE ) ;

procedure SET_FILL_BUNDLE_INDEX(
  FILL_BUNDLE_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_INTERIOR_STYLE(
  INTERIOR_STYLE : in CGI_TYPES.INTERIOR_STYLE_TYPE ) ;

procedure SET_FILL_COLOUR(
  FILL_COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

procedure SET_HATCH_INDEX(
  HATCH_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_PATTERN_INDEX(
  PATTERN_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_FILL_REFERENCE_POINT(
  REFERENCE_POINT : in CGI_TYPES.VDC_POINT_TYPE ) ;

procedure SET_PATTERN_SIZE(
  PATTERN_HEIGHT_VECTOR : in CGI_TYPES.PATTERN_VECTOR_TYPE ;
  PATTERN_WIDTH_VECTOR  : in CGI_TYPES.PATTERN_VECTOR_TYPE ) ;

procedure SET_EDGE_BUNDLE_INDEX(
  EDGE_BUNDLE_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_EDGE_TYPE(
  EDGE_TYPE_INDICATOR : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_EDGE_WIDTH(
  EDGE_WIDTH_SPECIFIER : in CGI_TYPES.SIZE_SPECIF_TYPE ) ;

procedure SET_EDGE_COLOUR(
  EDGE_COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

procedure SET_EDGE_CLIPPING_MODE(
  CLIPPING_MODE : in CGI_TYPES.CLIP_MODE_TYPE ) ;

procedure SET_EDGE_VISIBILITY(
  EDGE_VISIBILITY : in CGI_TYPES.EDGE_VISIBILITY_TYPE ) ;

procedure SET_CLIP_INDICATOR(
  CLIP_INDICATOR : in CGI_TYPES.ON_OFF_FLAG_TYPE ) ;

procedure SET_CLIP_RECTANGLE(
```

```
    CLIP_RECTANGLE : in CGI_TYPES.CLIP_RECTANGLE_TYPE ) ;

  procedure SET_LINE_WIDTH_SPECIF_MODE(
    LINE_WIDTH_SPECIF_MODE : in CGI_TYPES.SPECIF_MODE_TYPE ) ;

  procedure SET_EDGE_WIDTH_SPECIF_MODE(
    EDGE_WIDTH_SPECIF_MODE : in CGI_TYPES.SPECIF_MODE_TYPE ) ;

  procedure SET_MARKER_SIZE_SPECIF_MODE(
    MARKER_SIZE_SPECIF_MODE : in CGI_TYPES.SPECIF_MODE_TYPE ) ;

  procedure SET_COLOUR_SELECTION_MODE(
    COLOUR_SELECTION_MODE : in CGI_TYPES.COLOUR_SELECTION_MODE_TYPE ) ;

  procedure SET_COLOUR_VALUE_EXTENT(
    MIN_COLOUR_VALUES  : in CGI_TYPES.DIRECT_COLOUR_VALUE_TYPE ;
    MAX_COLOUR_VALUES : in CGI_TYPES.DIRECT_COLOUR_VALUE_TYPE ) ;

  procedure SET_BACKGROUND_COLOUR(
    COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

  procedure SET_AUXILIARY_COLOUR(
    AUXILIARY_COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

  procedure SET_TRANSPARENCY(
    TRANSPARENCY_SWITCH : in CGI_TYPES.TRANSPARENCY_TYPE ) ;

  procedure SET_COLOUR_TABLE(
    STARTING_INDEX : in CGI_TYPES.COLOUR_INDEX_TYPE ;
    COLOUR_LIST      : in CGI_TYPES.COLOUR_TABLE_TYPE ) ;

  procedure SET_LINE_REP(
    LINE_BUNDLE_INDEX      : in CGI_TYPES.INDEX_TYPE ;
    LINE_TYPE_INDICATOR    : in CGI_TYPES.INDEX_TYPE ;
    LINE_WIDTH_SPECIFIER   : in CGI_TYPES.SIZE_SPECIF_TYPE ;
    LINE_COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

  procedure SET_MARKER_REP(
    MARKER_BUNDLE_INDEX      : in CGI_TYPES.INDEX_TYPE ;
    MARKER_TYPE_INDICATOR    : in CGI_TYPES.INDEX_TYPE ;
    MARKER_SIZE_SPECIFIER    : in CGI_TYPES.SIZE_SPECIF_TYPE ;
    MARKER_COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

  procedure SET_TEXT_REP(
    TEXT_BUNDLE_INDEX      : in CGI_TYPES.INDEX_TYPE ;
    TEXT_FONT_INDEX        : in CGI_TYPES.INDEX_TYPE ;
    TEXT_PREC              : in CGI_TYPES.TEXT_PREC_TYPE ;
    CHAR_SPACING           : in CGI_TYPES.REAL_TYPE ;
    CHAR_EXPAN_FACTOR      : in CGI_TYPES.REAL_TYPE ;
    TEXT_COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;
```

```
procedure SET_FILL_REP(
  FILL_BUNDLE_INDEX      : in CGI_TYPES.INDEX_TYPE ;
  INTERIOR_STYLE         : in CGI_TYPES.INTERIOR_STYLE_TYPE ;
  FILL_COLOUR_SPECIFIER  : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ;
  HATCH_INDEX            : in CGI_TYPES.INDEX_TYPE ;
  PATTERN_INDEX          : in CGI_TYPES.INDEX_TYPE ;
  FILL_BITMAP_ID         : in CGI_TYPES.BITMAP_ID_TYPE ;
  FILL_BITMAP_REGION     : in CGI_TYPES.BITMAP_REGION_TYPE ) ;

procedure SET_EDGE_REP(
  EDGE_BUNDLE_INDEX      : in CGI_TYPES.INDEX_TYPE ;
  EDGE_TYPE_INDICATOR    : in CGI_TYPES.INDEX_TYPE ;
  EDGE_WIDTH_SPECIFIER   : in CGI_TYPES.SIZE_SPECIF_TYPE ;
  EDGE_COLOUR_SPECIFIER  : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ;
  EDGE_VISIBILITY        : in CGI_TYPES.EDGE_VISIBILITY_TYPE ) ;

procedure DELETE_BUNDLE_REP(
  BUNDLE_TABLE_TYPE : in CGI_TYPES.BUNDLE_TABLE_TYPE ;
  BUNDLE_INDEX      : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_ASPECT_SOURCE_FLAGS(
  LIST_OF_ASF_TYPE_VALUE_PAIRS : in CGI_TYPES.ASF_RECORD_ARRAY_TYPE ) ;

procedure SET_PATTERN_TABLE(
  PATTERN_TABLE_INDEX                : in CGI_TYPES.INDEX_TYPE ;
  PATTERN_DIMENSIONS                 : in CGI_TYPES.PATTERN_DIMENSION_TYPE ;
  LOCAL_COLOUR_PREC_REQUIREMENT      : in CGI_TYPES.INTEGER_TYPE ;
  PATTERN_COLOUR_SPECIFIERS          : in CGI_TYPES.COLOUR_ARRAY_TYPE ) ;

procedure DELETE_PATTERN(
  PATTERN_TABLE_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_FONT_LIST(
  FONT_NAMES : in CGI_TYPES.FIXED_STRING_ARRAY_TYPE ) ;

procedure SET_CHAR_SET_LIST(
  LIST_OF_CHAR_SETS : in CGI_TYPES.CHAR_SET_ARRAY_TYPE ) ;

procedure SAVE_PRIMITIVE_ATTRIBUTES(
  ATTRIBUTE_SET_NAME : in CGI_TYPES.ASN_TYPE ) ;

procedure RESTORE_PRIMITIVE_ATTRIBUTES(
  ATTRIBUTE_SET_NAME : in CGI_TYPES.ASN_TYPE ) ;

procedure DELETE_PRIMITIVE_ATTRIBUTE_SAVE_SET(
  ATTRIBUTE_SET_NAME : in CGI_TYPES.ASN_TYPE ) ;

procedure BEGIN_FIGURE ;

procedure END_FIGURE ;
```

```
    procedure NEW_REGION ;

    procedure GET_TEXT_EXTENT(
      TEXT_POSITION                : in  CGI_TYPES.VDC_POINT_TYPE ;
      CHAR_STRING                  : in  CGI_TYPES.STRING_TYPE ;
      RESPONSE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
      CONCATENATION_VALIDITY       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
      CONCATENATION_POINT          : out CGI_TYPES.VDC_POINT_TYPE ;
      TEXT_EXTENT_PARALLELOGRAM    : out CGI_TYPES.
                                         TEXT_EXTENT_PARALLELOGRAM_TYPE ) ;

    procedure INQ_PRIMITIVE_SUPPORT_LEVELS(
      RESPONSE_VALIDITY                 : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
      MAX_NUM_POLYLINE_POINTS           : out CGI_TYPES.INTEGER_TYPE ;
      MAX_NUM_DISJOINT_POLYLINE_POINTS  : out CGI_TYPES.INTEGER_TYPE ;
      MAX_NUM_POLYGON_POINTS            : out CGI_TYPES.INTEGER_TYPE ;
      MAX_NUM_POLYGON_SET_POINTS        : out CGI_TYPES.INTEGER_TYPE ;
      MAX_NUM_POLYMARKER_POINTS         : out CGI_TYPES.INTEGER_TYPE ;
      MAX_NUM_CHARS_FOR_TEXT            : out CGI_TYPES.INTEGER_TYPE ;
      MAX_NUM_CELL_ARRAY_CELL_COLOURS   : out CGI_TYPES.INTEGER_TYPE ;
      LINE_EDGE_CONTINUITY_CAPABILITY   : out CGI_TYPES.
                                              LINE_EDGE_CONTINUITY_CAP_TYPE ;
      CELL_ARRAY_FILL_CAPABILITY        : out CGI_TYPES.CELL_ARRAY_FILL_CAP_TYPE ;
      CELL_ARRAY_ALIGNMENT_CAPABILITY   : out CGI_TYPES.
                                              CELL_ARRAY_ALIGNMENT_CAP_TYPE ;
      CELL_ARRAY_RENDERING_TECHNIQUE    : out CGI_TYPES.CELL_ARRAY_TECHNIQUE_TYPE ;
      COMPOUND_TEXT_CAPABILITY          : out CGI_TYPES.COMPOUND_OBJECT_CAP_TYPE ;
      CLOSED_FIGURE_CAPABILITY          : out CGI_TYPES.COMPOUND_OBJECT_CAP_TYPE
    ) ;

    procedure LOOKUP_GDP_SUPPORT(
      LIST_OF_GDP_IDS              : in  CGI_TYPES.INTEGER_ARRAY_TYPE ;
      RESPONSE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
      LIST_OF_SUPPORT_INDICATORS   : out CGI_TYPES.YES_NO_FLAG_ARRAY_TYPE ) ;

    procedure INQ_GDP_ATTRIBUTES(
      GDP_ID                       : in  CGI_TYPES.INTEGER_TYPE ;
      RESPONSE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
      NUM_VALID_ATTRIBUTE_GROUPS   : out CGI_TYPES.INTEGER_TYPE ;
      ARRAY_OF_ATTRIBUTE_GROUPS    : out CGI_TYPES.BUNDLE_TABLE_ARRAY_TYPE ) ;

    procedure INQ_LINE_CAPABILITY(
      RESPONSE_VALIDITY                  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
      NUM_OF_PREDEFINED_LINE_BUNDLES     : out CGI_TYPES.INTEGER_TYPE ;
      NUM_OF_SETTABLE_LINE_BUNDLES       : out CGI_TYPES.INTEGER_TYPE ;
      MAX_LINE_BUNDLE_INDEX              : out CGI_TYPES.INDEX_TYPE ;
      DYN_MOD_ACCEPTED_FOR_LINE_BUNDLES  : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
      NOMINAL_SCALED_LINE_WIDTH          : out CGI_TYPES.DEVICE_COORD_TYPE ;
      MIN_SCALED_LINE_WIDTH              : out CGI_TYPES.DEVICE_COORD_TYPE ;
      MAX_SCALED_LINE_WIDTH              : out CGI_TYPES.DEVICE_COORD_TYPE ;
```

```
    NUM_OF_VALID_LINE_CLIPPING_MODES     : out CGI_TYPES.INTEGER_TYPE ;
    ARRAY_OF_AVAIL_LINE_CLIPPING_MODES   : out CGI_TYPES.CLIP_MODE_ARRAY_TYPE ) ;

  procedure INQ_AVAIL_LINE_TYPES(
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE  : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_LINE_TYPES                   : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_AVAIL_SCALED_LINE_WIDTHS(
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE  : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_SCALED_LINE_WIDTHS           : out CGI_TYPES.DEVICE_COORD_ARRAY_TYPE
  ) ;

  procedure INQ_MARKER_CAPABILITY(
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    NUM_OF_PREDEFINED_MARKER_BUNDLES     : out CGI_TYPES.INTEGER_TYPE ;
    NUM_OF_SETTABLE_MARKER_BUNDLES       : out CGI_TYPES.INTEGER_TYPE ;
    MAX_MARKER_BUNDLE_INDEX              : out CGI_TYPES.INDEX_TYPE ;
    DYN_MOD_ACCEPTED_FOR_MARKER_BUNDLES  : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    NOMINAL_SCALED_MARKER_SIZE           : out CGI_TYPES.DEVICE_COORD_TYPE ;
    MINIMAL_SCALED_MARKER_SIZE           : out CGI_TYPES.DEVICE_COORD_TYPE ;
    MAX_SCALED_MARKER_SIZE               : out CGI_TYPES.DEVICE_COORD_TYPE ;
    NUM_OF_VALID_MARKER_CLIPPING_MODES   : out CGI_TYPES.INTEGER_TYPE ;
    ARRAY_OF_AVAIL_MARKER_CLIPPING_MODES : out CGI_TYPES.CLIP_MODE_ARRAY_TYPE
  ) ;

  procedure INQ_AVAIL_MARKER_TYPES(
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE  : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_MARKER_TYPES                 : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_AVAIL_SCALED_MARKER_SIZES(
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE  : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_SCALED_MARKER_SIZES          : out CGI_TYPES.DEVICE_COORD_ARRAY_TYPE
  ) ;

  procedure INQ_TEXT_CAPABILITY(
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    NUM_OF_PREDEFINED_TEXT_BUNDLES       : out CGI_TYPES.INTEGER_TYPE ;
    NUM_OF_SETTABLE_TEXT_BUNDLES         : out CGI_TYPES.INTEGER_TYPE ;
```

```
    MAX_TEXT_BUNDLE_INDEX                    : out CGI_TYPES.INDEX_TYPE ;
    DYN_MOD_ACCEPTED_FOR_TEXT_BUNDLES        : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    MAX_LENGTH_OF_FONT_LIST                  : out CGI_TYPES.INTEGER_TYPE ;
    DYN_MOD_ACCEPTED_FOR_FONT_LIST           : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    MAX_LENGTH_OF_CHAR_SET_LIST              : out CGI_TYPES.INTEGER_TYPE ) ;

  procedure INQ_AVAIL_CHAR_SETS(
    NUM_OF_LIST_ELEMENTS_REQUESTED           : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN         : in  CGI_TYPES.INTEGER_TYPE ;
    MAX_CHARS_PER_STRING                     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE      : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_AVAIL_CHAR_SETS                  : out CGI_TYPES.CHAR_SET_ARRAY_TYPE ) ;

  procedure INQ_AVAIL_TEXT_FONTS(
    NUM_OF_LIST_ELEMENTS_REQUESTED           : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN         : in  CGI_TYPES.INTEGER_TYPE ;
    MAX_CHARS_PER_STRING                     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE      : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_FONT_NAMES                       : out CGI_TYPES.FIXED_STRING_ARRAY_TYPE
  ) ;

  procedure INQ_FONT_CAPABILITIES(
    FONT_NAME                                : in  CGI_TYPES.FIXED_STRING_TYPE ;
    PREC                                     : in  CGI_TYPES.TEXT_PREC_TYPE ;
    RESPONSE_VALIDITY                        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MIN_CHAR_EXPANSION_FACTOR                : out CGI_TYPES.REAL_TYPE ;
    MAX_CHAR_EXPANSION_FACTOR                : out CGI_TYPES.REAL_TYPE ;
    MIN_CHAR_SPACING                         : out CGI_TYPES.REAL_TYPE ;
    MAX_CHAR_SPACING                         : out CGI_TYPES.REAL_TYPE ;
    MIN_CHAR_HEIGHT                          : out CGI_TYPES.REAL_TYPE ;
    MAX_CHAR_HEIGHT                          : out CGI_TYPES.REAL_TYPE ;
    SKEWED_VECTOR_SUPPORT                    : out CGI_TYPES.YES_NO_FLAG_TYPE ;
    MIRRORED_CHAR_SUPPORT                    : out CGI_TYPES.YES_NO_FLAG_TYPE ;
    NUM_OF_TEXT_PATH_ELEMENTS                : out CGI_TYPES.INTEGER_TYPE ;
    ARRAY_OF_SUPPORTED_TEXT_PATHS            : out CGI_TYPES.TEXT_PATH_ARRAY_TYPE ;
    NUM_OF_HOR_TEXT_ALIGNMENT_ELEMENTS       : out CGI_TYPES.INTEGER_TYPE ;
    SUPPORTED_HOR_TEXT_ALIGNMENTS            : out CGI_TYPES.
                                                   HOR_ALIGNMENT_ARRAY_TYPE ;
    NUM_OF_VERT_TEXT_ALIGNMENT_ELEMENTS      : out CGI_TYPES.INTEGER_TYPE ;
    SUPPORTED_VERT_TEXT_ALIGNMENTS           : out CGI_TYPES.
                                                   VERT_ALIGNMENT_ARRAY_TYPE ) ;

  procedure INQ_AVAIL_CHAR_EXPAN_FACTORS(
    FONT_NAME                                : in  CGI_TYPES.FIXED_STRING_TYPE ;
    PREC                                     : in  CGI_TYPES.TEXT_PREC_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED           : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN         : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
```

```
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_CHAR_EXPAN_FACTORS          : out CGI_TYPES.REAL_ARRAY_TYPE ) ;


  procedure INQ_AVAIL_CHAR_SPACINGS(
    FONT_NAME                           : in  CGI_TYPES.FIXED_STRING_TYPE ;
    PREC                                : in  CGI_TYPES.TEXT_PREC_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED      : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_CHAR_SPACINGS               : out CGI_TYPES.REAL_ARRAY_TYPE ) ;

   procedure INQ_AVAIL_CHAR_HEIGHTS(
    FONT_NAME                           : in  CGI_TYPES.FIXED_STRING_TYPE ;
    PREC                                : in  CGI_TYPES.TEXT_PREC_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED      : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_CHAR_HEIGHTS                : out CGI_TYPES.DEVICE_COORD_ARRAY_TYPE
  ) ;

  procedure INQ_AVAIL_CHAR_ORIENTATIONS(
    FONT_NAME                           : in  CGI_TYPES.FIXED_STRING_TYPE ;
    PREC                                : in  CGI_TYPES.TEXT_PREC_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED      : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_CHAR_ORIENTATIONS           : out CGI_TYPES.ORIENTATION_ARRAY_TYPE
  ) ;

  procedure INQ_FILL_CAPABILITY(
    RESPONSE_VALIDITY                   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    NUM_OF_PREDEFINED_FILL_BUNDLES      : out CGI_TYPES.INTEGER_TYPE ;
    NUM_OF_SETTABLE_FILL_BUNDLES        : out CGI_TYPES.INTEGER_TYPE ;
    MAX_FILL_BUNDLE_INDEX               : out CGI_TYPES.INDEX_TYPE ;
    DYN_MOD_ACCEPTED_FOR_FILL_BUNDLES   : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    NUM_OF_INTERIOR_STYLE_ELEMENTS      : out CGI_TYPES.INTEGER_TYPE ;
    ARRAY_OF_AVAIL_INTERIOR_STYLES      : out CGI_TYPES.
                                            INTERIOR_STYLE_ARRAY_TYPE ;
    NUM_OF_PREDEFINED_PATTERNS          : out CGI_TYPES.INTEGER_TYPE ;
    NUM_OF_SETTABLE_PATTERNS            : out CGI_TYPES.INTEGER_TYPE ;
    MAX_PATTERN_INDEX                   : out CGI_TYPES.INDEX_TYPE ;
    DYN_MOD_ACCEPTED_FOR_PATTERN_TABLE  : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    PREFERRED_PATTERN_SIZE_DIVISOR      : out CGI_TYPES.INTEGER_TYPE ;
    MAX_PATTERN_SIZE                    : out CGI_TYPES.PATTERN_DIMENSION_TYPE ;
    PATTERN_TRAN_SUPPORT                : out CGI_TYPES.
                                            PATTERN_TRAN_SUPPORT_TYPE ;
```

```
    PATTERN_FILL_FALLBACK                    : out CGI_TYPES.
                                                   PATTERN_FILL_FALLBACK_TYPE ) ;

  procedure INQ_AVAIL_HATCH_STYLES(
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE  : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_HATCH_STYLES                 : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_EDGE_CAPABILITY(
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    NUM_OF_PREDEFINED_EDGE_BUNDLES       : out CGI_TYPES.INTEGER_TYPE ;
    NUM_OF_SETTABLE_EDGE_BUNDLES         : out CGI_TYPES.INTEGER_TYPE ;
    MAX_EDGE_BUNDLE_INDEX                : out CGI_TYPES.INDEX_TYPE ;
    DYN_MOD_ACCEPTED_FOR_EDGE_BUNDLES    : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    NOMINAL_SCALED_EDGE_WIDTH            : out CGI_TYPES.DEVICE_COORD_TYPE ;
    MIN_SCALED_EDGE_WIDTH                : out CGI_TYPES.DEVICE_COORD_TYPE ;
    MAX_SCALED_EDGE_WIDTH                : out CGI_TYPES.DEVICE_COORD_TYPE ;
    NUM_OF_VALID_EDGE_CLIPPING_MODES     : out CGI_TYPES.INTEGER_TYPE ;
    ARRAY_OF_AVAIL_EDGE_CLIPPING_MODES   : out CGI_TYPES.CLIP_MODE_ARRAY_TYPE ;
    REALIZATION_OF_EDGE_WIDTH            : out CGI_TYPES.
                                                   EDGE_WIDTH_REALIZATION_TYPE ) ;

  procedure INQ_AVAIL_EDGE_TYPES(
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE  : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_EDGE_TYPES                   : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_AVAIL_SCALED_EDGE_WIDTHS(
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE  : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_EDGE_WIDTHS                  : out CGI_TYPES.DEVICE_COORD_ARRAY_TYPE
  ) ;

  procedure INQ_COLOUR_CAPABILITY(
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    NUM_SIMULT_AVAIL_DIRECT_COLOURS      : out CGI_TYPES.INTEGER_TYPE ;
    NUM_SIMULT_AVAIL_INDEX_COLOURS       : out CGI_TYPES.INTEGER_TYPE ;
    NUM_AVAIL_COLOURS                    : out CGI_TYPES.INTEGER_TYPE ;
    NUM_AVAIL_INTENSITIES                : out CGI_TYPES.INTENSITY_ARRAY_TYPE ;
    COLOUR_SELECTION_MODE_AVAIL          : out CGI_TYPES.
                                                   COLOUR_SELECTION_MODE_AVAIL_TYPE ;
    DYN_MOD_ACCEPTED_FOR_COLOUR          : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    COLOUR_OVERWRITE_CAPABILITY          : out CGI_TYPES.YES_NO_FLAG_TYPE ;
    COLOUR_REALIZATION                   : out CGI_TYPES.COLOUR_REALIZATION_TYPE ;
```

```
    MONOCHROMATIC_DEVICE                : out CGI_TYPES.YES_NO_FLAG_TYPE ;
    BACKGROUND_COLOUR_CAPABILITY        : out CGI_TYPES.BACKGROUND_COLOUR_CAP_TYPE
  ) ;

  procedure INQ_CIE_CHARACTERISTICS(
    CIE_1931_CHROMATICITY_COORD_FOR_RED   : out CGI_TYPES.CIE_CHROMATICITY_TYPE ;
    CIE_1931_CHROMATICITY_COORD_FOR_GREEN : out CGI_TYPES.CIE_CHROMATICITY_TYPE ;
    CIE_1931_CHROMATICITY_COORD_FOR_BLUE  : out CGI_TYPES.CIE_CHROMATICITY_TYPE ;
    CIE_1931_TRISTIMULUS_VALUES_FOR_WHITE : out CGI_TYPES.
                                              CIE_TRISTIMULUS_VALUE_TYPE ) ;

  procedure INQ_MAX_SIMULTANEOUS_ATTRIBUTE_SETS(
    RESPONSE_VALIDITY                   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MAX_SIMULTANEOUS_ATTRIBUTE_SETS : out CGI_TYPES.FIXED_INTEGER_16_TYPE ) ;

  procedure INQ_SUPPORTED_CHAR_CODING_ANNOUNCERS(
    RESPONSE_VALIDITY                     : out  CGI_TYPES.VALIDITY_FLAG_TYPE ;
    NUM_VALID_CHAR_CODING_ANNOUNCERS : out CGI_TYPES.INTEGER_TYPE ;
    VALID_CHAR_CODING_ANNOUNCERS          : out CGI_TYPES.
                                              CHAR_CODING_ANNOUNCER_ARRAY_TYPE
  ) ;

  procedure INQ_LINE_ATTRIBUTES(
    RESPONSE_VALIDITY             : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    LINE_BUNDLE_INDEX             : out CGI_TYPES.INDEX_TYPE ;
    LINE_TYPE                     : out CGI_TYPES.INDEX_TYPE ;
    LINE_WIDTH_SPECIF_MODE        : out CGI_TYPES.SPECIF_MODE_TYPE ;
    LINE_WIDTH                    : out CGI_TYPES.SIZE_SPECIF_TYPE ;
    LINE_COLOUR_SELECTION_MODE : out CGI_TYPES.COLOUR_SELECTION_MODE_TYPE ;
    LINE_COLOUR                   : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ;
    LINE_CLIPPING_MODE            : out CGI_TYPES.CLIP_MODE_TYPE ) ;

  procedure INQ_LINE_BUNDLE_INDICES(
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_STATE_LIST     : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_LINE_BUNDLE_INDICES      : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_LINE_REP(
    LINE_BUNDLE_INDEX             : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY             : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    LINE_TYPE                     : out CGI_TYPES.INDEX_TYPE ;
    LINE_WIDTH_SPECIF_MODE        : out CGI_TYPES.SPECIF_MODE_TYPE ;
    LINE_WIDTH                    : out CGI_TYPES.SIZE_SPECIF_TYPE ;
    LINE_COLOUR_SELECTION_MODE : out CGI_TYPES.COLOUR_SELECTION_MODE_TYPE ;
    LINE_COLOUR                   : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

  procedure INQ_MARKER_ATTRIBUTES(
    RESPONSE_VALIDITY                 : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
```

```
    MARKER_BUNDLE_INDEX                : out CGI_TYPES.INDEX_TYPE ;
    MARKER_TYPE                        : out CGI_TYPES.INDEX_TYPE ;
    MARKER_SIZE_SPECIF_MODE            : out CGI_TYPES.SPECIF_MODE_TYPE ;
    MARKER_SIZE                        : out CGI_TYPES.SIZE_SPECIF_TYPE ;
    MARKER_COLOUR_SELECTION_MODE : out CGI_TYPES.COLOUR_SELECTION_MODE_TYPE ;
    MARKER_COLOUR                      : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ;
    MARKER_CLIPPING_MODE               : out CGI_TYPES.CLIP_MODE_TYPE ) ;

  procedure INQ_MARKER_BUNDLE_INDICES(
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_STATE_LIST       : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_MARKER_BUNDLE_INDICES      : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_MARKER_REP(
    MARKER_BUNDLE_INDEX                : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MARKER_TYPE                        : out CGI_TYPES.INDEX_TYPE ;
    MARKER_SIZE_SPECIF_MODE            : out CGI_TYPES.SPECIF_MODE_TYPE ;
    MARKER_SIZE                        : out CGI_TYPES.SIZE_SPECIF_TYPE ;
    MARKER_COLOUR_SELECTION_MODE : out CGI_TYPES.COLOUR_SELECTION_MODE_TYPE ;
    MARKER_COLOUR                      : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

  procedure INQ_TEXT_ATTRIBUTES(
    RESPONSE_VALIDITY                  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TEXT_BUNDLE_INDEX                  : out CGI_TYPES.INDEX_TYPE ;
    TEXT_FONT_INDEX                    : out CGI_TYPES.INDEX_TYPE ;
    FONT_PREC                          : out CGI_TYPES.TEXT_PREC_TYPE ;
    CHAR_EXPAN_FACTOR                  : out CGI_TYPES.REAL_TYPE ;
    CHAR_SPACING                       : out CGI_TYPES.REAL_TYPE ;
    SELECTION_MODE_OF_TEXT_COLOUR : out CGI_TYPES.COLOUR_SELECTION_MODE_TYPE ;
    TEXT_COLOUR                        : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ;
    CHAR_HEIGHT                        : out CGI_TYPES.VDC_TYPE ;
    CHAR_UP_VECTOR                     : out CGI_TYPES.CHAR_VECTOR_TYPE ;
    CHAR_BASE_VECTOR                   : out CGI_TYPES.CHAR_VECTOR_TYPE ;
    TEXT_PATH                          : out CGI_TYPES.TEXT_PATH_TYPE ;
    HOR_TEXT_ALIGNMENT                 : out CGI_TYPES.HOR_ALIGNMENT_TYPE ;
    CONT_HOR_ALIGNMENT                 : out CGI_TYPES.REAL_TYPE ;
    VERT_TEXT_ALIGNMENT                : out CGI_TYPES.VERT_ALIGNMENT_TYPE ;
    CONT_VERT_ALIGNMENT                : out CGI_TYPES.REAL_TYPE ;
    CHAR_SET_INDEX                     : out CGI_TYPES.INDEX_TYPE ;
    ALTERNATE_CHAR_SET_INDEX           : out CGI_TYPES.INDEX_TYPE ;
    CHAR_CODING_ANNOUNCER              : out CGI_TYPES.CODING_TECHNIQUE_TYPE ) ;

  procedure INQ_TEXT_BUNDLE_INDICES(
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_STATE_LIST       : out CGI_TYPES.INTEGER_TYPE ;
```

```
    LIST_OF_TEXT_BUNDLE_INDICES          : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_TEXT_REP(
    TEXT_BUNDLE_INDEX                    : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TEXT_FONT_INDEX                      : out CGI_TYPES.INDEX_TYPE ;
    TEXT_PREC                            : out CGI_TYPES.TEXT_PREC_TYPE ;
    CHAR_EXPAN_FACTOR                    : out CGI_TYPES.REAL_TYPE ;
    CHAR_SPACING                         : out CGI_TYPES.REAL_TYPE ;
    TEXT_COLOUR_SELECTION_MODE : out CGI_TYPES.COLOUR_SELECTION_MODE_TYPE ;
    TEXT_COLOUR                          : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

  procedure INQ_FILL_ATTRIBUTES(
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    FILL_BUNDLE_INDEX                    : out CGI_TYPES.INDEX_TYPE ;
    INTERIOR_STYLE                       : out CGI_TYPES.INTERIOR_STYLE_TYPE ;
    FILL_COLOUR_SELECTION_MODE : out CGI_TYPES.COLOUR_SELECTION_MODE_TYPE ;
    FILL_COLOUR                          : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ;
    HATCH_INDEX                          : out CGI_TYPES.INDEX_TYPE ;
    PATTERN_INDEX                        : out CGI_TYPES.INDEX_TYPE ;
    FILL_BITMAP_ID                       : out CGI_TYPES.BITMAP_ID_TYPE ;
    FILL_BITMAP_REGION                   : out CGI_TYPES.BITMAP_REGION_TYPE ;
    FILL_REFERENCE_POINT                 : out CGI_TYPES.VDC_POINT_TYPE ;
    PATTERN_SIZE                         : out CGI_TYPES.PATTERN_SIZE_TYPE ) ;

  procedure INQ_PATTERN_DIMENSIONS(
    PATTERN_INDEX                        : in  CGI_TYPES.INDEX_TYPE ;
    LOCAL_COLOUR_PREC_REQUIREMENT        : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    NUM_OF_HOR_SAMPLES                   : out CGI_TYPES.INTEGER_TYPE ;
    NUM_OF_VERT_SAMPLES                  : out CGI_TYPES.INTEGER_TYPE ;
    LOCAL_COLOUR_PREC                    : out CGI_TYPES.INTEGER_TYPE ;
    SELECTION_MODE_OF_COLOUR_SPECIFIERS : out CGI_TYPES.
                                            COLOUR_SELECTION_MODE_TYPE ) ;

  procedure INQ_PATTERN(
    PATTERN_INDEX              : in  CGI_TYPES.INDEX_TYPE ;
    LOCAL_COLOUR_PREC          : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY          : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    LIST_OF_COLOUR_SPECIFIERS : out CGI_TYPES.COLOUR_SPECIFIER_ARRAY_TYPE ) ;

  procedure INQ_PATTERN_INDICES(
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_STATE_LIST     : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_PATTERN_INDICES          : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_FILL_BUNDLE_INDICES(
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
```

```
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_STATE_LIST     : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_FILL_BUNDLE_INDICES      : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_FILL_REP(
    FILL_BUNDLE_INDEX           : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    INTERIOR_STYLE              : out CGI_TYPES.INTERIOR_STYLE_TYPE ;
    FILL_COLOUR_SELECTION_MODE  : out CGI_TYPES.COLOUR_SELECTION_MODE_TYPE ;
    FILL_COLOUR                 : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ;
    HATCH_INDEX                 : out CGI_TYPES.INDEX_TYPE ;
    PATTERN_INDEX               : out CGI_TYPES.INDEX_TYPE ;
    FILL_BITMAP_ID              : out CGI_TYPES.BITMAP_ID_TYPE ;
    FILL_BITMAP_REGION          : out CGI_TYPES.BITMAP_REGION_TYPE ) ;

  procedure INQ_EDGE_ATTRIBUTES(
    RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    EDGE_BUNDLE_INDEX           : out CGI_TYPES.INDEX_TYPE ;
    EDGE_VISIBILITY             : out CGI_TYPES.VISIBILITY_TYPE ;
    EDGE_TYPE                   : out CGI_TYPES.INDEX_TYPE ;
    EDGE_WIDTH_SPECIF_MODE      : out CGI_TYPES.SPECIF_MODE_TYPE ;
    EDGE_WIDTH                  : out CGI_TYPES.SIZE_SPECIF_TYPE ;
    EDGE_COLOUR_SELECTION_MODE  : out CGI_TYPES.COLOUR_SELECTION_MODE_TYPE ;
    EDGE_COLOUR                 : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ;
    EDGE_CLIPPING_MODE          : out CGI_TYPES.CLIP_MODE_TYPE ) ;

  procedure INQ_EDGE_BUNDLE_INDICES(
    NUM_OF_LIST_ELEMENTS_REQUESTED : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY              : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_STATE_LIST   : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_EDGE_BUNDLE_INDICES    : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_EDGE_REP(
    EDGE_BUNDLE_INDEX           : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    EDGE_VISIBILITY             : out CGI_TYPES.VISIBILITY_TYPE ;
    EDGE_TYPE                   : out CGI_TYPES.INDEX_TYPE ;
    EDGE_WIDTH_SPECIF_MODE      : out CGI_TYPES.SPECIF_MODE_TYPE ;
    EDGE_WIDTH                  : out CGI_TYPES.SIZE_SPECIF_TYPE ;
    EDGE_COLOUR_SELECTION_MODE  : out CGI_TYPES.COLOUR_SELECTION_MODE_TYPE ;
    EDGE_COLOUR                 : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

  procedure INQ_OUTPUT_STATE(
    RESPONSE_VALIDITY       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    OUTPUT_STATE            : out CGI_TYPES.OUTPUT_STATE_TYPE ;
    LINE_WIDTH_SPECIF_MODE  : out CGI_TYPES.SPECIF_MODE_TYPE ;
    MARKER_SIZE_SPECIF_MODE : out CGI_TYPES.SPECIF_MODE_TYPE ;
    EDGE_WIDTH_SPECIF_MODE  : out CGI_TYPES.SPECIF_MODE_TYPE ) ;
```

```
procedure INQ_OBJECT_CLIPPING(
  RESPONSE_VALIDITY : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  CLIP_INDICATOR    : out CGI_TYPES.ON_OFF_FLAG_TYPE ;
  CLIP_RECTANGLE    : out CGI_TYPES.CLIP_RECTANGLE_TYPE ) ;

procedure INQ_ATTRIBUTE_SET_NAMES_IN_USE(
  NUM_OF_LIST_ELEMENTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_STATE_LIST     : out CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_ATTRIBUTE_SET_NAMES      : out CGI_TYPES.ASN_ARRAY_TYPE ) ;

procedure INQ_COLOUR_STATE(
  RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  COLOUR_SELECTION_MODE            : out CGI_TYPES.
                                         COLOUR_SELECTION_MODE_TYPE ;
  COLOUR_VALUE_EXTENT              : out CGI_TYPES.
                                         COLOUR_VALUE_EXTENT_TYPE ;
  AUXILIARY_COLOUR_SELECTION_MODE  : out CGI_TYPES.
                                         COLOUR_SELECTION_MODE_TYPE ;
  AUXILIARY_COLOUR                 : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ;
  TRANSPARENCY                     : out CGI_TYPES.TRANSPARENCY_TYPE ;
  BACKGROUND_COLOUR_SELECTION_MODE : out CGI_TYPES.
                                         COLOUR_SELECTION_MODE_TYPE ;
  BACKGROUND_COLOUR                : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

procedure INQ_COLOUR_TABLE_ENTRIES(
  NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE  : out CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_COLOUR_TABLE_ENTRIES         : out CGI_TYPES.COLOUR_TABLE_TYPE ) ;

procedure INQ_FONT_LIST(
  NUM_OF_LIST_ELEMENTS_REQUESTED           : in  CGI_TYPES.INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN         : in  CGI_TYPES.INTEGER_TYPE ;
  MAX_CHARS_PER_STRING                     : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY                        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_LENGTH_OF_LONGEST_STRING_PRESENT   : out CGI_TYPES.INTEGER_TYPE ;
  TOTAL_ELEMENTS_IN_STATE_LIST             : out CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_FONT_NAMES                       : out CGI_TYPES.
                                                 FIXED_STRING_ARRAY_TYPE ) ;

procedure INQ_CHAR_SET_LIST(
  NUM_OF_LIST_ELEMENTS_REQUESTED           : in  CGI_TYPES.INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN         : in  CGI_TYPES.INTEGER_TYPE ;
  MAX_CHARS_PER_STRING                     : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY                        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_LENGTH_OF_LONGEST_STRING_PRESENT   : out CGI_TYPES.INTEGER_TYPE ;
  TOTAL_ELEMENTS_IN_STATE_LIST             : out CGI_TYPES.INTEGER_TYPE ;
```

```
    LIST_OF_CHAR_SETS                          : out CGI_TYPES.CHAR_SET_ARRAY_TYPE
  ) ;

  procedure LOOKUP_ASPECT_SOURCE_FLAGS(
    LIST_OF_ASF_TYPES_REQUESTED : in  CGI_TYPES.ASF_ARRAY_TYPE ;
    RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    LIST_OF_ASF_VALUES          : out CGI_TYPES.ASF_VALUE_ARRAY_TYPE ) ;

                    ---------- CGI Segment Functions ----------

  procedure GET_NEW_SEGMENT_ID(
    RESPONSE_VALIDITY : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    SEGMENT_ID        : out CGI_TYPES.SEGMENT_ID_TYPE ) ;

  procedure CREATE_SEGMENT(
    SEGMENT_ID : in CGI_TYPES.SEGMENT_ID_TYPE ) ;

  procedure REOPEN_SEGMENT(
    SEGMENT_ID : in CGI_TYPES.SEGMENT_ID_TYPE ) ;

  procedure CLOSE_SEGMENT ;

  procedure COPY_SEGMENT(
    SOURCE_SEGMENT_ID          : in CGI_TYPES.SEGMENT_ID_TYPE ;
    COPY_TRAN                  : in CGI_TYPES.SEGMENT_TRANSFORMATION_TYPE ;
    SEGMENT_TRAN_ASSOCIATION : in CGI_TYPES.YES_NO_FLAG_TYPE ) ;

  procedure DELETE_SEGMENT(
    SEGMENT_ID : in CGI_TYPES.SEGMENT_ID_TYPE ) ;


  procedure DELETE_ALL_SEGMENTS ;

  procedure RENAME_SEGMENT(
    OLD_SEGMENT_ID  : in CGI_TYPES.SEGMENT_ID_TYPE ;
    NEW_SEGMENT_ID : in CGI_TYPES.SEGMENT_ID_TYPE ) ;

  procedure DRAW_ALL_SEGMENTS ;

  procedure SET_IMPL_SEG_REGEN_MODE(
    IMPLICIT_SEGMENT_REGEN_MODE : in CGI_TYPES.IMPLICIT_SEGMENT_REGEN_MODE_TYPE
  ) ;

  procedure RESET_REGEN_PENDING ;

  procedure SET_PICK_ID(
    PICK_ID : in CGI_TYPES.PICK_ID_TYPE ) ;

  procedure SET_SEGMENT_VISIBILITY(
    SEGMENT_ID : in CGI_TYPES.SEGMENT_ID_TYPE ;
```

```
    VISIBILITY    : in CGI_TYPES.VISIBILITY_TYPE ) ;

  procedure SET_SEGMENT_TRANSFORMATION(
    SEGMENT_ID      : in CGI_TYPES.SEGMENT_ID_TYPE ;
    SEGMENT_TRAN : in CGI_TYPES.SEGMENT_TRANSFORMATION_TYPE ) ;

  procedure SET_SEGMENT_HIGHLIGHTING(
    SEGMENT_ID    : in CGI_TYPES.SEGMENT_ID_TYPE ;
    HIGHLIGHTING : in CGI_TYPES.HIGHLIGHTING_TYPE ) ;

  procedure SET_SEGMENT_DISP_PRIORITY(
    SEGMENT_ID    : in CGI_TYPES.SEGMENT_ID_TYPE ;
    DISP_PRIORITY : in CGI_TYPES.INTEGER_TYPE ) ;

  procedure SET_SEGMENT_DETECTABILITY(
    SEGMENT_ID      : in CGI_TYPES.SEGMENT_ID_TYPE ;
    DETECTABILITY : in CGI_TYPES.DETECTABILITY_TYPE ) ;

  procedure SET_SEGMENT_PICK_PRIORITY(
    SEGMENT_ID    : in CGI_TYPES.SEGMENT_ID_TYPE ;
    PICK_PRIORITY : in CGI_TYPES.INTEGER_TYPE ) ;

  procedure SIMULATE_PICK(
    PICK_POINT                          : in  CGI_TYPES.VDC_POINT_TYPE ;
    PICK_APERTURE                       : in  CGI_TYPES.PICK_APERTURE_TYPE ;
    NUM_OF_REQUESTED_PICK_VALUES        : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_AVAIL_PICK_VALUES      : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_PICK_VALUES                 : out CGI_TYPES.PICK_VALUE_ARRAY_TYPE ) ;

  procedure SET_INHERITANCE_FILTER(
    FILTER_SELECTION_LIST : in CGI_TYPES.FILTER_SELECTION_LIST_TYPE ;
    SELECTION_SETTING     : in CGI_TYPES.INHERITANCE_FILTER_TYPE ) ;

  procedure SET_CLIPPING_INHERITANCE(
    SELECTION_SETTING : CGI_TYPES.CLIP_INHERITANCE_FILTER_TYPE ) ;

  procedure INQ_SEGMENT_CAPABILITY(
    RESPONSE_VALIDITY                       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    NUM_OF_SUPPORTED_DISPLAY_PRIORITIES     : out CGI_TYPES.INTEGER_TYPE ;
    NUM_OF_SUPPORTED_PICK_PRIORITIES        : out CGI_TYPES.INTEGER_TYPE ;
    MAX_NUM_OF_SIMULTANEOUS_SEGMENTS        : out CGI_TYPES.INTEGER_TYPE ;
    MAX_LEN_OF_SIMULATE_PICK_LIST           : out CGI_TYPES.INTEGER_TYPE ;
    DYN_MOD_FOR_ADDING_TO_OPEN_SEGMENT      : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    DYN_MOD_FOR_SEGMENT_DELETION            : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    DYN_MOD_FOR_SEGMENT_TRANSFORMATION      : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    DYN_MOD_FOR_DISPLAY_PRIORITY_CHANGE     : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    DYN_MOD_FOR_VISIBILITY_TO_INVISIBLE     : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    DYN_MOD_FOR_VISIBILITY_TO_VISIBLE       : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
    DYN_MOD_FOR_HIGHLIGHTING_CHANGE         : out CGI_TYPES.DYN_MOD_FLAG_TYPE ;
```

```
    EFFECTIVE_RELATIVE_DISPLAY_PRIORITY      : out CGI_TYPES.
                                                 EFFECTIVE_PRIORITY_TYPE ;
    PICK_APERTURE_SHAPE                      : out CGI_TYPES.
                                                 PICK_APERTURE_SHAPE_TYPE ;
    TRANSFORMED_CLIP_REGION_EFFECTIVENESS : out CGI_TYPES.CLIP_EFFECTIVENESS_TYPE
  ) ;

  procedure INQ_SEGMENT_STATE(
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    SEGMENT_OPEN_STATE               : out CGI_TYPES.SEGMENT_OPEN_STATE_TYPE ;
    ID_OF_OPEN_SEGMENT               : out CGI_TYPES.SEGMENT_ID_TYPE ;
    PICK_ID                          : out CGI_TYPES.PICK_ID_TYPE ;
    IMPLICIT_SEGMENT_REGEN_MODE : out CGI_TYPES.
                                        IMPLICIT_SEGMENT_REGEN_MODE_TYPE ;
    REGEN_PENDING                    : out CGI_TYPES.YES_NO_FLAG_TYPE ) ;

  procedure INQ_INHERITANCE_FILTER_SETTINGS(
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_PRESENT           : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_ATTRIBUTE_DESIGNATORS    : out CGI_TYPES.
                                           INHERITANCE_VALUE_ARRAY_TYPE ) ;

  procedure INQ_CLIPPING_INHERITANCE(
    RESPONSE_VALIDITY       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    CLIP_INHERITANCE_FILTER : out CGI_TYPES.CLIP_INHERITANCE_FILTER_TYPE ) ;

  procedure INQ_SEGMENT_IDS_IN_USE(
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_PRESENT           : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_SEGMENT_IDS              : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_INDIV_SEGMENT_STATE(
    SEGMENT_ID                : in  CGI_TYPES.SEGMENT_ID_TYPE ;
    RESPONSE_VALIDITY         : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    SEGMENT_TRAN              : out CGI_TYPES.SEGMENT_TRANSFORMATION_TYPE ;
    SEGMENT_VISIBILITY        : out CGI_TYPES.VISIBILITY_TYPE ;
    SEGMENT_HIGHLIGHTING      : out CGI_TYPES.HIGHLIGHTING_TYPE ;
    SEGMENT_DISPLAY_PRIORITY  : out CGI_TYPES.INTEGER_TYPE ;
    SEGMENT_DETECTABILITY     : out CGI_TYPES.DETECTABILITY_TYPE ;
    SEGMENT_PICK_PRIORITY     : out CGI_TYPES.INTEGER_TYPE ) ;

                    ---------- CGI Input Functions ----------

  procedure INITIALIZE_LOGICAL_INPUT_DEVICE(
    INPUT_CLASS        : in CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ) ;
```

```
procedure RELEASE_LOGICAL_INPUT_DEVICE(
  INPUT_CLASS        : in CGI_TYPES.INPUT_CLASS_TYPE ;
  INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

procedure SET_ECHO_CONTROLS(
  INPUT_CLASS        : in CGI_TYPES.INPUT_CLASS_TYPE ;
  INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
  PROMPT_CONTROL     : in CGI_TYPES.DISABLED_ENABLED_FLAG_TYPE ;
  ECHO_CONTROL       : in CGI_TYPES.DISABLED_ENABLED_FLAG_TYPE ;
  ACK_CONTROL        : in CGI_TYPES.DISABLED_ENABLED_FLAG_TYPE ) ;

procedure PUT_CURRENT_LOCATOR_MEASURE(
  INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY   : in CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALUE      : in CGI_TYPES.VDC_POINT_TYPE ) ;

procedure PUT_CURRENT_STROKE_MEASURE(
  INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY   : in CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALUE      : in CGI_TYPES.POINT_LIST_TYPE ) ;

procedure PUT_CURRENT_VALUATOR_MEASURE(
  INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY   : in CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALUE      : in CGI_TYPES.REAL_TYPE ) ;

procedure PUT_CURRENT_CHOICE_MEASURE(
  INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY   : in CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALUE      : in CGI_TYPES.INTEGER_TYPE ) ;

procedure PUT_PICK_CURRENT_MEASURE(
  INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY   : in CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALUE      : in CGI_TYPES.PICK_VALUE_ARRAY_TYPE ) ;

procedure PUT_CURRENT_STRING_MEASURE(
  INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY   : in CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALUE      : in CGI_TYPES.STRING_TYPE ) ;

procedure PUT_CURRENT_RASTER_MEASURE(
  INPUT_DEVICE_INDEX              : in CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY                : in CGI_TYPES.VALIDITY_FLAG_TYPE ;
  XCOUNT                          : in CGI_TYPES.INTEGER_TYPE ;
  YCOUNT                          : in CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_INPUT_COLOUR_VALUES : in CGI_TYPES.INPUT_COLOUR_ARRAY_TYPE ) ;

procedure PUT_CURRENT_GENERAL_MEASURE(
  INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
```

```
    MEASURE_VALIDITY   : in CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MEASURE_VALUE      : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure SET_ECHO_DATA(
    INPUT_CLASS        : in CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
    PROMPT_TYPE        : in CGI_TYPES.INDEX_TYPE ;
    ECHO_TYPE          : in CGI_TYPES.INDEX_TYPE ;
    ACK_TYPE           : in CGI_TYPES.INDEX_TYPE ;
    ECHO_AREA          : in CGI_TYPES.ECHO_AREA_TYPE ;
    ECHO_DATA_RECORD   : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure SET_LOCATOR_DEVICE_DATA(
    INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
    INPUT_EXTENT       : in CGI_TYPES.INPUT_EXTENT_TYPE ;
    INPUT_VIEWPORT     : in CGI_TYPES.INPUT_VIEWPORT_TYPE ;
    ECHO_VIEWPORT      : in CGI_TYPES.ECHO_AREA_TYPE ;
    DATA_RECORD        : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure SET_STROKE_DEVICE_DATA(
    INPUT_DEVICE_INDEX                : in CGI_TYPES.INDEX_TYPE ;
    MAX_NUM_OF_POINTS                 : in CGI_TYPES.INTEGER_TYPE ;
    SAMPLING_INTERVAL                 : in CGI_TYPES.STROKE_SAMPLING_INTERVAL_TYPE ;
    MIN_TIME_INTERVAL_PER_SAMPLE      : in CGI_TYPES.INTEGER_TYPE ;
    MAX_TIME_INTERVAL_PER_SAMPLE      : in CGI_TYPES.INTEGER_TYPE ;
    INPUT_EXTENT                      : in CGI_TYPES.INPUT_EXTENT_TYPE ;
    INPUT_VIEWPORT                    : in CGI_TYPES.INPUT_VIEWPORT_TYPE ;
    ECHO_VIEWPORT                     : in CGI_TYPES.ECHO_AREA_TYPE ;
    DATA_RECORD                       : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure SET_VALUATOR_DEVICE_DATA(
    INPUT_DEVICE_INDEX    : in CGI_TYPES.INDEX_TYPE ;
    MIN_VALUE_OF_RANGE    : in CGI_TYPES.REAL_TYPE ;
    MAX_VALUE_OF_RANGE    : in CGI_TYPES.REAL_TYPE ;
    DATA_RECORD           : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure SET_CHOICE_DEVICE_DATA(
    INPUT_DEVICE_INDEX                : in CGI_TYPES.INDEX_TYPE ;
    MAX_NUM_OF_CHOICE_ALTERNATIVES    : in CGI_TYPES.INTEGER_TYPE ;
    DATA_RECORD                       : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE
  ) ;

  procedure SET_PICK_DEVICE_DATA(
    INPUT_DEVICE_INDEX          : in CGI_TYPES.INDEX_TYPE ;
    PICK_APERTURE               : in CGI_TYPES.PICK_APERTURE_TYPE ;
    MAX_NUM_OF_SEGMENT_PICKS    : in CGI_TYPES.INTEGER_TYPE ;
    INPUT_EXTENT                : in CGI_TYPES.INPUT_EXTENT_TYPE ;
    INPUT_VIEWPORT              : in CGI_TYPES.INPUT_VIEWPORT_TYPE ;
    ECHO_VIEWPORT               : in CGI_TYPES.ECHO_AREA_TYPE ;
    DATA_RECORD                 : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;
```

```
procedure SET_STRING_DEVICE_DATA(
  INPUT_DEVICE_INDEX                  : in CGI_TYPES.INDEX_TYPE ;
  MAX_STRING_SIZE                     : in CGI_TYPES.INTEGER_TYPE ;
  INPUT_CHAR_SET_INDEX                : in CGI_TYPES.INDEX_TYPE ;
  ALTERNATE_INPUT_CHAR_SET_INDEX      : in CGI_TYPES.INDEX_TYPE ;
  INPUT_CHAR_CODING_ANNOUNCER         : in CGI_TYPES.CODING_TECHNIQUE_TYPE ;
  DATA_RECORD                         : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE
) ;

procedure SET_RASTER_DEVICE_DATA(
  INPUT_DEVICE_INDEX                  : in CGI_TYPES.INDEX_TYPE ;
  SPOT_CENTRE_SEPARATIONS             : in CGI_TYPES.SPOT_CENTRE_SEPARATION_TYPE ;
  COLOUR_CAPABILITY                   : in CGI_TYPES.YES_NO_FLAG_TYPE ;
  THRESHOLD_LEVEL                     : in CGI_TYPES.INTEGER_TYPE ;
  BITS_PER_COLOUR                     : in CGI_TYPES.INTEGER_TYPE ;
  SOURCE_WINDOW_FIRST_CORNER          : in CGI_TYPES.RASTER_WINDOW_CORNER_TYPE ;
  SOURCE_WINDOW_SECOND_CORNER         : in CGI_TYPES.RASTER_WINDOW_CORNER_TYPE ;
  DATA_RECORD                         : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

procedure SET_GENERAL_DEVICE_DATA(
  INPUT_DEVICE_INDEX      : in CGI_TYPES.INDEX_TYPE ;
  MAX_DATA_RECORD_SIZE    : in CGI_TYPES.INTEGER_TYPE ;
  MEASURE_FORMAT_ID       : in CGI_TYPES.INTEGER_TYPE ;
  DATA_RECORD             : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

procedure ASSOCIATE_TRIGGERS(
  INPUT_CLASS             : in CGI_TYPES.INPUT_CLASS_TYPE ;
  INPUT_DEVICE_INDEX      : in CGI_TYPES.INDEX_TYPE ;
  LIST_OF_TRIGGERS        : in CGI_TYPES.INDEX_ARRAY_TYPE ) ;

procedure GET_ADDITIONAL_STROKE_DATA(
  NUM_OF_REQUESTED_POINTS : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  LIST_OF_POINTS          : out CGI_TYPES.POINT_LIST_TYPE ) ;

procedure GET_ADDITIONAL_PICK_DATA(
  NUM_OF_REQUESTED_PICK_VALUES : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  LIST_OF_PICK_VALUES          : out CGI_TYPES.PICK_VALUE_ARRAY_TYPE ) ;

procedure GET_ADDITIONAL_STRING_DATA(
  NUM_OF_REQUESTED_CHARS : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY      : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  STRING_DATA            : out CGI_TYPES.STRING_TYPE ) ;

procedure GET_ADDITIONAL_RASTER_DATA(
  NUM_OF_REQUESTED_COLOUR_VALUES : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY              : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  LIST_OF_INPUT_COLOUR_VALUES    : out CGI_TYPES.INPUT_COLOUR_ARRAY_TYPE ) ;
```

```
procedure REQUEST_LOCATOR(
  INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
  TIMEOUT            : in  CGI_TYPES.REAL_TYPE ;
  RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  REQUEST_STATUS     : out CGI_TYPES.REQUEST_STATUS_TYPE ;
  TRIGGER            : out CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  POSITION           : out CGI_TYPES.VDC_POINT_TYPE ) ;

procedure REQUEST_STROKE(
  INPUT_DEVICE_INDEX          : in  CGI_TYPES.INDEX_TYPE ;
  TIMEOUT                     : in  CGI_TYPES.REAL_TYPE ;
  NUM_OF_REQUESTED_POINTS     : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  REQUEST_STATUS              : out CGI_TYPES.REQUEST_STATUS_TYPE ;
  TRIGGER                     : out CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_NUM_OF_POINTS         : out CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_POINTS              : out CGI_TYPES.POINT_LIST_TYPE ) ;

procedure REQUEST_VALUATOR(
  INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
  TIMEOUT            : in  CGI_TYPES.REAL_TYPE ;
  RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  REQUEST_STATUS     : out CGI_TYPES.REQUEST_STATUS_TYPE ;
  TRIGGER            : out CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  VALUE              : out CGI_TYPES.REAL_TYPE ) ;

procedure REQUEST_CHOICE(
  INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
  TIMEOUT            : in  CGI_TYPES.REAL_TYPE ;
  RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  REQUEST_STATUS     : out CGI_TYPES.REQUEST_STATUS_TYPE ;
  TRIGGER            : out CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  CHOICE_NUMBER      : out CGI_TYPES.INTEGER_TYPE ) ;

procedure REQUEST_PICK(
  INPUT_DEVICE_INDEX          : in  CGI_TYPES.INDEX_TYPE ;
  TIMEOUT                     : in  CGI_TYPES.REAL_TYPE ;
  NUM_OF_REQUESTED_PICK_VALUES : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  REQUEST_STATUS              : out CGI_TYPES.REQUEST_STATUS_TYPE ;
  TRIGGER                     : out CGI_TYPES.INDEX_TYPE ;
  MEASURE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_NUM_OF_PICK_VALUES    : out CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_PICK_VALUES         : out CGI_TYPES.PICK_VALUE_ARRAY_TYPE ) ;

procedure REQUEST_STRING(
```

```
    INPUT_DEVICE_INDEX          : in  CGI_TYPES.INDEX_TYPE ;
    TIMEOUT                     : in  CGI_TYPES.REAL_TYPE ;
    NUM_OF_REQUESTED_CHARS      : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS              : out CGI_TYPES.REQUEST_STATUS_TYPE ;
    TRIGGER                     : out CGI_TYPES.INDEX_TYPE ;
    MEASURE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_CHARS          : out CGI_TYPES.INTEGER_TYPE ;
    STRING_MEASURE              : out CGI_TYPES.STRING_TYPE ) ;

  procedure REQUEST_RASTER(
    INPUT_DEVICE_INDEX                          : in  CGI_TYPES.INDEX_TYPE ;
    TIMEOUT                                     : in  CGI_TYPES.REAL_TYPE ;
    NUM_OF_REQUESTED_INPUT_COLOUR_VALUES        : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS                              : out CGI_TYPES.REQUEST_STATUS_TYPE ;
    TRIGGER                                     : out CGI_TYPES.INDEX_TYPE ;
    MEASURE_VALIDITY                            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_INPUT_COLOUR_VALUES            : out CGI_TYPES.INTEGER_TYPE ;
    XCOUNT                                      : out CGI_TYPES.INTEGER_TYPE ;
    YCOUNT                                      : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_INPUT_COLOUR_VALUES                 : out CGI_TYPES.
                                                      INPUT_COLOUR_ARRAY_TYPE ) ;

  procedure REQUEST_GENERAL(
    INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
    TIMEOUT            : in  CGI_TYPES.REAL_TYPE ;
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS     : out CGI_TYPES.REQUEST_STATUS_TYPE ;
    TRIGGER            : out CGI_TYPES.INDEX_TYPE ;
    MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    DATA_RECORD        : out CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure SET_SAMPLING_STATE(
    INPUT_CLASS        : in CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
    SAMPLE_STATE       : in CGI_TYPES.DISABLED_ENABLED_FLAG_TYPE ) ;

  procedure SAMPLE_LOCATOR(
    INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    POSITION           : out CGI_TYPES.VDC_POINT_TYPE ) ;

  procedure SAMPLE_STROKE(
    INPUT_DEVICE_INDEX          : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_REQUESTED_POINTS     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_POINTS         : out CGI_TYPES.INTEGER_TYPE ;
```

```
    LIST_OF_POINTS                  : out CGI_TYPES.POINT_LIST_TYPE ) ;

  procedure SAMPLE_VALUATOR(
    INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    VALUE              : out CGI_TYPES.REAL_TYPE ) ;

  procedure SAMPLE_CHOICE(
    INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    CHOICE_NUMBER      : out CGI_TYPES.INTEGER_TYPE ) ;

  procedure SAMPLE_PICK(
    INPUT_DEVICE_INDEX              : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_REQUESTED_PICK_VALUES    : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY               : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_PICK_VALUES        : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_PICK_VALUES             : out CGI_TYPES.PICK_VALUE_ARRAY_TYPE ) ;

  procedure SAMPLE_STRING(
    INPUT_DEVICE_INDEX      : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_REQUESTED_CHARS  : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_CHARS      : out CGI_TYPES.INTEGER_TYPE ;
    STRING_MEASURE          : out CGI_TYPES.STRING_TYPE ) ;

  procedure SAMPLE_RASTER(
    INPUT_DEVICE_INDEX                  : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_REQUESTED_INPUT_COLOUR_VALUES : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_INPUT_COLOUR_VALUES    : out CGI_TYPES.INTEGER_TYPE ;
    XCOUNT                              : out CGI_TYPES.INTEGER_TYPE ;
    YCOUNT                              : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_INPUT_COLOUR_VALUES         : out CGI_TYPES.
                                            INPUT_COLOUR_ARRAY_TYPE ) ;

  procedure SAMPLE_GENERAL(
    INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    DATA_RECORD        : out CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure INITIALIZE_ECHO_REQUEST(
    INPUT_CLASS        : in CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ;
```

```
    TIMEOUT                 : in CGI_TYPES.REAL_TYPE ) ;

  procedure ECHO_REQUEST_LOCATOR(
    INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS     : out CGI_TYPES.REQUEST_STATUS_TYPE ;
    TRIGGER            : out CGI_TYPES.INDEX_TYPE ;
    MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    POSITION           : out CGI_TYPES.VDC_POINT_TYPE ) ;

  procedure ECHO_REQUEST_STROKE(
    INPUT_DEVICE_INDEX          : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_REQUESTED_POINTS     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS              : out CGI_TYPES.REQUEST_STATUS_TYPE ;
    TRIGGER                     : out CGI_TYPES.INDEX_TYPE ;
    MEASURE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_POINTS         : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_POINTS              : out CGI_TYPES.POINT_LIST_TYPE ) ;

  procedure ECHO_REQUEST_VALUATOR(
    INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS     : out CGI_TYPES.REQUEST_STATUS_TYPE ;
    TRIGGER            : out CGI_TYPES.INDEX_TYPE ;
    MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    VALUE              : out CGI_TYPES.REAL_TYPE ) ;

  procedure ECHO_REQUEST_CHOICE(
    INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS     : out CGI_TYPES.REQUEST_STATUS_TYPE ;
    TRIGGER            : out CGI_TYPES.INDEX_TYPE ;
    MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    CHOICE_NUMBER      : out CGI_TYPES.INTEGER_TYPE ) ;

  procedure ECHO_REQUEST_PICK(
    INPUT_DEVICE_INDEX          : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_REQUESTED_PICK_VALUES : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS              : out CGI_TYPES.REQUEST_STATUS_TYPE ;
    TRIGGER                     : out CGI_TYPES.INDEX_TYPE ;
    MEASURE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_PICK_VALUES    : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_PICK_VALUES         : out CGI_TYPES.PICK_VALUE_ARRAY_TYPE ) ;

  procedure ECHO_REQUEST_STRING(
    INPUT_DEVICE_INDEX      : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_REQUESTED_CHARS  : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
```

```
    REQUEST_STATUS              : out CGI_TYPES.REQUEST_STATUS_TYPE ;
    TRIGGER                     : out CGI_TYPES.INDEX_TYPE ;
    MEASURE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_CHARS          : out CGI_TYPES.INTEGER_TYPE ;
    STRING_MEASURE              : out CGI_TYPES.STRING_TYPE ) ;

  procedure ECHO_REQUEST_RASTER(
    INPUT_DEVICE_INDEX                    : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_REQUESTED_INPUT_COLOUR_VALUES  : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                     : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS                        : out CGI_TYPES.REQUEST_STATUS_TYPE ;
    TRIGGER                               : out CGI_TYPES.INDEX_TYPE ;
    MEASURE_VALIDITY                      : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_NUM_OF_INPUT_COLOUR_VALUES      : out CGI_TYPES.INTEGER_TYPE ;
    XCOUNT                                : out CGI_TYPES.INTEGER_TYPE ;
    YCOUNT                                : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_INPUT_COLOUR_VALUES           : out CGI_TYPES.
                                                INPUT_COLOUR_ARRAY_TYPE ) ;

  procedure ECHO_REQUEST_GENERAL(
    INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    REQUEST_STATUS     : out CGI_TYPES.REQUEST_STATUS_TYPE ;
    TRIGGER            : out CGI_TYPES.INDEX_TYPE ;
    MEASURE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    DATA_RECORD        : out CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure INITIALIZE_EVENT_QUEUE(
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    INITIAL_TIME_STAMP : out CGI_TYPES.REAL_TYPE ) ;

  procedure RELEASE_EVENT_QUEUE ;

  procedure ENABLE_EVENTS(
    INPUT_CLASS        : in CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

  procedure DISABLE_EVENTS(
    INPUT_CLASS        : in CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

  procedure EVENT_QUEUE_BLOCK_CONTROL(
    CONTROL : in CGI_TYPES.BLOCK_CONTROL_TYPE ) ;

  procedure FLUSH_EVENTS ;

  procedure FLUSH_DEVICE_EVENTS(
    INPUT_CLASS        : in CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX : in CGI_TYPES.INDEX_TYPE ) ;
```

```
procedure AWAIT_EVENT(
  TIMEOUT                : in  CGI_TYPES.REAL_TYPE ;
  RESPONSE_VALIDITY      : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  EVENT_STATUS           : out CGI_TYPES.EVENT_STATUS_TYPE ;
  INPUT_CLASS            : out CGI_TYPES.INPUT_CLASS_TYPE ;
  INPUT_DEVICE_INDEX     : out CGI_TYPES.INDEX_TYPE ;
  INPUT_TRIGGER_INDEX    : out CGI_TYPES.INDEX_TYPE ;
  TIMESTAMP              : out CGI_TYPES.REAL_TYPE ) ;

procedure DEQUEUE_LOCATOR_EVENT(
  RESPONSE_VALIDITY : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  POSITION          : out CGI_TYPES.VDC_POINT_TYPE ) ;

procedure DEQUEUE_STROKE_EVENT(
  NUM_OF_REQUESTED_POINTS : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALIDITY        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_NUM_OF_POINTS     : out CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_POINTS          : out CGI_TYPES.POINT_LIST_TYPE ) ;

procedure DEQUEUE_VALUATOR_EVENT(
  RESPONSE_VALIDITY : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  VALUE             : out CGI_TYPES.REAL_TYPE ) ;

procedure DEQUEUE_CHOICE_EVENT(
  RESPONSE_VALIDITY : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  CHOICE_NUMBER     : out CGI_TYPES.INTEGER_TYPE ) ;

procedure DEQUEUE_PICK_EVENT(
  NUM_OF_REQUESTED_PICK_VALUES : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY            : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALIDITY             : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_NUM_OF_PICK_VALUES     : out CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_PICK_VALUES          : out CGI_TYPES.PICK_VALUE_ARRAY_TYPE ) ;

procedure DEQUEUE_STRING_EVENT(
  NUM_OF_REQUESTED_CHARS : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY      : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALIDITY       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_NUM_OF_CHARS     : out CGI_TYPES.INTEGER_TYPE ;
  STRING_MEASURE         : out CGI_TYPES.STRING_TYPE ) ;

procedure DEQUEUE_RASTER_EVENT(
  NUM_OF_REQUESTED_INPUT_COLOUR_VALUES : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MEASURE_VALIDITY                     : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_NUM_OF_INPUT_COLOUR_VALUES     : out CGI_TYPES.INTEGER_TYPE ;
```

```
    XCOUNT                          : out CGI_TYPES.INTEGER_TYPE ;
    YCOUNT                          : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_INPUT_COLOUR_VALUES     : out CGI_TYPES.
                                          INPUT_COLOUR_ARRAY_TYPE ) ;

  procedure DEQUEUE_GENERAL_EVENT(
    RESPONSE_VALIDITY : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MEASURE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    DATA_RECORD       : out CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure EVENT_QUEUE_TRANSFER(
    RESPONSE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    EVENT_REPORTS_LIST : out CGI_DATA_RECORD_UTILS.EVENT_REPORTS_LIST_TYPE ) ;

  procedure INITIALIZE_ECHO_OUTPUT(
    INPUT_CLASS       : in CGI_TYPES.INPUT_CLASS_TYPE ;
    ECHO_ENTITY_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

  procedure RELEASE_ECHO_OUTPUT(
    INPUT_CLASS       : in CGI_TYPES.INPUT_CLASS_TYPE ;
    ECHO_ENTITY_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

  procedure SET_ECHO_OUTPUT_CONTROLS(
    INPUT_CLASS       : in CGI_TYPES.INPUT_CLASS_TYPE ;
    ECHO_ENTITY_INDEX : in CGI_TYPES.INDEX_TYPE ;
    PROMPT_CONTROL    : in CGI_TYPES.ECHO_ENTITY_PROMPT_CONTROL_TYPE ;
    ECHO_CONTROL      : in CGI_TYPES.ECHO_ENTITY_ECHO_CONTROL_TYPE ) ;

  procedure PERFORM_ACK(
    INPUT_CLASS       : in CGI_TYPES.INPUT_CLASS_TYPE ;
    ECHO_ENTITY_INDEX : in CGI_TYPES.INDEX_TYPE ) ;

  procedure UPDATE_LOCATOR_ECHO_OUTPUT(
    ECHO_ENTITY_INDEX : in CGI_TYPES.INDEX_TYPE ;
    POSITION          : in CGI_TYPES.VDC_POINT_TYPE ) ;

  procedure UPDATE_STROKE_ECHO_OUTPUT(
    ECHO_ENTITY_INDEX : in CGI_TYPES.INDEX_TYPE ;
    LIST_OF_POINTS    : in CGI_TYPES.POINT_LIST_TYPE ) ;

  procedure UPDATE_VALUATOR_ECHO_OUTPUT(
    ECHO_ENTITY_INDEX : in CGI_TYPES.INDEX_TYPE ;
    VALUE             : in CGI_TYPES.REAL_TYPE ) ;

  procedure UPDATE_CHOICE_ECHO_OUTPUT(
    ECHO_ENTITY_INDEX : in CGI_TYPES.INDEX_TYPE ;
    CHOICE_NUMBER     : in CGI_TYPES.INTEGER_TYPE ) ;

  procedure UPDATE_PICK_ECHO_OUTPUT(
    ECHO_ENTITY_INDEX   : in CGI_TYPES.INDEX_TYPE ;
```

```
    LIST_OF_PICK_VALUES : in CGI_TYPES.PICK_VALUE_ARRAY_TYPE ) ;

  procedure UPDATE_STRING_ECHO_OUTPUT(
    ECHO_ENTITY_INDEX : in CGI_TYPES.INDEX_TYPE ;
    STRING_ECHO       : in CGI_TYPES.STRING_TYPE ) ;

  procedure UPDATE_RASTER_ECHO_OUTPUT(
    ECHO_ENTITY_INDEX              : in CGI_TYPES.INDEX_TYPE ;
    XCOUNT                         : in CGI_TYPES.INTEGER_TYPE ;
    YCOUNT                         : in CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_INPUT_COLOUR_VALUES : in CGI_TYPES.INPUT_COLOUR_ARRAY_TYPE ) ;

  procedure UPDATE_GENERAL_ECHO_OUTPUT(
    ECHO_ENTITY_INDEX  : in CGI_TYPES.INDEX_TYPE ;
    MEASURE_FORMAT_ID : in CGI_TYPES.INTEGER_TYPE ;
    DATA_RECORD        : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure SET_ECHO_OUTPUT_DATA(
    INPUT_CLASS                        : in CGI_TYPES.INPUT_CLASS_TYPE ;
    ECHO_ENTITY_INDEX                  : in CGI_TYPES.INDEX_TYPE ;
    PROMPT_TYPE                        : in CGI_TYPES.INDEX_TYPE ;
    ECHO_TYPE                          : in CGI_TYPES.INDEX_TYPE ;
    ACK_TYPE                           : in CGI_TYPES.INDEX_TYPE ;
    ECHO_AREA                          : in CGI_TYPES.ECHO_AREA_TYPE ;
    ECHO_OUTPUT_ECHO_DATA_RECORD : in CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE
  ) ;

  procedure INQ_INPUT_CAPABILITY(
    RESPONSE_VALIDITY     : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TIMEOUT_CAP           : out CGI_TYPES.TIMEOUT_CAP_TYPE ;
    TIMESTAMP_IMPLEMENTATION : out CGI_TYPES.TIMESTAMP_SUPPORT_TYPE ) ;

  procedure INQ_AVAIL_INPUT_DEVICES(
    INPUT_CLASS                          : in  CGI_TYPES.INPUT_CLASS_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_AVAIL_INPUT_DEVICE_INDICES   : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_COMMON_INPUT_DEVICE_PROPERTIES(
    INPUT_CLASS                          : in  CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX                   : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    PHYSICAL_DEVICE_ID                   : out CGI_TYPES.INTEGER_TYPE ;
    NUM_OF_NON_DISSOCIABLE_TRIGGERS : out CGI_TYPES.INTEGER_TYPE ;
    REQUEST_SUPPORT                      : out CGI_TYPES.YES_NO_FLAG_TYPE ;
    ECHO_REQUEST_SUPPORT                 : out CGI_TYPES.YES_NO_FLAG_TYPE ;
    SAMPLE_SUPPORT                       : out CGI_TYPES.YES_NO_FLAG_TYPE ;
    EVENT_SUPPORT                        : out CGI_TYPES.YES_NO_FLAG_TYPE ;
```

```
    PUT_CURRENT_MEASURE_EFFECTIVE    : out CGI_TYPES.YES_NO_FLAG_TYPE ;
    ECHO_CHANGE_SUPPORT              : out CGI_TYPES.YES_NO_FLAG_TYPE ) ;

  procedure INQ_SUPPORTED_ECHO_TYPES(
    INPUT_CLASS                              : in  CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX                       : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED           : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN         : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE      : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_ECHO_TYPES                       : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_SUPPORTED_PROMPT_TYPES(
    INPUT_CLASS                              : in  CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX                       : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED           : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN         : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE      : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_PROMPT_TYPES                     : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_SUPPORTED_ACK_TYPES(
    INPUT_CLASS                              : in  CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX                       : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED           : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN         : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE      : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_ACK_TYPES                        : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_ASSOCIABLE_TRIGGERS(
    INPUT_CLASS                              : in  CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX                       : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED           : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN         : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                        : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE      : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_TRIGGERS                         : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_LOCATOR_CAPABILITIES(
    INPUT_DEVICE_INDEX                : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                 : out CGI_TYPES.VALIDITY_FLAG_TYPE;
    LOWER_LEFT_CORNER                 : out CGI_TYPES.INPUT_SURFACE_POINT_TYPE ;
    UPPER_RIGHT_CORNER                : out CGI_TYPES.INPUT_SURFACE_POINT_TYPE ;
    PHYSICAL_INPUT_SURFACE_SIZE       : out CGI_TYPES.PHYSICAL_INPUT_SURFACE_SIZE_TYPE ;
    INPUT_SURFACE_INTERPRETATION      : out CGI_TYPES.SURFACE_INTERPRETATION_TYPE ;
    NUM_DISTINGUISHABLE_XY_STEPS      : out CGI_TYPES.XY_STEP_TYPE ) ;

  procedure INQ_STROKE_CAPABILITIES(
    INPUT_DEVICE_INDEX                : in  CGI_TYPES.INDEX_TYPE ;
```

```
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE;
    MAX_NUM_OF_POINTS                : out CGI_TYPES.INTEGER_TYPE ;
    LOWER_LEFT_CORNER                : out CGI_TYPES.INPUT_SURFACE_POINT_TYPE ;
    UPPER_RIGHT_CORNER               : out CGI_TYPES.INPUT_SURFACE_POINT_TYPE ;
    PHYSICAL_INPUT_SURFACE_SIZE      : out CGI_TYPES.PHYSICAL_INPUT_SURFACE_SIZE_TYPE ;
    INPUT_SURFACE_INTERPRETATION     : out CGI_TYPES.SURFACE_INTERPRETATION_TYPE ;
    NUM_DISTINGUISHABLE_XY_STEPS     : out CGI_TYPES.XY_STEP_TYPE ) ;

  procedure INQ_CHOICE_CAPABILITIES(
    INPUT_DEVICE_INDEX               : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MAX_NUM_OF_CHOICE_ALTERNATIVES   : out CGI_TYPES.INTEGER_TYPE ) ;

  procedure INQ_PICK_CAPABILITIES(
    INPUT_DEVICE_INDEX               : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE;
    MAX_NUM_OF_SEGMENT_PICKS         : out CGI_TYPES.INTEGER_TYPE ;
    LOWER_LEFT_CORNER                : out CGI_TYPES.INPUT_SURFACE_POINT_TYPE ;
    UPPER_RIGHT_CORNER               : out CGI_TYPES.INPUT_SURFACE_POINT_TYPE ;
    PHYSICAL_INPUT_SURFACE_SIZE      : out CGI_TYPES.PHYSICAL_INPUT_SURFACE_SIZE_TYPE ;
    INPUT_SURFACE_INTERPRETATION     : out CGI_TYPES.SURFACE_INTERPRETATION_TYPE ;
    NUM_DISTINGUISHABLE_XY_STEPS     : out CGI_TYPES.XY_STEP_TYPE ) ;

  procedure INQ_STRING_CAPABILITIES(
    INPUT_DEVICE_INDEX               : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MAX_STRING_BUFFER_SIZE           : out CGI_TYPES.INTEGER_TYPE ;
    VALID_CHAR_CODING_ANNOUNCERS     : out CGI_TYPES.INTEGER_TYPE ;
    AVAIL_CHAR_CODING_ANNOUNCERS     : out CGI_TYPES.
                                           CHAR_CODING_ANNOUNCER_ARRAY_TYPE ) ;

  procedure INQ_AVAIL_INPUT_CHAR_SETS(
    INPUT_DEVICE_INDEX               : in  CGI_TYPES.INDEX_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_AVAIL_INPUT_CHAR_SETS    : out CGI_TYPES.CHAR_SET_ARRAY_TYPE ) ;

  procedure INQ_RASTER_INPUT_CAPABILITIES(
    INPUT_DEVICE_INDEX               : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    PHYSICAL_INPUT_SURFACE_SIZE      : out CGI_TYPES.PHYSICAL_INPUT_SURFACE_SIZE_TYPE ;
    COLOUR_CAPABILITY                : out CGI_TYPES.YES_NO_FLAG_TYPE ;
    NUMBER_OF_LEVELS                 : out CGI_TYPES.INTEGER_TYPE ;
    MAX_NUM_OF_BITS_PER_COLOUR       : out CGI_TYPES.INTEGER_TYPE ;
    MAX_HEIGHT                       : out CGI_TYPES.INTEGER_TYPE ;
    MAX_WIDTH                        : out CGI_TYPES.INTEGER_TYPE ;
    SPOT_CENTRE_INTERPRETATION       : out CGI_TYPES.SPOT_CENTRE_INTERPRETATION_TYPE ) ;
```

```
procedure INQ_PERMITTED_RASTER_SPOT_CENTRE_SEPARATIONS(
  INPUT_DEVICE_INDEX                          : in  CGI_TYPES.INDEX_TYPE ;
  NUM_OF_LIST_ELEMENTS_REQUESTED              : in  CGI_TYPES.INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN            : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY                           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_SPOT_CENTRE_SEPARATIONS             : out CGI_TYPES.
                                                SPOT_CENTRE_SEPARATION_TYPE ) ;

procedure INQ_GENERAL_CAPABILITIES(
  INPUT_DEVICE_INDEX      : in  CGI_TYPES.INDEX_TYPE ;
  RESPONSE_VALIDITY       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  MAX_DATA_RECORD_SIZE : out CGI_TYPES.INTEGER_TYPE ) ;

procedure INQ_SUPPORTED_GENERAL_MEASURE_FORMATS(
  INPUT_DEVICE_INDEX                          : in  CGI_TYPES.INDEX_TYPE ;
  NUM_OF_LIST_ELEMENTS_REQUESTED              : in  CGI_TYPES.INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN            : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY                           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_MEASURE_FORMAT_IDS                  : out CGI_TYPES.INTEGER_ARRAY_TYPE ) ;

procedure INQ_COMMON_LID_STATE(
  INPUT_CLASS                        : in  CGI_TYPES.INPUT_CLASS_TYPE ;
  INPUT_DEVICE_INDEX                 : in  CGI_TYPES.INDEX_TYPE ;
  RESPONSE_VALIDITY                  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  INPUT_DEVICE_STATE                 : out CGI_TYPES.INPUT_DEVICE_STATE_TYPE ;
  SAMPLE_STATE                       : out CGI_TYPES.DISABLED_ENABLED_FLAG_TYPE ;
  ECHO_CONTROL                       : out CGI_TYPES.DISABLED_ENABLED_FLAG_TYPE ;
  ECHO_TYPE                          : out CGI_TYPES.INDEX_TYPE ;
  ECHO_AREA_SPECIF_MODE              : out CGI_TYPES.VIEWPORT_SPECIF_MODE_TYPE ;
  ECHO_AREA_METRIC_SCALE_FACTOR : out CGI_TYPES.REAL_TYPE ;
  ECHO_AREA                          : out CGI_TYPES.ECHO_AREA_TYPE ;
  PROMPT_CONTROL                     : out CGI_TYPES.DISABLED_ENABLED_FLAG_TYPE ;
  PROMPT_TYPE                        : out CGI_TYPES.INDEX_TYPE ;
  ACK_CONTROL                        : out CGI_TYPES.DISABLED_ENABLED_FLAG_TYPE ;
  ACK_TYPE                           : out CGI_TYPES.INDEX_TYPE ) ;

procedure INQ_ASSOCIATED_TRIGGERS(
  INPUT_CLASS                                 : in  CGI_TYPES.INPUT_CLASS_TYPE ;
  INPUT_DEVICE_INDEX                          : in  CGI_TYPES.INDEX_TYPE ;
  NUM_OF_LIST_ELEMENTS_REQUESTED              : in  CGI_TYPES.INTEGER_TYPE ;
  INDEX_OF_FIRST_ELEMENT_TO_RETURN            : in  CGI_TYPES.INTEGER_TYPE ;
  RESPONSE_VALIDITY                           : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
  TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
  LIST_OF_ASSOCIATED_TRIGGERS                 : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

procedure INQ_ECHO_DATA_RECORD(
  INPUT_CLASS        : in  CGI_TYPES.INPUT_CLASS_TYPE ;
  INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
```

```
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    ECHO_DATA_RECORD   : out CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure INQ_INPUT_DEVICE_DATA_RECORD(
    INPUT_CLASS        : in  CGI_TYPES.INPUT_CLASS_TYPE ;
    INPUT_DEVICE_INDEX : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    DATA_RECORD        : out CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

  procedure INQ_LOCATOR_STATE(
    INPUT_DEVICE_INDEX                 : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    INPUT_EXTENT                       : out CGI_TYPES.INPUT_EXTENT_TYPE ;
    INPUT_VIEWPORT                     : out CGI_TYPES.INPUT_VIEWPORT_TYPE ;
    ECHO_VIEWPORT_SPECIF_MODE          : out CGI_TYPES.
                                             VIEWPORT_SPECIF_MODE_TYPE ;
    ECHO_VIEWPORT_METRIC_SCALE_FACTOR : out CGI_TYPES.REAL_TYPE ;
    ECHO_VIEWPORT                      : out CGI_TYPES.ECHO_AREA_TYPE ) ;

  procedure INQ_STROKE_STATE(
    INPUT_DEVICE_INDEX                 : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MAX_NUM_OF_POINTS                  : out CGI_TYPES.INTEGER_TYPE ;
    SAMPLING_INTERVAL_X_DISPLACEMENT   : out CGI_TYPES.VDC_TYPE ;
    SAMPLING_INTERVAL_Y_DISPLACEMENT   : out CGI_TYPES.VDC_TYPE ;
    MIN_TIME_INTERVAL_PER_SAMPLE       : out CGI_TYPES.REAL_TYPE ;
    MAX_TIME_INTERVAL_PER_SAMPLE       : out CGI_TYPES.REAL_TYPE ;
    INPUT_EXTENT                       : out CGI_TYPES.INPUT_EXTENT_TYPE ;
    INPUT_VIEWPORT                     : out CGI_TYPES.INPUT_VIEWPORT_TYPE ;
    ECHO_VIEWPORT_SPECIF_MODE          : out CGI_TYPES.
                                             VIEWPORT_SPECIF_MODE_TYPE ;
    ECHO_VIEWPORT_METRIC_SCALE_FACTOR : out CGI_TYPES.REAL_TYPE ;
    ECHO_VIEWPORT                      : out CGI_TYPES.ECHO_AREA_TYPE ) ;

  procedure INQ_VALUATOR_STATE(
    INPUT_DEVICE_INDEX  : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MIN_VALUE_OF_RANGE  : out CGI_TYPES.REAL_TYPE ;
    MAX_VALUE_OF_RANGE : out CGI_TYPES.REAL_TYPE ) ;

  procedure INQ_CHOICE_STATE(
    INPUT_DEVICE_INDEX                : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                 : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MAX_NUM_OF_CHOICE_ALTERNATIVES : out CGI_TYPES.INTEGER_TYPE ) ;

  procedure INQ_PICK_STATE(
    INPUT_DEVICE_INDEX                 : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                  : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    PICK_APERTURE                      : out CGI_TYPES.PICK_APERTURE_TYPE ;
    MAX_NUM_OF_SEGMENT_PICKS           : out CGI_TYPES.INTEGER_TYPE ;
```

```
    INPUT_EXTENT                              : out CGI_TYPES.INPUT_EXTENT_TYPE ;
    INPUT_VIEWPORT                            : out CGI_TYPES.INPUT_VIEWPORT_TYPE ;
    ECHO_VIEWPORT_SPECIF_MODE                 : out CGI_TYPES.
                                                    VIEWPORT_SPECIF_MODE_TYPE ;
    ECHO_VIEWPORT_METRIC_SCALE_FACTOR : out CGI_TYPES.REAL_TYPE ;
    ECHO_VIEWPORT                             : out CGI_TYPES.ECHO_AREA_TYPE ) ;

  procedure INQ_STRING_STATE(
    INPUT_DEVICE_INDEX                : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                 : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MAX_STRING_SIZE                   : out CGI_TYPES.INTEGER_TYPE ;
    INPUT_CHAR_SET_INDEX              : out CGI_TYPES.INDEX_TYPE ;
    ALTERNATE_INPUT_CHAR_SET_INDEX : out CGI_TYPES.INDEX_TYPE ;
    INPUT_CHAR_CODING_ANNOUNCER       : out CGI_TYPES.CODING_TECHNIQUE_TYPE ) ;

  procedure INQ_RASTER_INPUT_STATE(
    INPUT_DEVICE_INDEX                : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY                 : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    SPOT_CENTRE_SEPARATIONS           : out CGI_TYPES.SPOT_CENTRE_SEPARATION_TYPE ;
    COLOUR_CAPABILITY                 : out CGI_TYPES.YES_NO_FLAG_TYPE ;
    THRESHOLD_LEVEL                   : out CGI_TYPES.INTEGER_TYPE ;
    BITS_PER_COLOUR                   : out CGI_TYPES.INTEGER_TYPE ;
    SOURCE_WINDOW_FIRST_CORNER        : out CGI_TYPES.RASTER_WINDOW_CORNER_TYPE ;
    SOURCE_WINDOW_SECOND_CORNER : out CGI_TYPES.RASTER_WINDOW_CORNER_TYPE ) ;

  procedure INQ_GENERAL_STATE(
    INPUT_DEVICE_INDEX      : in  CGI_TYPES.INDEX_TYPE ;
    RESPONSE_VALIDITY       : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MAX_DATA_RECORD_SIZE : out CGI_TYPES.INTEGER_TYPE ;
    MEASURE_FORMAT_ID       : out CGI_TYPES.INTEGER_TYPE ) ;

  procedure INQ_EVENT_INPUT_STATE(
    RESPONSE_VALIDITY                 : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    EVENT_QUEUE_STATE                 : out CGI_TYPES.EVENT_QUEUE_STATE_TYPE ;
    EVENT_QUEUE_BLOCK_STATE           : out CGI_TYPES.BLOCK_CONTROL_TYPE ;
    UNREPORTED_OVERFLOW_STATE : out CGI_TYPES.UNREPORTED_OVERFLOW_STATE_TYPE ;
    OVERFLOW_DEVICE_CLASS             : out CGI_TYPES.INPUT_CLASS_TYPE ;
    OVERFLOW_DEVICE_INDEX             : out CGI_TYPES.INDEX_TYPE ;
    OVERFLOW_TRIGGER_INDEX            : out CGI_TYPES.INDEX_TYPE ;
    OVERFLOW_TIMESTAMP                : out CGI_TYPES.REAL_TYPE ;
    UNREPORTED_BREAK_STATE            : out CGI_TYPES.UNREPORTED_BREAK_STATE_TYPE ;
    BREAK_DEVICE_CLASS                : out CGI_TYPES.INPUT_CLASS_TYPE ;
    BREAK_DEVICE_INDEX                : out CGI_TYPES.INDEX_TYPE ;
    BREAK_TIMESTAMP                   : out CGI_TYPES.REAL_TYPE ) ;

  procedure INQ_ECHO_OUTPUT_CAPABILITIES(
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    MAX_NUM_OF_LOCATOR_ECHO_ENTITIES     : out CGI_TYPES.INTEGER_TYPE ;
    MAX_NUM_OF_STROKE_ECHO_ENTITIES      : out CGI_TYPES.INTEGER_TYPE ;
    MAX_NUM_OF_VALUATOR_ECHO_ENTITIES : out CGI_TYPES.INTEGER_TYPE ;
```

```
    MAX_NUM_OF_CHOICE_ECHO_ENTITIES      : out CGI_TYPES.INTEGER_TYPE ;
    MAX_NUM_OF_PICK_ECHO_ENTITIES        : out CGI_TYPES.INTEGER_TYPE ;
    MAX_NUM_OF_STRING_ECHO_ENTITIES      : out CGI_TYPES.INTEGER_TYPE ;
    MAX_NUM_OF_RASTER_ECHO_ENTITIES      : out CGI_TYPES.INTEGER_TYPE ;
    MAX_NUM_OF_GENERAL_ECHO_ENTITIES     : out CGI_TYPES.INTEGER_TYPE ;
    MAX_NUM_OF_STROKE_POINTS             : out CGI_TYPES.INTEGER_TYPE ;
    MAX_STRING_BUFFER_SIZE               : out CGI_TYPES.INTEGER_TYPE ;
    MAX_PIXEL_HEIGHT                     : out CGI_TYPES.INTEGER_TYPE ;
    MAX_PIXEL_WIDTH                      : out CGI_TYPES.INTEGER_TYPE ;
    MAX_GENERAL_INPUT_DATA_RECORD_SIZE : out CGI_TYPES.INTEGER_TYPE ) ;

  procedure INQ_ECHO_OUTPUT_ECHO_TYPES(
    INPUT_CLASS                          : in  CGI_TYPES.INPUT_CLASS_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_ECHO_TYPES_SUPPORTED         : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_ECHO_OUTPUT_PROMPT_TYPES(
    INPUT_CLASS                          : in  CGI_TYPES.INPUT_CLASS_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_PROMPT_TYPES_SUPPORTED       : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_ECHO_OUTPUT_ACK_TYPES(
    INPUT_CLASS                          : in  CGI_TYPES.INPUT_CLASS_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_ACK_TYPES_SUPPORTED          : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;

  procedure INQ_SUPPORTED_GENERAL_FORMAT_IDS(
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_MEASURE_FORMAT_IDS           : out CGI_TYPES.INTEGER_ARRAY_TYPE ) ;

  procedure INQ_CURRENTLY_EXISTING_ECHO_ENTITIES(
    INPUT_CLASS                          : in  CGI_TYPES.INPUT_CLASS_TYPE ;
    NUM_OF_LIST_ELEMENTS_REQUESTED       : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN     : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_CURRENT_INPUT_ECHO_ENTITIES  : out CGI_TYPES.INDEX_ARRAY_TYPE ) ;
```

```
    procedure INQ_ECHO_ENTITY_STATE(
      INPUT_CLASS                    : in  CGI_TYPES.INPUT_CLASS_TYPE ;
      ECHO_ENTITY_INDEX              : in  CGI_TYPES.INDEX_TYPE ;
      RESPONSE_VALIDITY              : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
      ECHO_ENTITY_STATE              : out CGI_TYPES.ECHO_ENTITY_STATE_TYPE ;
      ECHO_CONTROL                   : out CGI_TYPES.
                                           ECHO_ENTITY_ECHO_CONTROL_TYPE ;
      ECHO_TYPE                      : out CGI_TYPES.INDEX_TYPE ;
      PROMPT_CONTROL                 : out CGI_TYPES.
                                           ECHO_ENTITY_PROMPT_CONTROL_TYPE;
      PROMPT_TYPE                    : out CGI_TYPES.INDEX_TYPE ;
      ACK_TYPE                       : out CGI_TYPES.INDEX_TYPE ;
      ECHO_AREA_SPECIF_MODE          : out CGI_TYPES.VIEWPORT_SPECIF_MODE_TYPE ;
      ECHO_AREA_METRIC_SCALE_FACTOR  : out CGI_TYPES.REAL_TYPE ;
      ECHO_AREA                      : out CGI_TYPES.ECHO_AREA_TYPE ) ;

    procedure INQ_ECHO_OUTPUT_DATA_RECORD(
      INPUT_CLASS          : in  CGI_TYPES.INPUT_CLASS_TYPE ;
      ECHO_ENTITY_INDEX    : in  CGI_TYPES.INDEX_TYPE ;
      RESPONSE_VALIDITY    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
      ECHO_DATA_RECORD     : out CGI_DATA_RECORD_UTILS.DATA_RECORD_TYPE ) ;

                        ---------- CGI Raster Functions ----------

    procedure GET_NEW_BITMAP_ID(
      RESPONSE_VALIDITY    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
      BITMAP_ID            : out CGI_TYPES.BITMAP_ID_TYPE ) ;

    procedure CREATE_BITMAP(
      BITMAP_ID            : in CGI_TYPES.BITMAP_ID_TYPE ;
      BITMAP_EXTENT        : in CGI_TYPES.BITMAP_EXTENT_TYPE ;
      BITMAP_DEPTH         : in CGI_TYPES.DEPTH_TYPE ;
      BITMAP_DISPLAYABILITY : in CGI_TYPES.DISPLAYABILITY_TYPE ) ;

    procedure DELETE_BITMAP(
      BITMAP_ID : in CGI_TYPES.BITMAP_ID_TYPE ) ;

    procedure SET_DRAWING_BITMAP(
      BITMAP_ID : in CGI_TYPES.BITMAP_ID_TYPE ) ;

    procedure SET_DISPLAY_BITMAP(
      BITMAP_ID : in CGI_TYPES.BITMAP_ID_TYPE ) ;

    procedure SET_BITMAP_FOREGROUND_COLOUR(
      COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

    procedure SET_BITMAP_BACKGROUND_COLOUR(
      COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

    procedure SET_TRANSPARENT_COLOUR(
```

```
    COLOUR_SPECIFIER : in CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

  procedure SET_DRAWING_MODE(
    DRAWING_MODE : in CGI_TYPES.DRAWING_MODE_SPECIFIER_TYPE ) ;

  procedure SET_FILL_BITMAP(
    BITMAP_ID                : in CGI_TYPES.BITMAP_ID_TYPE ;
    PATTERN_BITMAP_REGION : in CGI_TYPES.BITMAP_REGION_TYPE ) ;

  procedure PIXEL_ARRAY(
    ORIGIN_POINT                          : in CGI_TYPES.VDC_POINT_TYPE ;
    X_DIMENSION                           : in CGI_TYPES.INTEGER_TYPE ;
    Y_DIMENSION                           : in CGI_TYPES.INTEGER_TYPE ;
    X_SCALE                               : in CGI_TYPES.INTEGER_TYPE ;
    Y_SCALE                               : in CGI_TYPES.INTEGER_TYPE ;
    X_DIRECTION                           : in CGI_TYPES.VDC_DIRECTION_TYPE ;
    Y_DIRECTION                           : in CGI_TYPES.VDC_DIRECTION_TYPE ;
    DRAWING_MODE                          : in CGI_TYPES.DRAWING_MODE_SPECIFIER_TYPE ;
    TRANSPARENCY                          : in CGI_TYPES.TRANSPARENCY_TYPE ;
    LOCAL_COLOUR_PREC_REQUIREMENT : in CGI_TYPES.PREC_REQUIREMENT_TYPE ;
    COLOUR_SPECIFIERS                     : in CGI_TYPES.COLOUR_ARRAY_TYPE ;
    FINAL_FLAG  : in CGI_TYPES.FINAL_FLAG_TYPE := CGI_TYPES.FINAL ) ;

  procedure GET_PIXEL_ARRAY(
    SOURCE_BITMAP_ID     : in  CGI_TYPES.BITMAP_ID_TYPE ;
    ORIGIN_POINT         : in  CGI_TYPES.VDC_POINT_TYPE ;
    X_DIMENSION          : in  CGI_TYPES.INTEGER_TYPE ;
    Y_DIMENSION          : in  CGI_TYPES.INTEGER_TYPE ;
    X_DIRECTION          : in  CGI_TYPES.VDC_DIRECTION_TYPE ;
    Y_DIRECTION          : in  CGI_TYPES.VDC_DIRECTION_TYPE ;
    LOCAL_COLOUR_PREC    : in  CGI_TYPES.PREC_REQUIREMENT_TYPE ;
    RESPONSE_VALIDITY    : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    PIXEL_VALIDITY_FLAG : out CGI_TYPES.PIXEL_VALIDITY_FLAG_TYPE ;
    VALID_X_RANGE        : out CGI_TYPES.VALID_PIXEL_RANGE_TYPE ;
    VALID_Y_RANGE        : out CGI_TYPES.VALID_PIXEL_RANGE_TYPE ;
    COLOUR_SPECIFIERS    : out CGI_TYPES.COLOUR_ARRAY_TYPE ) ;

  procedure GET_PIXEL_ARRAY_DIMS(
    SOURCE_BITMAP_ID                      : in  CGI_TYPES.BITMAP_ID_TYPE ;
    REGION                                : in  CGI_TYPES.BITMAP_REGION_TYPE ;
    LOCAL_COLOUR_PREC_REQUIREMENT : in  CGI_TYPES.PREC_REQUIREMENT_TYPE ;
    RESPONSE_VALIDITY                     : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    LOCAL_COLOUR_PREC                     : out CGI_TYPES.PREC_REQUIREMENT_TYPE ;
    X_DIMENSION                           : out CGI_TYPES.INTEGER_TYPE ;
    Y_DIMENSION                           : out CGI_TYPES.INTEGER_TYPE ) ;

  procedure SOURCE_DESTINATION_BITBLT(
    SOURCE_BITMAP_ID    : in CGI_TYPES.BITMAP_ID_TYPE ;
    SOURCE_ORIGIN       : in CGI_TYPES.VDC_POINT_TYPE ;
    DESTINATION_ORIGIN : in CGI_TYPES.VDC_POINT_TYPE ;
```

```
    X_OFFSET                : in CGI_TYPES.VDC_TYPE ;
    Y_OFFSET                : in CGI_TYPES.VDC_TYPE ;
    DRAWING_MODE            : in CGI_TYPES.DRAWING_MODE_SPECIFIER_TYPE ;
    TRANSPARENCY            : in CGI_TYPES.TRANSPARENCY_TYPE ) ;

  procedure TILE_3_OPERAND_BITBLT(
    PATTERN_BITMAP_ID       : in CGI_TYPES.BITMAP_ID_TYPE ;
    PATTERN_REGION          : in CGI_TYPES.BITMAP_REGION_TYPE ;
    REFERENCE_POINT         : in CGI_TYPES.VDC_POINT_TYPE ;
    SOURCE_BITMAP_ID        : in CGI_TYPES.BITMAP_ID_TYPE ;
    SOURCE_ORIGIN           : in CGI_TYPES.VDC_POINT_TYPE ;
    DESTINATION_ORIGIN      : in CGI_TYPES.VDC_POINT_TYPE ;
    X_OFFSET                : in CGI_TYPES.VDC_TYPE ;
    Y_OFFSET                : in CGI_TYPES.VDC_TYPE ;
    DRAWING_MODE_3          : in CGI_TYPES.DRAWING_MODE_SPECIFIER_TYPE ;
    TRANSPARENCY            : in CGI_TYPES.TRANSPARENCY_TYPE ) ;

  procedure INQ_RASTER_CAPABILITY(
    RESPONSE_VALIDITY                          : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    NUM_OF_PREDEFINED_DISPLAYABLE_BITMAPS      : out CGI_TYPES.INTEGER_TYPE ;
    DISPLAYABLE_BITMAP_CREATION_SUPPORT        : out CGI_TYPES.YES_NO_FLAG_TYPE ;
    BITMAP_FORMATS_SUPPORTED                   : out CGI_TYPES.DEPTHS_SUPPORTED_TYPE ;
    NUM_OF_BITS_PER_FULL_DEPTH_PIXEL           : out CGI_TYPES.INTEGER_TYPE ;
    DRAWING_MODE_TRANSPARENCY_SUPPORT          : out CGI_TYPES.
                                                     DRAWING_MODE_SUPPORT_TYPE ;
    DRAWING_MODE_3_TRANSPARENCY_SUPPORT        : out CGI_TYPES.
                                                     DRAWING_MODE_SUPPORT_TYPE ;
    NUM_OF_SUPPORTED_BITMAP_MODES              : out CGI_TYPES.INTEGER_TYPE ;
    ARRAY_OF_SUPPORTED_BITMAP_MODES            : out CGI_TYPES.
                                                     BITMAP_MODE_ARRAY_TYPE ;
    SIZE_OF_PIXEL                              : out CGI_TYPES.PIXEL_DIMENSION_TYPE ;
    PREFERRED_BITBLT_PATTERN_SIZE              : out CGI_TYPES.BITBLT_DIMENSION_TYPE ;
    SOURCE_BITMAP_TRUNCATION_CAPABILITY        : out CGI_TYPES.
                                                     BITMAP_TRUNCATION_TYPE ;
    PREVIOUS_DISPLAY_BITMAP_DATA               : out CGI_TYPES.
                                                     DISPLAY_BITMAP_DATA_TYPE ) ;

  procedure INQ_SUPPORTED_DRAWING_MODE_TRANSPARENCY_PAIRS(
    NUM_OF_LIST_ELEMENTS_REQUESTED      : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_SUPPORTED_PAIRS : out CGI_TYPES.DRAWING_MODE_TRANSPARENCY_ARRAY_TYPE
  ) ;

  procedure INQ_SUPPORTED_DRAWING_MODE_3_TRANSPARENCY_PAIRS(
    NUM_OF_LIST_ELEMENTS_REQUESTED      : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN    : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                   : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_DESCRIPTION_TABLE : out CGI_TYPES.INTEGER_TYPE ;
```

```
    LIST_OF_SUPPORTED_PAIRS : out CGI_TYPES.DRAWING_MODE_TRANSPARENCY_ARRAY_TYPE
  ) ;

  procedure INQ_RASTER_STATE(
    RESPONSE_VALIDITY                          : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    DISPLAY_BITMAP_ID                          : out CGI_TYPES.BITMAP_ID_TYPE ;
    DRAWING_BITMAP_ID                          : out CGI_TYPES.BITMAP_ID_TYPE ;
    DRAWING_MODE                               : out CGI_TYPES.
                                                     DRAWING_MODE_SPECIFIER_TYPE ;
    MAPPED_FOREGROUND_COLOUR_SELECTION         : out CGI_TYPES.
                                                     COLOUR_SELECTION_MODE_TYPE ;
    MAPPED_FOREGROUND_COLOUR                   : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ;
    MAPPED_BACKGROUND_COLOUR_SELECTION         : out CGI_TYPES.
                                                     COLOUR_SELECTION_MODE_TYPE ;
    MAPPED_BACKGROUND_COLOUR                   : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ;
    TRANSPARENT_COLOUR_SELECTION               : out CGI_TYPES.
                                                     COLOUR_SELECTION_MODE_TYPE ;
    TRANSPARENT_FOREGROUND_COLOUR              : out CGI_TYPES.COLOUR_SPECIFIER_TYPE ) ;

  procedure INQ_NON_DISPLAYABLE_BITMAP_IDS(
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_STATE_LIST     : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_BITMAP_IDS               : out CGI_TYPES.BITMAP_ID_ARRAY_TYPE ) ;

  procedure INQ_DISPLAYABLE_BITMAP_IDS(
    NUM_OF_LIST_ELEMENTS_REQUESTED   : in  CGI_TYPES.INTEGER_TYPE ;
    INDEX_OF_FIRST_ELEMENT_TO_RETURN : in  CGI_TYPES.INTEGER_TYPE ;
    RESPONSE_VALIDITY                : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    TOTAL_ELEMENTS_IN_STATE_LIST     : out CGI_TYPES.INTEGER_TYPE ;
    LIST_OF_BITMAP_IDS               : out CGI_TYPES.BITMAP_ID_ARRAY_TYPE ) ;

  procedure INQ_BITMAP_STATE(
    BITMAP_ID                  : in  CGI_TYPES.BITMAP_ID_TYPE ;
    RESPONSE_VALIDITY          : out CGI_TYPES.VALIDITY_FLAG_TYPE ;
    DEPTH_TYPE                 : out CGI_TYPES.DEPTH_TYPE ;
    DISPLAYABILITY             : out CGI_TYPES.DISPLAYABILITY_TYPE ;
    BITMAP_MODE                : out CGI_TYPES.BITMAP_MODE_TYPE ;
    BOTTOM_LEFT_PIXEL          : out CGI_TYPES.DEVICE_POINT_TYPE ;
    TOP_RIGHT_PIXEL            : out CGI_TYPES.DEVICE_POINT_TYPE ;
    VDC_EXTENT                 : out CGI_TYPES.VDC_EXTENT_TYPE ;
    ISOTROPY                   : out CGI_TYPES.ISOTROPY_FLAG_TYPE ;
    HOR_ALIGNMENT              : out CGI_TYPES.HOR_ALIGNMENT_FLAG_TYPE ;
    VERT_ALIGNMENT             : out CGI_TYPES.VERT_ALIGNMENT_FLAG_TYPE ;
    VIEWPORT_SPECIF_MODE       : out CGI_TYPES.VIEWPORT_SPECIF_MODE_TYPE ;
    VIEWPORT_METRIC_SCALE_FACTOR : out CGI_TYPES.REAL_TYPE ;
    REQUESTED_DEVICE_VIEWPORT  : out CGI_TYPES.DEVICE_VIEWPORT_TYPE ;
    EFFECTIVE_VIEWPORT         : out CGI_TYPES.DEVICE_VIEWPORT_TYPE ;
    SURFACE_CLIP_INDICATOR     : out CGI_TYPES.
```

```
                                        DRAWING_SURFACE_CLIP_INDICATOR_TYPE ;
    DSCRECT                           : out CGI_TYPES.DEVICE_VIEWPORT_TYPE ;
    DSCRECT_SPECIF_MODE               : out CGI_TYPES.VIEWPORT_SPECIF_MODE_TYPE ;
    DSCRECT_METRIC_SCALE_FACTOR       : out CGI_TYPES.REAL_TYPE ) ;

end CGI ;
```

A.5 Package specification CGI_PROFILE_ID_CONST

```
with CGI_TYPES ;

package CGI_PROFILE_ID_CONST is

   -- Package: CGI_PROFILE_ID_CONST
   -- Functional Description:
   --
   -- This package provides the definition of the profiles listed in the ISO/IEC 9636 (Part 1, Annex B).
   --
   -- The following constants define the various CGI profiles as specified in the ISO/IEC 9636
   -- (Part 1 Annex B). The profiles are declared exactly as they appear in the annex (no additional
   -- acronyms have been used), with the exception of underscores being used for word delimiters. The
   -- profile identifiers are declared in this way in order to aid in the ease of use for the application
   -- programmer.
   --
   --  Example
   --
   --    Annex B : LOCATOR INPUT
   --    Constant : LOCATOR_INPUT
   --
   -- Since Ada does not allow numeric data to denote the first character of a variable name, the
   -- following literals (identifiers) were altered as shown:
   --
   --  Example
   --
   --    Annex B : 1 WAY OUTPUT, 2 WAY OUTPUT
   --    Constant : ONE_WAY_OUTPUT, TWO_WAY_OUTPUT

   TWO_WAY_OUTPUT          : constant CGI_TYPES.PROFILE_ID_TYPE := 1 ;
   ONE_WAY_OUTPUT          : constant CGI_TYPES.PROFILE_ID_TYPE := 2 ;
   LOCATOR_INPUT           : constant CGI_TYPES.PROFILE_ID_TYPE := 3 ;
   STROKE_INPUT            : constant CGI_TYPES.PROFILE_ID_TYPE := 4 ;
   VALUATOR_INPUT          : constant CGI_TYPES.PROFILE_ID_TYPE := 5 ;
   CHOICE_INPUT            : constant CGI_TYPES.PROFILE_ID_TYPE := 6 ;
   STRING_INPUT            : constant CGI_TYPES.PROFILE_ID_TYPE := 7 ;
   RASTER_INPUT            : constant CGI_TYPES.PROFILE_ID_TYPE := 8 ;
   GKS_OUTPUT_0            : constant CGI_TYPES.PROFILE_ID_TYPE := 9 ;
   GKS_OUTPUT_1            : constant CGI_TYPES.PROFILE_ID_TYPE := 10 ;
   GKS_INPUT_B             : constant CGI_TYPES.PROFILE_ID_TYPE := 11 ;
   GKS_INPUT_C             : constant CGI_TYPES.PROFILE_ID_TYPE := 12 ;
   GKS_OUTIN_0B            : constant CGI_TYPES.PROFILE_ID_TYPE := 13 ;
   GKS_OUTIN_0C            : constant CGI_TYPES.PROFILE_ID_TYPE := 14 ;
   GKS_OUTIN_1B            : constant CGI_TYPES.PROFILE_ID_TYPE := 15 ;
   GKS_OUTIN_1C            : constant CGI_TYPES.PROFILE_ID_TYPE := 16 ;
   BASIC_CGM               : constant CGI_TYPES.PROFILE_ID_TYPE := 17 ;
   ADVANCED_1_WAY_OUTPUT   : constant CGI_TYPES.PROFILE_ID_TYPE := 18 ;
```

```
end CGI_PROFILE_ID_CONST ;
```

A.6 Package specification CGI_FUNCTION_ID_CONST

```
with CGI_TYPES ;

package CGI_FUNCTION_ID_CONST is

   -- Package: CGI_FUNCTION_ID_CONST
   -- Functional Description:
   --
   -- This package provides the definition of the function identifiers listed in the ISO/IEC 9636
   -- (Part 1, Annex A).
   --
   -- The following constants define the various CGI functions as specified in the ISO/IEC 9636
   -- (Part 1 Annex A). The functions are declared exactly as they appear in the annex (no
   -- additional acronyms have been used), with the exception of underscores being used for word
   -- delimiters. The functions identifiers are declared in this way in order to aid in the ease of use
   -- for the application programmer.
   --
   --  Example
   --
   --    Annex A : EXECUTE DEFERRED ACTIONS
   --    Constant : EXECUTE_DEFERRED_ACTIONS
   --
   -- Since Ada programming restrictions will not allow the use of the reserved word terminate as a
   -- variable name, the following function identifiers have been changed.
   --
   --    Annex A : INITIALIZE
   --    Constant : INITIALIZE_CGI_SESSION
   --
   --    Annex A : TERMINATE
   --    Constant : TERMINATE_CGI_SESSION

   INITIALIZE_CGI_SESSION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 1 ;

   TERMINATE_CGI_SESSION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 2 ;

   EXECUTE_DEFERRED_ACTIONS
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 3 ;

   DEFERRAL_MODE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 4 ;

   PREPARE_DRAWING_SURFACE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 5 ;

   END_PAGE
```

```
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 6 ;

  VDC_TYPE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 7 ;

  VDC_INTEGER_PRECISION_REQUIREMENTS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 8 ;

  VDC_REAL_PRECISION_REQUIREMENTS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 9 ;

  VDC_EXTENT
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 10 ;

  DEVICE_VIEWPORT
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 11 ;

  DEVICE_VIEWPORT_SPECIFICATION_MODE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 12 ;

  DEVICE_VIEWPORT_MAPPING
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 13 ;

  DRAWING_SURFACE_CLIP_RECTANGLE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 14 ;

  DRAWING_SURFACE_CLIP_INDICATOR
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 15 ;

  DEQUEUE_ERROR_REPORTS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 16 ;

  ERROR_HANDLING_CONTROL
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 17 ;

  INTEGER_PRECISION_REQUIREMENT
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 18 ;

  REAL_PRECISION_REQUIREMENTS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 19 ;

  INDEX_PRECISION_REQUIREMENT
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 20 ;

  COLOUR_PRECISION_REQUIREMENT
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 21 ;

  COLOUR_INDEX_PRECISION_REQUIREMENT
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 22 ;
```

```
    CLIENT_SPECIFIED_NAME_PRECISION_REQUIREMENT
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 23 ;

    MESSAGE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 24 ;

    ESCAPE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 25 ;

    GET_ESCAPE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 26 ;

    STATE_LIST_INQUIRY_SOURCE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 27 ;

    INQUIRE_DEVICE_IDENTIFICATION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 28 ;

    INQUIRE_DEVICE_DESCRIPTION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 29 ;

    LOOKUP_FUNCTION_SUPPORT
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 30 ;

    LOOKUP_PROFILE_SUPPORT
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 31 ;

    INQUIRE_LIST_OF_PROFILE_SUPPORT_INDICATORS
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 32 ;

    INQUIRE_SUPPORTED_VDC_TYPES
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 33 ;

    INQUIRE_DEVICE_CONTROL_CAPABILITY
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 34 ;

    LOOKUP_ESCAPE_SUPPORT
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 35 ;

    LOOKUP_GET_ESCAPE_SUPPORT
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 36 ;

    INQUIRE_CONTROL_STATE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 37 ;

    INQUIRE_CURRENT_PRECISION_REQUIREMENTS
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 38 ;

    INQUIRE_VDC_TO_DEVICE_MAPPING
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 39 ;
```

```
    INQUIRE_ERROR_HANDLING
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 40 ;

    INQUIRE_MISCELLANEOUS_CONTROL_STATE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 41 ;

    POLYLINE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 42 ;

    DISJOINT_POLYLINE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 43 ;

    CIRCULAR_ARC_3_POINT
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 44 ;

    CIRCULAR_ARC_CENTRE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 45 ;

    CIRCULAR_ARC_CENTRE_REVERSED
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 46 ;

    ELLIPTICAL_ARC
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 47 ;

    CONNECTING_EDGE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 48 ;

    POLYMARKER
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 49 ;

    TEXT
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 50 ;

    RESTRICTED_TEXT
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 51 ;

    APPEND_TEXT
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 52 ;

    POLYGON
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 53 ;

    POLYGON_SET
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 54 ;

    RECTANGLE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 55 ;

    CIRCLE
```

```
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 56 ;

CIRCULAR_ARC_3_POINT_CLOSE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 57 ;

CIRCULAR_ARC_CENTRE_CLOSE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 58 ;

ELLIPSE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 59 ;

ELLIPTICAL_ARC_CLOSE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 60 ;

CELL_ARRAY
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 61 ;

GENERALIZED_DRAWING_PRIMITIVE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 62 ;

LINE_BUNDLE_INDEX
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 63 ;

LINE_TYPE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 64 ;

LINE_WIDTH
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 65 ;

LINE_COLOUR
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 66 ;

LINE_CLIPPING_MODE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 67 ;

MARKER_BUNDLE_INDEX
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 68 ;

MARKER_TYPE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 69 ;

MARKER_SIZE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 70 ;

MARKER_COLOUR
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 71 ;

MARKER_CLIPPING_MODE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 72 ;
```

```
    TEXT_BUNDLE_INDEX
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 73 ;

    TEXT_FONT_INDEX
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 74 ;

    TEXT_PRECISION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 75 ;

    CHARACTER_EXPANSION_FACTOR
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 76 ;

    CHARACTER_SPACING
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 77 ;

    TEXT_COLOUR
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 78 ;

    CHARACTER_HEIGHT
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 79 ;


    CHARACTER_ORIENTATION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 80 ;

    TEXT_PATH
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 81 ;

    TEXT_ALIGNMENT
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 82 ;

    CHARACTER_SET_INDEX
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 83 ;

    ALTERNATE_CHARACTER_SET_INDEX
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 84 ;

    CHARACTER_CODING_ANNOUNCER
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 85 ;

    FILL_BUNDLE_INDEX
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 86 ;

    INTERIOR_STYLE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 87 ;

    FILL_COLOUR
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 88 ;

    HATCH_INDEX
```

```
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 89 ;

  PATTERN_INDEX
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 90 ;

  FILL_REFERENCE_POINT
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 91 ;

  PATTERN_SIZE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 92 ;

  EDGE_BUNDLE_INDEX
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 93 ;

  EDGE_TYPE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 94 ;

  EDGE_WIDTH
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 95 ;

  EDGE_COLOUR
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 96 ;

  EDGE_CLIPPING_MODE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 97 ;

  EDGE_VISIBILITY
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 98 ;

  CLIP_INDICATOR
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 99 ;

  CLIP_RECTANGLE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 100 ;

  LINE_WIDTH_SPECIFICATION_MODE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 101 ;

  EDGE_WIDTH_SPECIFICATION_MODE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 102 ;

  MARKER_SIZE_SPECIFICATION_MODE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 103 ;

  COLOUR_SELECTION_MODE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 104 ;

  COLOUR_VALUE_EXTENT
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 105 ;
```

```
    BACKGROUND_COLOUR
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 106 ;

    AUXILIARY_COLOUR
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 107 ;

    TRANSPARENCY
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 108 ;

    COLOUR_TABLE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 109 ;

    LINE_REPRESENTATION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 110 ;

    MARKER_REPRESENTATION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 111 ;

    TEXT_REPRESENTATION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 112 ;

    FILL_REPRESENTATION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 113 ;

    EDGE_REPRESENTATION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 114 ;

    DELETE_BUNDLE_REPRESENTATION
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 115 ;

    ASPECT_SOURCE_FLAGS
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 116 ;

    PATTERN_TABLE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 117 ;

    DELETE_PATTERN
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 118 ;

    FONT_LIST
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 119 ;

    CHARACTER_SET_LIST
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 120 ;

    SAVE_PRIMITIVE_ATTRIBUTES
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 121 ;

    RESTORE_PRIMITIVE_ATTRIBUTES
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 122 ;
```

```
DELETE_PRIMITIVE_ATTRIBUTE_SAVE_SET
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 123 ;

BEGIN_FIGURE
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 124 ;

END_FIGURE
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 125 ;

NEW_REGION
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 126 ;

GET_TEXT_EXTENT
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 127 ;

INQUIRE_PRIMITIVE_SUPPORT_LEVELS
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 128 ;

LOOKUP_GDP_SUPPORT
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 129 ;

INQUIRE_GDP_ATTRIBUTES
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 130 ;

INQUIRE_LINE_CAPABILITY
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 131 ;

INQUIRE_LIST_OF_AVAILABLE_LINE_TYPES
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 132 ;

INQUIRE_LIST_OF_AVAILABLE_SCALED_LINE_WIDTHS
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 133 ;

INQUIRE_MARKER_CAPABILITY
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 134 ;

INQUIRE_LIST_OF_AVAILABLE_MARKER_TYPES
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 135 ;

INQUIRE_LIST_OF_AVAILABLE_SCALED_MARKER_SIZES
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 136 ;

INQUIRE_TEXT_CAPABILITY
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 137 ;

INQUIRE_LIST_OF_AVAILABLE_CHARACTER_SETS
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 138 ;

INQUIRE_LIST_OF_AVAILABLE_TEXT_FONTS
```

```
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 139 ;

  INQUIRE_FONT_CAPABILITIES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 140 ;

  INQUIRE_LIST_OF_AVAILABLE_CHARACTER_EXPANSION_FACTORS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 141 ;

  INQUIRE_LIST_OF_AVAILABLE_CHARACTER_SPACINGS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 142 ;

  INQUIRE_LIST_OF_AVAILABLE_CHARACTER_HEIGHTS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 143 ;

  INQUIRE_LIST_OF_AVAILABLE_CHARACTER_ORIENTATIONS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 144 ;

  INQUIRE_FILL_CAPABILITY
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 145 ;

  INQUIRE_LIST_OF_AVAILABLE_HATCH_STYLES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 146 ;

  INQUIRE_EDGE_CAPABILITY
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 147 ;

  INQUIRE_LIST_OF_AVAILABLE_EDGE_TYPES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 148 ;

  INQUIRE_LIST_OF_AVAILABLE_SCALED_EDGE_WIDTHS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 149 ;

  INQUIRE_COLOUR_CAPABILITY
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 150 ;

  INQUIRE_CIE_CHARACTERISTICS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 151 ;

  INQUIRE_MAXIMUM_NUMBER_OF_SIMULTANEOUSLY_SAVED_ATTRIBUTE_SETS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 152 ;

  INQUIRE_ARRAY_OF_SUPPORTED_CHARACTER_CODING_ANNOUNCERS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 153 ;

  INQUIRE_LINE_ATTRIBUTES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 154 ;

  INQUIRE_LIST_OF_LINE_BUNDLE_INDICES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 155 ;
```

```
INQUIRE_LINE_REPRESENTATION
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 156 ;

INQUIRE_MARKER_ATTRIBUTES
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 157 ;

INQUIRE_LIST_OF_MARKER_BUNDLE_INDICES
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 158 ;

INQUIRE_MARKER_REPRESENTATION
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 159 ;

INQUIRE_TEXT_ATTRIBUTES
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 160 ;

INQUIRE_LIST_OF_TEXT_BUNDLE_INDICES
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 161 ;

INQUIRE_TEXT_REPRESENTATION
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 162 ;

INQUIRE_FILL_ATTRIBUTES
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 163 ;

INQUIRE_PATTERN_DIMENSIONS
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 164 ;

INQUIRE_PATTERN
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 165 ;

INQUIRE_LIST_OF_PATTERN_INDICES
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 166 ;

INQUIRE_LIST_OF_FILL_BUNDLE_INDICES
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 167 ;

INQUIRE_FILL_REPRESENTATION
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 168 ;

INQUIRE_EDGE_ATTRIBUTES
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 169 ;

INQUIRE_LIST_OF_EDGE_BUNDLE_INDICES
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 170 ;

INQUIRE_EDGE_REPRESENTATION
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 171 ;

INQUIRE_OUTPUT_STATE
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 172 ;
```

```
INQUIRE_OBJECT_CLIPPING
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 173 ;

INQUIRE_LIST_OF_ATTRIBUTE_SET_NAMES_IN_USE
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 174 ;

INQUIRE_COLOUR_STATE
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 175 ;

INQUIRE_LIST_OF_COLOUR_TABLE_ENTRIES
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 176 ;

INQUIRE_FONT_LIST
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 177 ;

INQUIRE_CHARACTER_SET_LIST
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 178 ;

LOOKUP_ASPECT_SOURCE_FLAGS
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 179 ;

GET_NEW_SEGMENT_IDENTIFIER
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 180 ;

CREATE_SEGMENT
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 181 ;

REOPEN_SEGMENT
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 182 ;

CLOSE_SEGMENT
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 183 ;

COPY_SEGMENT
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 184 ;

DELETE_SEGMENT
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 185 ;

DELETE_ALL_SEGMENTS
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 186 ;

RENAME_SEGMENT
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 187 ;

DRAW_ALL_SEGMENTS
  : constant CGI_TYPES.FUNCTION_ID_TYPE := 188 ;

IMPLICIT_SEGMENT_REGENERATION_MODE
```

```
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 189 ;

  RESET_REGENERATION_PENDING
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 190 ;

  PICK_IDENTIFIER
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 191 ;

  SEGMENT_VISIBILITY
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 192 ;

  SEGMENT_TRANSFORMATION
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 193 ;

  SEGMENT_HIGHLIGHTING
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 194 ;

  SEGMENT_DISPLAY_PRIORITY
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 195 ;

  SEGMENT_DETECTABILITY
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 196 ;

  SEGMENT_PICK_PRIORITY
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 197 ;

  SIMULATE_PICK
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 198 ;

  INHERITANCE_FILTER
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 199 ;

  CLIPPING_INHERITANCE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 200 ;

  INQUIRE_SEGMENT_CAPABILITY
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 201 ;

  INQUIRE_SEGMENT_STATE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 202 ;

  INQUIRE_LIST_OF_INHERITANCE_FILTER_SETTINGS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 203 ;

  INQUIRE_CLIPPING_INHERITANCE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 204 ;

  INQUIRE_LIST_OF_SEGMENT_IDENTIFIERS_IN_USE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 205 ;
```

```
    INQUIRE_INDIVIDUAL_SEGMENT_STATE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 206 ;

    INITIALIZE_LOGICAL_INPUT_DEVICE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 207 ;

    RELEASE_LOGICAL_INPUT_DEVICE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 208 ;

    ECHO_CONTROLS
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 209 ;

    PUT_CURRENT_LOCATOR_MEASURE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 210 ;

    PUT_CURRENT_STROKE_MEASURE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 211 ;

    PUT_CURRENT_VALUATOR_MEASURE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 212 ;

    PUT_CURRENT_CHOICE_MEASURE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 213 ;

    PUT_CURRENT_PICK_MEASURE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 214 ;

    PUT_CURRENT_STRING_MEASURE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 215 ;

    PUT_CURRENT_RASTER_MEASURE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 216 ;

    PUT_CURRENT_GENERAL_MEASURE
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 217 ;

    ECHO_DATA
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 218 ;

    LOCATOR_DEVICE_DATA
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 219 ;

    STROKE_DEVICE_DATA
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 220 ;

    VALUATOR_DEVICE_DATA
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 221 ;

    CHOICE_DEVICE_DATA
      : constant CGI_TYPES.FUNCTION_ID_TYPE := 222 ;
```

```
PICK_DEVICE_DATA
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 223 ;

STRING_DEVICE_DATA
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 224 ;

RASTER_DEVICE_DATA
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 225 ;

GENERAL_DEVICE_DATA
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 226 ;

ASSOCIATE_TRIGGERS
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 227 ;

GET_ADDITIONAL_STROKE_DATA
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 228 ;

GET_ADDITIONAL_PICK_DATA
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 229 ;

GET_ADDITIONAL_STRING_DATA
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 230 ;

GET_ADDITIONAL_RASTER_DATA
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 231 ;

REQUEST_LOCATOR
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 232 ;

REQUEST_STROKE
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 233 ;

REQUEST_VALUATOR
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 234 ;

REQUEST_CHOICE
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 235 ;

REQUEST_PICK
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 236 ;

REQUEST_STRING
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 237 ;

REQUEST_RASTER
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 238 ;

REQUEST_GENERAL
```

```
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 239 ;

  SAMPLING_STATE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 240 ;

  SAMPLE_LOCATOR
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 241 ;

  SAMPLE_STROKE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 242 ;

  SAMPLE_VALUATOR
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 243 ;

  SAMPLE_CHOICE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 244 ;

  SAMPLE_PICK
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 245 ;

  SAMPLE_STRING
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 246 ;

  SAMPLE_RASTER
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 247 ;

  SAMPLE_GENERAL
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 248 ;

  INITIALIZE_ECHO_REQUEST
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 249 ;

  ECHO_REQUEST_LOCATOR
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 250 ;

  ECHO_REQUEST_STROKE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 251 ;

  ECHO_REQUEST_VALUATOR
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 252 ;

  ECHO_REQUEST_CHOICE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 253 ;

  ECHO_REQUEST_PICK
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 254 ;

  ECHO_REQUEST_STRING
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 255 ;
```

```
ECHO_REQUEST_RASTER
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 256 ;

ECHO_REQUEST_GENERAL
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 257 ;

INITIALIZE_EVENT_QUEUE
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 258 ;

RELEASE_EVENT_QUEUE
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 259 ;

ENABLE_EVENTS
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 260 ;

DISABLE_EVENTS
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 261 ;

EVENT_QUEUE_BLOCK_CONTROL
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 262 ;

FLUSH_EVENTS
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 263 ;

FLUSH_DEVICE_EVENTS
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 264 ;

AWAIT_EVENT
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 265 ;

DEQUEUE_LOCATOR_EVENT
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 266 ;

DEQUEUE_STROKE_EVENT
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 267 ;

DEQUEUE_VALUATOR_EVENT
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 268 ;

DEQUEUE_CHOICE_EVENT
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 269 ;

DEQUEUE_PICK_EVENT
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 270 ;

DEQUEUE_STRING_EVENT
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 271 ;

DEQUEUE_RASTER_EVENT
   : constant CGI_TYPES.FUNCTION_ID_TYPE := 272 ;
```

```
   DEQUEUE_GENERAL_EVENT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 273 ;

   EVENT_QUEUE_TRANSFER
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 274 ;

   INITIALIZE_ECHO_OUTPUT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 275 ;

   RELEASE_ECHO_OUTPUT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 276 ;

   ECHO_OUTPUT_CONTROLS
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 277 ;

   PERFORM_ACKNOWLEDGEMENT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 278 ;

   UPDATE_LOCATOR_ECHO_OUTPUT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 279 ;

   UPDATE_STROKE_ECHO_OUTPUT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 280 ;

   UPDATE_VALUATOR_ECHO_OUTPUT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 281 ;

   UPDATE_CHOICE_ECHO_OUTPUT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 282 ;

   UPDATE_PICK_ECHO_OUTPUT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 283 ;

   UPDATE_STRING_ECHO_OUTPUT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 284 ;

   UPDATE_RASTER_ECHO_OUTPUT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 285 ;

   UPDATE_GENERAL_ECHO_OUTPUT
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 286 ;

   ECHO_OUTPUT_DATA
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 287 ;

   INQUIRE_INPUT_CAPABILITY
     : constant CGI_TYPES.FUNCTION_ID_TYPE := 288 ;

   INQUIRE_LIST_OF_AVAILABLE_INPUT_DEVICES
```

```
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 289 ;

  INQUIRE_COMMON_INPUT_DEVICE_PROPERTIES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 290 ;

  INQUIRE_LIST_OF_SUPPORTED_ECHO_TYPES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 291 ;

  INQUIRE_LIST_OF_SUPPORTED_PROMPT_TYPES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 292 ;

  INQUIRE_LIST_OF_SUPPORTED_ACKNOWLEDGEMENT_TYPES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 293 ;

  INQUIRE_LIST_OF_ASSOCIABLE_TRIGGERS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 294 ;

  INQUIRE_LOCATOR_CAPABILITIES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 295 ;

  INQUIRE_STROKE_CAPABILITIES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 296 ;

  INQUIRE_CHOICE_CAPABILITIES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 297 ;

  INQUIRE_PICK_CAPABILITIES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 298 ;

  INQUIRE_STRING_CAPABILITIES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 299 ;

  INQUIRE_LIST_OF_AVAILABLE_INPUT_CHARACTER_SETS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 300 ;

  INQUIRE_RASTER_INPUT_CAPABILITIES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 301 ;

  INQUIRE_LIST_OF_PERMITTED_RASTER_SPOT_CENTRE_SEPARATIONS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 302 ;

  INQUIRE_GENERAL_CAPABILITIES
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 303 ;

  INQUIRE_LIST_OF_SUPPORTED_GENERAL_MEASURE_FORMATS
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 304 ;

  INQUIRE_COMMON_LOGICAL_INPUT_DEVICE_STATE
    : constant CGI_TYPES.FUNCTION_ID_TYPE := 305 ;
```