

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
AVA Documents
Document Sharing Hub

Contents

Foreword	iv
Introduction	v
1 Scope	1
2 Normative references	2
3 Principles	3
3.1 Conformance	3
3.2 Implications of the language	3
3.2.1 Functional mapping	3
3.2.2 Implementation and host dependencies	5
3.2.3 Error handling	5
3.2.4 Continuation of functions	7
3.2.5 Packed data formats	8
3.2.6 Events and event report lists	8
3.2.7 Data mapping	9
3.2.8 Multi-tasking	11
3.2.9 Packaging	11
3.2.10 Client program environment	13
3.2.11 Registration	13
4 Tables	14
4.1 Abbreviations used in the Ada language binding	14
4.2 Abbreviation policy in construction of identifiers	14
4.3 CGI function names	15
4.3.1 Alphabetical by bound name	15
4.3.2 Alphabetical by CGI function name	21
5 CGI configuration values	28
6 Type definitions	34
6.1 Array index ranging	35
6.2 Representation of CGI basic data types	36
6.3 Representation of CGI strings	42
6.4 Representation of CGI data records	44
6.5 Representation of CGI abstract data types	45
6.6 Representation of CGI enumerated data types	53

© ISO/IEC 1994

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

6.7 CGI Ada record types	67
6.8 CGI Ada subtypes	77
6.9 CGI Ada array types	78
6.10 CGI Ada access types	91
6.11 CGI exceptions	92
7 CGI/Ada functions	93
7.1 Part 2 control functions	93
7.2 Part 3 output functions	100
7.3 Part 4 segment functions	125
7.4 Part 5 input functions	130
7.5 Part 6 raster functions	155
7.6 Binding defined utility functions	160
7.6.1 Data record utilities	160
7.6.1.1 Data record utility constants	160
7.6.1.2 Data record utility types	163
7.6.1.3 Data record utility functions	164
7.6.2 String utilities	174
7.6.2.1 String utility functions	174
7.6.3 Error handling utilities	175
7.6.3.1 Error handling utility functions	175
7.6.4 Data packing utilities	177
7.6.4.1 Data packing utility types	177
7.6.4.2 Data packing utility functions	178
Annex A	181
A.1 Package specification CGI_CONFIG	182
A.2 Package specification CGI_TYPES	187
A.3 Package specification CGI_DATA_RECORD_UTILS	218
A.4 Package specification CGI	225
A.5 Package specification CGI_PROFILE_ID_CONST	271
A.6 Package specification CGI_FUNCTION_ID_CONST	273
A.7 Package specification CGI_REGISTRATION_CONST	295
A.8 Package specification CGI_ERROR_CONST	297
A.9 Package specification CGI_STRING_UTILS	310
A.10 Package specification CGI_ERROR_HANDLING_UTILS	311
A.11 Package specification CGI_PACKING_UTILS	312
Annex B	315
B.1 Example Program 1: Star	315
B.2 Example Program 2: Name Object	319
B.3 Example Program 3: Text	328
B.4 Example Program 4: Load CGI Database	331
B.4 Example Program 5: Event Queue Pkg	335
Annex C	340
Annex D	346
Annex E	354

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 succession 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 handling 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

Tables

15

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
 CONNECTING EDGE
 COPY SEGMENT
 CREATE BITMAP
 CREATE SEGMENT
 DEFERRAL MODE
 DELETE ALL SEGMENTS
 DELETE BITMAP
 DELETE BUNDLE REPRESENTATION
 DELETE PATTERN
 DELETE PRIMITIVE ATTRIBUTE SAVE SET
 DELETE SEGMENT
 DEQUEUE CHOICE EVENT
 DEQUEUE ERROR REPORTS
 DEQUEUE GENERAL EVENT
 DEQUEUE LOCATOR EVENT
 DEQUEUE PICK EVENT
 DEQUEUE RASTER EVENT
 DEQUEUE STRING EVENT
 DEQUEUE STROKE EVENT
 DEQUEUE VALUATOR EVENT
 DEVICE VIEWPORT
 DEVICE VIEWPORT MAPPING
 DEVICE VIEWPORT SPECIFICATION MODE
 DISABLE EVENTS
 DISJOINT POLYLINE
 DISPLAY BITMAP
 DRAW ALL SEGMENTS
 DRAWING BITMAP
 DRAWING MODE
 DRAWING SURFACE CLIP INDICATOR
 DRAWING SURFACE CLIP RECTANGLE
 ECHO CONTROLS
 ECHO DATA
 ECHO OUTPUT CONTROLS
 ECHO OUTPUT DATA
 ECHO REQUEST CHOICE
 ECHO REQUEST GENERAL
 ECHO REQUEST LOCATOR
 ECHO REQUEST PICK
 ECHO REQUEST RASTER
 ECHO REQUEST STRING
 ECHO REQUEST STROKE
 ECHO REQUEST VALUATOR
 EDGE BUNDLE INDEX
 EDGE CLIPPING MODE
 EDGE COLOUR
 EDGE REPRESENTATION
 EDGE TYPE
 EDGE VISIBILITY
 EDGE WIDTH
 EDGE WIDTH SPECIFICATION MODE
 ELLIPSE
 ELLIPTICAL ARC
 ELLIPTICAL ARC CLOSE
 ENABLE EVENTS
 END FIGURE
 END PAGE
 ERROR HANDLING CONTROL
 ESCAPE

set_colour_value_extent
 connecting_edge
 copy_segment
 create_bitmap
 create_segment
 set_deferral_mode
 delete_all_segments
 delete_bitmap
 delete_bundle_rep
 delete_pattern
 delete_primitive_attribute_save_set
 delete_segment
 dequeue_choice_event
 dequeue_error_reports
 dequeue_general_event
 dequeue_locator_event
 dequeue_pick_event
 dequeue_raster_event
 dequeue_string_event
 dequeue_stroke_event
 dequeue_valuator_event
 set_device_viewport
 set_device_viewport_mapping
 set_device_viewport_specif_mode
 disable_events
 disjoint_polyline
 display_bitmap
 draw_all_segments
 set_drawing_bitmap
 set_drawing_mode
 set_drawing_surface_clip_indicator
 set_drawing_surface_clip_rectangle
 set_echo_controls
 set_echo_data
 set_echo_output_controls
 set_echo_output_data
 echo_request_choice
 echo_request_general
 echo_request_locator
 echo_request_pick
 echo_request_raster
 echo_request_string
 echo_request_stroke
 echo_request_valuator
 set_edge_bundle_index
 set_edge_clipping_mode
 set_edge_colour
 set_edge_rep
 set_edge_type
 set_edge_visibility
 set_edge_width
 set_edge_width_specif_mode
 ellipse
 elliptical_arc
 elliptical_arc_close
 enable_events
 end_figure
 end_page
 set_error_handling_control
 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
 INQUIRE GDP ATTRIBUTES
 INQUIRE GENERAL CAPABILITIES
 INQUIRE GENERAL STATE
 INQUIRE INDIVIDUAL SEGMENT STATE
 INQUIRE INPUT CAPABILITY
 INQUIRE INPUT DEVICE DATA RECORD
 INQUIRE LINE ATTRIBUTES
 INQUIRE LINE CAPABILITY
 INQUIRE LINE REPRESENTATION
 INQUIRE LIST OF ASSOCIABLE TRIGGERS
 INQUIRE LIST OF ASSOCIATED TRIGGERS
 INQUIRE LIST OF ATTRIBUTE SET NAMES IN USE
 INQUIRE LIST OF AVAILABLE CHARACTER EXPANSION FACTORS
 INQUIRE LIST OF AVAILABLE CHARACTER HEIGHTS
 INQUIRE LIST OF AVAILABLE CHARACTER ORIENTATIONS
 INQUIRE LIST OF AVAILABLE CHARACTER SETS
 INQUIRE LIST OF AVAILABLE CHARACTER SPACINGS
 INQUIRE LIST OF AVAILABLE EDGE TYPES
 INQUIRE LIST OF AVAILABLE HATCH STYLES
 INQUIRE LIST OF AVAILABLE INPUT CHARACTER SETS
 INQUIRE LIST OF AVAILABLE INPUT DEVICES
 INQUIRE LIST OF AVAILABLE LINE TYPES
 INQUIRE LIST OF AVAILABLE MARKER TYPES
 INQUIRE LIST OF AVAILABLE SCALED EDGE WIDTHS
 INQUIRE LIST OF AVAILABLE SCALED LINE WIDTHS
 INQUIRE LIST OF AVAILABLE SCALED MARKER SIZES
 INQUIRE LIST OF AVAILABLE TEXT FONTS
 INQUIRE LIST OF COLOUR TABLE ENTRIES
 INQUIRE LIST OF CURRENTLY EXISTING ECHO ENTITIES
 INQUIRE LIST OF DISPLAYABLE BITMAP IDENTIFIERS
 INQUIRE LIST OF ECHO OUTPUT ACKNOWLEDGEMENT TYPES
 INQUIRE LIST OF ECHO OUTPUT ECHO TYPES
 INQUIRE LIST OF ECHO OUTPUT PROMPT TYPES
 INQUIRE LIST OF EDGE BUNDLE INDICES
 INQUIRE LIST OF FILL BUNDLE INDICES
 INQUIRE LIST OF INHERITANCE FILTER SETTINGS
 INQUIRE LIST OF LINE BUNDLE INDICES
 INQUIRE LIST OF MARKER BUNDLE INDICES
 INQUIRE LIST OF NON DISPLAYABLE BITMAP IDENTIFIERS
 INQUIRE LIST OF PATTERN INDICES
 INQUIRE LIST OF PERMITTED RASTER SPOT CENTRE SEPARATION
 INQUIRE LIST OF PROFILE SUPPORT INDICATORS
 INQUIRE LIST OF SEGMENT IDENTIFIERS IN USE
 INQUIRE LIST OF SUPPORTED ACKNOWLEDGEMENT TYPES
 INQUIRE LIST OF SUPPORTED DRAWING MODE 3 TRANSPARENCY PAIRS
 INQUIRE LIST OF SUPPORTED DRAWING MODE TRANSPARENCY PAIRS
 INQUIRE LIST OF SUPPORTED ECHO TYPES
 INQUIRE LIST OF SUPPORTED GENERAL FORMAT IDENTIFIERS
 INQUIRE LIST OF SUPPORTED GENERAL MEASURE FORMATS
 INQUIRE LIST OF SUPPORTED PROMPT TYPES
 INQUIRE LIST OF TEXT BUNDLE INDICES
 INQUIRE LOCATOR CAPABILITIES
 INQUIRE LOCATOR STATE
 INQUIRE MARKER ATTRIBUTES
 INQUIRE MARKER CAPABILITY
 INQUIRE MARKER REPRESENTATION
 INQUIRE MAXIMUM NUMBER OF SIMULTANEOUSLY SAVED ATTRIBUTE SETS
 INQUIRE MISCELLANEOUS CONTROL STATE
 INQUIRE OBJECT CLIPPING

inq_font_list
 inq_gdp_attributes
 inq_general_capabilities
 inq_general_state
 inq_indiv_segment_state
 inq_input_capability
 inq_input_device_data_record
 inq_line_attributes
 inq_line_capability
 inq_line_rep
 inq_associable_triggers
 inq_associated_triggers
 inq_attribute_set_names_in_use
 inq_avail_char_expan_factors
 inq_avail_char_heights
 inq_avail_char_orientations
 inq_avail_char_sets
 inq_avail_char_spacings
 inq_avail_edge_types
 inq_avail_hatch_styles
 inq_avail_input_char_sets
 inq_avail_input_devices
 inq_avail_line_types
 inq_avail_marker_types
 inq_avail_scaled_edge_widths
 inq_avail_scaled_line_widths
 inq_avail_scaled_marker_sizes
 inq_avail_text_fonts
 inq_colour_table_entries
 inq_currently_existing_echo_entities
 inq_displayable_bitmap_ids
 inq_echo_output_ack_types
 inq_echo_output_echo_types
 inq_echo_output_prompt_types
 inq_edge_bundle_indices
 inq_fill_bundle_indices
 inq_inheritance_filter_settings
 inq_line_bundle_indices
 inq_marker_bundle_indices
 inq_non_displayable_bitmap_ids
 inq_pattern_indices
 inq_permitted_raster_spot_centre_separations
 inq_profile_support_indicators
 inq_segment_ids_in_use
 inq_supported_ack_types
 inq_supported_drawing_mode_3_transparency_pairs
 inq_supported_drawing_mode_transparency_pairs
 inq_supported_echo_types
 inq_supported_general_format_ids
 inq_supported_general_measure_formats
 inq_supported_prompt_types
 inq_text_bundle_indices
 inq_locator_capabilities
 inq_locator_state
 inq_marker_attributes
 inq_marker_capability
 inq_marker_rep
 inq_max_simultaneous_attribute_sets
 inq_miscellaneous_control_state
 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
UPDATE CHOICE ECHO OUTPUT
UPDATE GENERAL ECHO OUTPUT
UPDATE LOCATOR ECHO OUTPUT
UPDATE PICK ECHO OUTPUT
UPDATE RASTER ECHO OUTPUT
UPDATE STRING ECHO OUTPUT
UPDATE STROKE ECHO OUTPUT
UPDATE VALUATOR ECHO OUTPUT
VALUATOR DEVICE DATA
VDC EXTENT
VDC INTEGER PRECISION REQUIREMENTS
VDC REAL PRECISION REQUIREMENTS
VDC TYPE

set_transparent_colour
update_choice_echo_output
update_general_echo_output
update_locator_echo_output
update_pick_echo_output
update_raster_echo_output
update_string_echo_output
update_stroke_echo_output
update_valuator_echo_output
set_valuator_device_data
set_vdc_extent
set_vdc_integer_prec_requirements
set_vdc_real_prec_requirements
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.

CGI Basic Data Types

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
