INTERNATIONAL STANDARD

ISO/IEC 9638-3

First edition
1994-12-15

Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Language bindings — Part 3: Ada

Technologies de l'information — Infographie — Techniques interfaciales pour dialogues avec dispositifs graphiques (CGI) — Liants de langage — Partie 3: Ada

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

Contents

Foreword

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

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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

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

- *Part 1: FORTRAN*
- *Part 2: PASCAL*
- *Part 3: Ada*

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

Introduction

The Computer Graphics Interface (CGI) (ISO/IEC 9636) is specified in a language independent manner and needs to be embedded in language dependent layers (language bindings) for use with particular programming languages.

The purpose of this document is to define a standard binding of CGI to the Ada computer programming language.

This page intentionally left blank

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

Part 3:
Ada

1 Scope

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

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9638. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9638 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 8652:1987, *Programming Languages - Ada.*

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

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

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

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

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

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

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

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

3 Principles

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

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

3.1 Conformance

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

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

- The semantics of an implementation shall be those stated in the CGI standard as modified or extended for Ada as stated in this binding document.
- The package(s) corresponding to CGI shall be an available Ada library unit, with all names as specified by this document or as modified for one or more CGI profiles.

3.2 Implications of the language

3.2.1 Functional mapping

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

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

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

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

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

ISO/IEC 9636 Function : Request <input class>

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

ISO/IEC 9636 Function : Sample <input class>

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

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

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

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

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

Dequeue Raster Event
Dequeue General Event

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

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

3.2.2 Implementation and host dependencies

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

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

3.2.3 Error handling

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

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

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

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

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

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

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

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

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

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

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

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

3.2.4 Continuation of functions

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

The functions which have been bound in this manner are:

Cell Array
Generalized Drawing Primitive
Pixel Array
Polygon
Polyline

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

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

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

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

3.2.5 Packed data formats

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

Cell Array
Pixel Array
Colour Array

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

3.2.6 Events and event report lists

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

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

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

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


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

3.2.7 Data mapping

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

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

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