INTERNATIONAL STANDARD

ISO/IEC 9636-4

First edition
1991-12-15

Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification —

Part 4: Segments

Technologies de l'information — Infographie — Interfaces pour l'infographie — Spécifications fonctionnelles —

Partie 4: Segments

Reference number
ISO/IEC 9636-4:1991(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 9636-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology.*

ISO/IEC 9636 consists of the following parts, under the general title *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification*:

- *Part 1: Overview, profiles, and conformance*
- *Part 2: Control*
- *Part 3: Output*
- *Part 4: Segments*
- *Part 5: Input and echoing*
- *Part 6: Raster*

Annexes A and B form an integral part of this part of ISO/IEC 9636. Annexes C and D are for information only.

Introduction

This part of ISO/IEC 9636 describes the functions of the Computer Graphics Interface concerned with segment storage.

The functionality incorporated in this part of ISO/IEC 9636 is concerned with creating, modifying, and manipulating graphic pictures using segments.

The functionality described in this part of ISO/IEC 9636 pertains to Virtual Devices of class OUTPUT and OUTIN.

This page intentionally left blank

Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices (CGI) – Functional specification –

Part 4:
Segments

1 Scope

This part of ISO/IEC 9636 defines those functions of the Computer Graphics Interface concerned with the creation, modification, and manipulation of graphic pictures using segments.

This part of ISO/IEC 9636 is part 4 of ISO/IEC 9636, and should be read in conjunction with ISO/IEC 9636-1, ISO/IEC 9636-2, and ISO/IEC 9636-3. The relationship of this part of ISO/IEC 9636 to the other parts of ISO/IEC 9636 is described in ISO/IEC 9636-1 and in clause 4.

The functionality described in this part of ISO/IEC 9636 pertains to Virtual Devices of class OUTPUT and OUTIN.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9636. 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 9636 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 7942 : 1985 *Information processing systems – Computer graphics – Graphical Kernel System (GKS) functional description.*

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

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

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

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

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

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

ISO/IEC 9637-2 : -[1)] *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding — Part 2: Binary encoding.*

1) To be published.

3 Concepts

3.1 Introduction

This part of ISO/IEC 9636 specifies how graphic objects may be grouped in segments and be identified by a unique segment identifier. The graphic objects stored in segments are composed of graphic primitives with associated attribute values. This part of ISO/IEC 9636 defines a set of functions for creating, modifying, and manipulating segments. This functionality is divided into the following areas:

- *Segment manipulation functions*, including segment creation, deletion, renaming, and copying. Graphic objects may also be appended to the end of an existing segment. Graphic objects within a segment, however, cannot be modified or deleted.
- *Segment attribute functions*, including modification of segment attributes (e.g. transformation, visibility). Some segment attributes affect the rendered appearance of segments and can be a basis for feedback during graphic picture manipulation. Other attributes affect how input concepts and segments are associated in support of pick input.
- *Segment inquiry functions*, by which access is provided to the information in the description tables and state lists concerned with segments.

3.1.1 Relationship of CGI segments to the graphic output pipeline

The overview of the CGI Graphic Object Pipeline presented in ISO/IEC 9636-1, clause 5, describes the relationship between segments and the pipeline. Additional details are described in this part of ISO/IEC 9636 in connection with the particular functional areas that are affected.

3.2 Creating segments

3.2.1 Segment identifiers

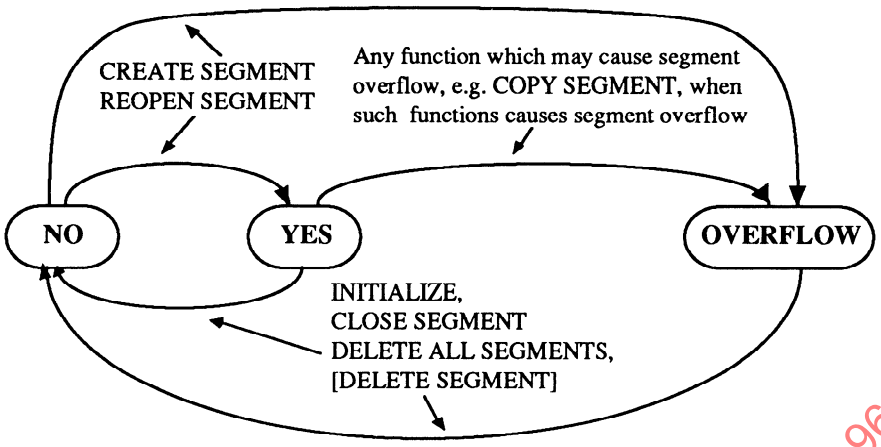
Each segment has a unique segment identifier associated with it. The client can refer to a specific segment by using its associated segment identifier. Once a segment is deleted, the identifier that was associated with it may be reused for another segment definition. The segment identifier associated with a segment may be changed with the RENAME SEGMENT function. The Maximum Number of Simultaneously Existing Segments entry in the Segment Description Table indicates the number of segments which may coexist at one time (this number is implementation-dependent). The List of Segment Identifiers in use may be inquired from the Segment State List.

The client provides a segment identifier as a parameter to the CREATE SEGMENT function used to create a new segment. For cases in which the client does not wish to keep a record of which identifiers are available, the function GET NEW SEGMENT IDENTIFIER is provided. This function will return an available segment identifier. A segment identifier is a parameter of most of the functions defined in this part of ISO/IEC 9636.

3.2.2 Creating and closing segments

Segments are created when the CREATE SEGMENT function is invoked. This is the only means of creating a segment. A segment is opened by CREATE SEGMENT and the Segment Open State entry in the Segment State List is set to YES. There can be at most one segment open at any one time. While a segment is open, graphic objects passed along the Graphic Object Pipeline are stored in the segment (a process termed segment definition) and rendered on the drawing surface if the segment's Visibility attribute is VISIBLE. The open segment is closed by invoking the function CLOSE SEGMENT and the Segment Open State entry in the Segment State List is set to NO (see figure 1).

An existing segment may be reopened at a later time with the REOPEN SEGMENT function. When reopened, subsequent graphic objects are appended to the open segment using the same conceptual mechanism as when the segment was created. A reopened segment is closed by invoking the function CLOSE SEGMENT.



NOTE – DELETE SEGMENT will change the state from YES or OVERFLOW to NO only if the currently open segment is deleted.

Figure 1 – Segment Open state transitions

Note that objects stored in segments have associated with them all applicable individual attribute values independent of the value of the corresponding ASF value. Thus by changing the ASF (by means of INHERITANCE FILTER, see 3.5), it is still possible to display the object with the individual attribute value associated on creation even when the corresponding ASF value was initially BUNDLED.

3.2.3 Non-retained data

Any graphic object passing along the pipeline when there is no segment open is termed non-retained data. Such graphic objects are rendered onto the drawing surface in the usual manner as described in ISO/IEC 9636-3. However, non-retained data is never rendered again as a result of implicit or explicit regeneration.

3.2.4 Segment storage overflow

Segment storage overflow may occur when storing graphic objects in segments. If overflow occurs, all graphic objects prior to the first object that causes the overflow are stored in the segment. The graphic object causing the overflow as well as graphic objects following the segment storage overflow and prior to segment closure are rendered as non-retained data and a class 6 error is generated.

Once an overflow has occurred, the Segment Open State in the Segment State List is set to OVERFLOW. Invoking CLOSE SEGMENT, DELETE SEGMENT, or DELETE ALL SEGMENTS is the only way to change the Segment Open State from OVERFLOW to NO (other than by INITIALIZE) (see figure 1).

Segment storage overflow may also occur when a segment is created or reopened. In this case, the CREATE SEGMENT or REOPEN SEGMENT function is ignored, a class 6 error is generated, and graphic objects following the segment storage overflow and before the segment is closed are rendered as non-retained data.

3.3 Segment attributes

3.3.1 Introduction

The segment attribute values associated with a particular segment affect that segment's rendering and pick input properties. A segment's attribute values, including Segment Transformation, are set by default when the segment is initially created and can be changed individually at any time while the segment still exists. Segment attribute values may be inquired from the Individual Segment State Lists.

The rendering of graphic objects within a segment while the segment is being defined depends on the segment's attributes of Visibility, Highlighting, Display Priority, and Segment Transformation, and the current setting of the Implicit Segment Regeneration Mode. For example, if Visibility is set to INVISIBLE before the creation of the first graphic object to be stored in the segment, the segment is not rendered. If Visibility is changed part way through the segment definition and the Implicit Segment Regeneration Mode is SUPPRESSED, each new object will be rendered as it is received without regard to the Display Priority attribute.

3.3.2 Segment highlighting

The Highlighting attribute of a segment can take the values NORMAL or HIGHLIGHTED. The form in which highlighting is represented is implementation-dependent. However, the appearance of a segment when highlighted shall be different from when it is not. When a segment's highlighting attribute is changed, all of the graphic objects of the segment are displayed based on the Implicit Segment Regeneration Mode and the segment's Display Priority. (See 3.4.)

3.3.3 Segment visibility

The Visibility attribute of a segment can take the value of INVISIBLE or VISIBLE. When a segment's Visibility attribute is set to be VISIBLE, all of the graphic objects of the segment are displayed based upon the Implicit Segment Regeneration Mode and the segment's Display Priority. A segment can be picked if and only if it is both VISIBLE and DETECTABLE. If segment storage overflow occurs, objects following the segment storage overflow and before the segment is closed will be rendered as non-retained data independent of the value of the Visibility attribute.

3.3.4 Segment detectability

The Detectability attribute of a segment can take the value of UNDETECTABLE or DETECTABLE. Segment detectability does not affect the display or appearance of segments. A segment can be picked if and only if it is both VISIBLE and DETECTABLE.

3.3.5 Segment display priority

The Display Priority attribute of a segment determines how overlapping segments are displayed. In general, segments with higher display priorities will be displayed as if they were in front of segments with lower display priorities. This part of ISO/IEC 9636 allows latitude in cases where overlapping segments have the same display priority. The preferred behaviour is as follows:

> The effective relative display priority of a set of overlapping segments having the same display priority value is consistent with their relative creation time order. In particular, the most recently created segments will appear in front of segments created earlier (this is consistent with the effect of drawing the same graphic objects as non-retained data in the same order as the segments were created). Reopening a segment does not change the effective relative display priority of the segment.

The Effective Relative Display Priority entry in the Segment Description Table indicates the supported capability, as follows:

- A level of TIME ORDER indicates that the most recently created segment is displayed in front, as described above;
- A level of TIME ORDER - REOPEN is similar to TIME ORDER, except that a change in the effective relative display priority is possible for a segment if subsequently reopened;
- A level of OTHER indicates that there is no guaranteed effective relative display priority behaviour.

The SEGMENT DISPLAY PRIORITY function is used to change the display priority of a segment. When a segment's Display Priority is changed, the segment's display is based on the Implicit Segment Regeneration Mode and the new Display Priority. (See 3.4.)

3.3.6 Segment pick priority

The Pick Priority attribute of a segment is used to resolve the picking of segments which overlap. If two or more segments overlap and the pick location is within the intersection of these segments, the segments picked will be one, or more, of those with the highest pick priority. The returned list of pick values (see 5.4.1 and PICK input in ISO/IEC 9636-5) corresponds to all those segments having the highest identical pick priority and are returned in order of decreasing display priority.

3.3.7 Segment transformation

The Segment Transformation attribute specifies a coordinate transformation to be applied to the graphic objects contained in the segment. This transformation allows scaling, translation, and rotation of segments. The segment transformation is a transformation of VDC space to VDC space and is distinct from the VDC-to-Device Mapping (see ISO/IEC 9636-2, 3.3), which is a transformation of VDC space to DC space.

The Segment Transformation attribute is set by the SEGMENT TRANSFORMATION function. This function simply replaces the Segment Transformation attribute associated with a particular segment. A segment transformation is composed of a 2x2 scaling and rotation portion and a 2x1 translation portion. The default Segment Transformation for each segment is the identity transformation.

The Segment Transformation attribute is associated with the graphic objects of a segment when they enter into the graphic object pipeline. This associated transformation is then applied during the rendering of the graphic objects.

A graphic object is transformed by the transformation resulting from the concatenation of the segment transformation and the object's transformation attribute before the application of the VDC-to-Device Mapping. The method of concatenation is such that the final effect during rendering is as if the object's transformation attribute was applied first, followed by application of the segment transformation. This transformation is applied to the graphic object's VDC point data and all associated attributes having VDC paramaterizations, with the exception of the associated clip rectangle.

NOTE – If the client wishes to accumulate or concatenate transformations for the purpose of setting the Segment Transformation attribute, it is necessary to do this above the CGI. The use of segment transformations may produce coordinates that cannot be expressed within the VDC range; this is handled in an implementation-dependent way.

3.4 Segment display

3.4.1 Introduction

Segment display is the process which produces a visible image on a drawing surface from the graphic objects in a segment. The functions COPY SEGMENT and DRAW ALL SEGMENTS provide for the display of segments, individually and collectively. Segments may also be displayed implicitly, under some circumstances, without using the above functions (see 3.4.2).

In addition to the attributes associated with the graphic objects stored in segments, the Highlighting, Visibility, Display Priority, and Segment Transformation attributes of segments affect the appearance of the displayed segment.

Some devices cannot immediately change a picture. A plotter for example can only add to a picture; the plotter would need to advance the paper and redraw the picture to show the effect of a change of Segment Transformation. Similarly, a change of a segment's Display Priority attribute might cause different sections of a picture to become visible or obscured, and Visibility and Highlighting attributes may also cause the picture to alter.

On devices which cannot erase or change part of a picture, Implicit Segment Regeneration Mode of SUPPRESSED is more efficient in terms of time and material. Implicit Segment Regeneration Mode in combination with the Visibility attribute gives the client the ability to accumulate picture changes. At some later time, the client can explicitly utilize PREPARE DRAWING SURFACE and DRAW ALL SEGMENTS.

3.4.2 Segment regeneration

Conceptually, the contents of segment storage, along with the values of various state list and bundle table entries, describe a certain picture. In addition to the stored graphic objects, the state list information which may affect the appearance of the picture is shown in table 1.

Segment creation or deletion, as well as dynamic changes to any of the state list entries in table 1, may affect the accuracy of the rendered picture (i.e. whether or not the current, rendered picture representation accurately reflects the picture described in segment storage and state lists). The process of bringing the rendered picture representation to a state consistent with the current contents of segment storage and state lists is referred to as *regeneration*.

Table 1 – State list information which may affect the appearance of the picture

VDC-to-Device Mapping	*Segment Attributes:*
Colour table entries	Display Priority
Background Colour	Visibility
Bundled primitive attribute values (depending on associated ASFs)	Highlighting
	Segment Transformation
Pattern table entries	
List of Font Names	

When there is a discrepancy between the current contents of the drawing surface and the described picture, a *regeneration is pending*. The Segment State List includes a Regeneration Pending entry, which indicates whether or not any such discrepancy currently exists. Changes to the picture, or to the state list information that controls the appearance of the picture (see table 1) may cause the Regeneration Pending entry to be set to YES. If there is any visible picture discrepancy, the implementation shall set the entry to YES. The implementation may also set the entry to YES in case that a change effects no visible discrepancy, but the implementation is not able to determine that there is no discrepancy. Once set, Regeneration Pending remains set until an implicit regeneration or until the function RESET REGENERATION PENDING has been invoked.

There are a number of CGI functions which may change the contents of the drawing surface in such a way that it no longer accurately reflects the contents of segment storage, but do not cause Regeneration Pending to be set. This set of functions includes COPY SEGMENT when there is no segment open and all graphic primitive functions executed when there is no segment open (thereby creating non-retained data). It also includes bitblts, PIXEL ARRAY, DRAWING BITMAP, and PREPARE DRAWING SURFACE. It is the client's responsibility to be aware of the implications of the use of these functions in conjunction with segments and handle any desired regeneration explicitly.

For performance reasons, it may be preferable to batch a number of picture changes and then perform a single regeneration. A means to control regeneration behaviour is provided by the Implicit Segment Regeneration Mode entry in the Segment State List. When Implicit Segment Regeneration Mode is SUPPRESSED, the Virtual Device will not perform any regeneration even if one is pending.

For particular devices, some changes may not necessitate a regeneration. An important (and common) example concerns changes to the Colour Table on a raster device. Typically, such devices implement a colour table in hardware. Changing the colour representation associated with a given colour index immediately (and retroactively) affects all pixels drawn with that index. No regeneration process is required to bring the picture up to date.

For each state list entry corresponding to information in table 1, a Dynamic Modification Accepted For entry in a description table indicates which changes

- can be performed immediately (IMM);
- can be simulated (CBS) (see 3.4.3);
- lead to an implicit regeneration (IRG).

If the Dynamic Modification Accepted For <type of change> entry is IMM, the effect of a change to the relevant state list information will propagate immediately to the picture without any regeneration.

If the Dynamic Modification Accepted For <type of change> entry is IRG and Implicit Segment Regeneration Mode is SUPPRESSED, the effect of a change to the relevant state list information will be to set Regeneration Pending to YES but no regeneration will be initiated.

If the Dynamic Modification Accepted For <type of change> entry is IRG and Implicit Segment Regeneration Mode is ALLOWED, any change of the relevant state list information will initiate the regeneration process immediately after the function that caused the change of state list information.

If the Dynamic Modification Accepted For Adding Objects To Open Segment entry in the Segment Description Table is IRG and the Implicit Segment Regeneration Mode is SUPPRESSED, then each object added to the open segment will be rendered as it is received without regard to the Display Priority attribute value; this will cause the Regeneration Pending entry to be set to YES if there is any visible picture discrepancy. If the Implicit Segment Regeneration Mode is ALLOWED, then each object added to the open segment shall be rendered with the correct display priority (relative to possibly overlapping segments of higher display priority).

Other operations, beside changing the state list entries, may cause a discrepancy between the described picture and the drawing surface. These include deleting a segment and adding additional graphic objects to an open segment. The Segment Description Table has Dynamic Modification Accepted For flags for these changes. The distinction is also made between Visibility changes from visible to invisible and from invisible to visible.

If a regeneration is pending at the time of a transition of the Implicit Segment Regeneration Mode to ALLOWED, then a side effect of the IMPLICIT SEGMENT REGENERATION MODE function is to cause a regeneration to occur at that time.

For any function that causes a regeneration to occur, the regeneration process occurs after execution of the function and subsequent functions will not be interpreted until after the effects of the regeneration have been realized. Conceptually, the effect of a regeneration may be described as follows:

> The drawing surface is prepared (cleared), and copies of all of the graphic objects in all the visible segments in segment storage are inserted into the Graphic Object Pipeline as described in ISO/IEC 9636, and the Regeneration Pending entry is set to NO.

All non-retained data is erased if a regeneration occurs. The prediction of regeneration requires careful attention to the Dynamic Modification Accepted For flags, which may differ from one CGI implementation to another.

Regeneration should not be confused with Deferral Mode which controls an entirely different aspect of Virtual Device behaviour. One of the more important manifestations of deferral relates to buffering that may occur between calling a CGI function at a procedural interface and subsequent interpretation of that function at a data stream interpreter further downstream in the implementation. Thus deferral is concerned with the promptness with which function invocations will be acted upon. Regeneration effects never occur until the causing functions are actually acted upon.

3.4.3 Quick update methods

When segments are complex or there are many of them, regeneration may be a fairly time consuming process. The CGI provides a way to allow implementations to offer quick update methods for changes that would otherwise require a regeneration. Examples might include changes to segment attributes, such as a change in Segment Transformation, Visibility, Highlighting or Display Priority. Quick Update Methods may be used to gain performance in interactive situations at the risk of the picture no longer accurately reflecting the state of segment storage.

For example, a segment may be "erased" by drawing it in the background colour. In cases where this operation is feasible, performance might be significantly improved, especially when dealing with a large number of segments. Many clients may accept a compromise in picture accuracy in favour of greater interactiveness. At some appropriate later time (perhaps, upon operator request), the client can regenerate an uncompromised picture using DRAW ALL SEGMENTS.

The above type of picture repair is referred to as using a *quick update method*, as opposed to *regeneration*. Quick update methods are not claimed to be faithful to the segment model, but under many circumstances, are guaranteed to be faster. For many applications, there may be circumstances under which the client can know that the quick update will actually be perfect. For example, if the client never draws any overlapping segments.

Changes which have implementation defined quick update methods are indicated by a Dynamic Modification Accepted For <type of change> flag of value CBS. This indicates that the effect of such a change can be simulated. Any change which can be simulated can also be regenerated.

Quick update behaviour is permitted to occur if the Implicit Segment Regeneration Mode is set to UQUM, (for Use Quick Update Method) and the Dynamic Modification Accepted For <type of change> flag is CBS. If Implicit Segment Regeneration Mode is set to UQUM, regeneration remains suppressed.

The only point in time when a quick update can occur is when the function causing the need for it is invoked. If Implicit Segment Regeneration Mode is SUPPRESSED, then quick update methods are also suppressed. If the Implicit Segment Regeneration Mode is either SUPPRESSED or UQUM and a change is made for which Dynamic Modification Accepted For entry is CBS, then Regeneration Pending will be set to YES. An exception to this may occur when a particular implementation knows that the picture repair using quick update was, in fact, faithful to the segment model and state lists.

If the Implicit Segment Regeneration Mode is ALLOWED and a change is made for which the Dynamic Modification Accepted For flag is CBS or IRG, then an implicit segment regeneration will be performed as a result of the function causing the change.

Although control of the use of quick update methods is provided for in ISO/IEC 9636-4, the actual behaviour of an instance of a quick update is implementation-dependent.

3.4.4 Explicit segment display

The CGI functions COPY SEGMENT and DRAW ALL SEGMENTS may be used to explicitly display segments. DRAW ALL SEGMENTS displays all visible segments in a manner consistent with their display priorities. Its operation is independent of the Regeneration Pending state list entry and does not affect the setting of this entry. A single segment may be displayed by invoking the COPY SEGMENT function with the identity transformation as well as the INHERITANCE FILTER function with the filter selection list set to ALL and the selection setting set to SEGMENT. This segment will be rendered in front of any already rendered segments with no regard to display priority.

3.5 Copy segment and the inheritance filter

The COPY SEGMENT function copies the graphic objects of an identified source segment into the Graphic Object Pipeline, concatenating a coordinate transformation specified by the copy transformation parameter and optionally concatenating the source segment's Segment Transformation attribute with the transformation associated with the copied objects. The copy transformation affects all associated clip rectangles; the source segment's Segment Transformation does not affect clip rectangles.

The copied graphic objects may be altered in a variety of ways:

a) The inheritance filter controls whether individual attribute values of the graphic objects are re-associated from the CGI state lists (see 5.4.2).

b) The CLIPPING INHERITANCE function controls whether the effective clip region associated with the object is ignored or intersected with the Clip Rectangle in the General Attributes and Output Control State List.

c) A copy transformation is concatenated with the graphic objects' associated transformation and, optionally, the Segment Transformation of the source segment may also be concatenated according to the rules for transformations (see 3.3.7).

If the Segment Open State is NO when the COPY SEGMENT function is interpreted, then the copied objects are treated as non-retained data and the associated transformations are applied during rendering. If the Segment Open State is YES when COPY SEGMENT is interpreted, the group of objects from the source segment with their associated copy and (optional) source segment transformations are added to the open segment. The Segment Transformation attribute value of the open segment also affects the eventual rendering of the group of objects copied. Transformations associated during a copy segment operation are object attribute values. They do not affect the open segment, only the group of objects copied. These transformations are never dissociated during subsequent copy segment operations regardless of parameter values or inheritance filter settings. The replacement, transformation concatenation, and intersection of clip rectangles according to the setting of the inheritance filter is discussed in 5.2.5.

The client specifies a copy transformation parameter with the invocation of COPY SEGMENT. The copy transformation is associated with the graphic objects as they are copied and it affects any associated clip rectangles. The transformation may translate, scale, rotate, and skew, though some implementations may not support rotation or skewing of clip rectangles – whether this is supported or not is indicated by the Transform Clip Region Effectiveness entry in the Segment Description Table.

The client also specifies whether the source segment's Segment Transformation attribute value is also to be concatenated with the copied objects' associated transformation. If the source segment's Segment Transformation attribute value is concatenated, then the resulting association is such that when the transformations are subsequently applied, the segment transformation is applied before the copy transformation. In general, they are not commutative.

NOTE – Repeated copying may occur, in which case, multiple pairs of source segment and copy transformations may be applied to the objects. These will be applied in the order from the first copy through to the last copy.

Examples of the use of the COPY SEGMENT and INHERITANCE FILTER functions are given in annex D.

3.6 Delete and rename segments

The DELETE SEGMENT function is used to delete an individual segment. The function DELETE ALL SEGMENTS deletes all segments. Segment identifiers for segments which have been deleted are available for reuse.

If the open segment is to be deleted, this segment is implicitly closed before the deleting operation is performed. The Dynamic Modification Accepted For Segment Deletion entry in the Segment Description Table indicates whether an implicit regeneration is required on deleting a segment. When a segment is deleted, the Individual Segment State List associated with the segment is deleted, thereby removing all associated segment attributes.

The function RENAME SEGMENT may be used to rename an existing segment at any time. When a segment is renamed it becomes associated with a new segment identifier. The segment's old identifier becomes available for reuse by CREATE SEGMENT or can be returned from GET NEW SEGMENT IDENTIFIER. When a segment is renamed, the segment attributes associated with the segment are not changed.

3.7 Inquiry

Segment inquiry functions, as defined in clause 6, provide the client with the means to access the information in the Segment Description Table and the Segment State Lists. These description tables and state lists provide information about the capabilities and current segment state of the CGI Virtual Device.

Details about the relationship between the description table or state lists and the corresponding inquiry functions is described in ISO/IEC 9636-1, 5.2.7.

3.8 Picking

Picking is an input mechanism to identify segments interactively and is described in ISO/IEC 9636-5. When segments are picked, the pick values returned contain the segment identifiers and pick identifiers of the picked segments. The PICK IDENTIFIER function sets the value of Pick Identifier entry in the Segment State List. The current Pick Identifier is associated with each graphic object when it is created. The SIMULATE PICK function provides a client with a non-interactive mechanism to identify segments which intersect a designated region of VDC space. This region, termed a pick aperture, is defined by a width and height and is positioned about a point in VDC space. The pick aperture may lie outside of the VDC extent.

This part of ISO/IEC 9636 allows latitude in the implementation of the shape of the pick aperture. The Pick Aperture Shape entry in the Segment Description Table indicates which method is implemented as follows:

- A value of ELLIPSE indicates an elliptical pick aperture, centred on the pick location. The major and minor axes of the ellipse are aligned with the coordinate axes, with the horizontal diameter equal to the specified width and the vertical diameter equal to the specified height. This is the preferred implementation.
- A value of CIRCLE indicates a circular pick aperture, centred on the pick location with the radius equal to the larger of the specified width and height.
- A value of RECTANGLE indicates a rectangular pick aperture of specified width and height centred on the pick location.

A PICK input operation and the SIMULATE PICK function return a list of pick values (each pick value consists of a segment identifier and a pick identifier) in order of display priority. The segments associated with the pick values all satisfy the following conditions:

a) The segment visibility is VISIBLE;

b) The segment detectability is DETECTABLE;

c) The pick aperture, transformed by the current VDC-to-Device Mapping, intersects either the locus or shape of an object in the segment after all concatenated associated transformations have been applied;

d) No part of an object in any other segment with higher pick priority fulfills conditions a through c above.

Conceptually, picking behaves as follows. Each segment in segment storage is examined. Each object of a segment is tested for intersection with the pick aperture. Specifically, the rendered locus and shape of the graphic object are tested for intersection with the image of the pick aperture under the VDC-to-Device Mapping. For graphic objects that intersect this transformed pick aperture, the identifier of the containing segment and the object's associated pick identifier attribute value may be returned, subject to the segment's Pick Priority. The interaction of Pick Priority with picking is described in 3.3.6.

3.9 State restrictions

There is one state variable, the Segment Open State in the Segment State List. Segment Open State may have the value NO, YES, or OVERFLOW. Transitions between these states are described in detail in 3.2.

State restrictions defined for this part of ISO/IEC 9636 are given in table 2. Functions not listed in this table are not restricted.

Table 2 – Segment Open State function restrictions

Function	Restricted Segment Open States
CREATE SEGMENT	YES, OVERFLOW
REOPEN SEGMENT	YES, OVERFLOW
CLOSE SEGMENT	NO
COPY SEGMENT	(Note 1)

NOTE —

1) COPY SEGMENT is not allowed if the source segment to copy is the open segment.

4 Interactions with other parts of ISO/IEC 9636

4.1 Interactions with ISO/IEC 9636-2

4.1.1 INITIALIZE and TERMINATE

The function INITIALIZE

- Clears internal segment storage.
- Restores all values in the Segment State List to default values; in particular, it sets the Segment Open State to NO.
- Clears all error conditions.

There are no state restrictions on the use of INITIALIZE and TERMINATE, i.e. INITIALIZE and TERMINATE may be used at any time (see ISO/IEC 9636-2, 5.2.1 and 5.2.2).

The CGI does not require any change to segment storage as an effect of the function TERMINATE.

4.2 Interactions with ISO/IEC 9636-5

This part of ISO/IEC 9636 provides some functions which are associated with Input concepts in the area of pick input support. In particular, pick identifiers are defined as additional attribute values associated with graphic objects stored in segments. Further control of pick input is provided by the segment attributes Pick Priority, Detectability, and Visibility.

The function GET ADDITIONAL PICK DATA defined in ISO/IEC 9636-5 may be enabled following execution of the function SIMULATE PICK.

4.3 Interactions with ISO/IEC 9636-6

Segments are rendered, explicitly or implicitly, onto the currently selected drawing bitmap, which may be different from the current display bitmap.

All Raster functions are allowed when a segment is open. PIXEL ARRAY does not form a graphic object and is not stored in a segment.

5 Abstract specification of functions

5.1 Introduction

The CGI functions related to segments are defined in this clause.

The functions described in this clause are

- Segment manipulation functions.
- Segment attribute functions.
- Miscellaneous segment functions.

5.1.1 Data types employed

The abstract specifications of functions detail the functions in terms of input and output parameters. The data type of each parameter is selected from a standard set and is identified in the functional specification by a standard abbreviation.

The data types and the abbreviations used in this part of ISO/IEC 9636 are taken from the complete list of data types in ISO/IEC 9636-1, 5.2.10.

5.1.2 Validity of returned information

For all of the functions specified in this clause which solicit a response from the Virtual Device, a response validity flag is returned as INVALID if an error was detected in executing the function. In such cases, the other output parameters are undefined and no meaning should be applied to any of these parameter values.

5.2 Segment manipulation functions

5.2.1 GET NEW SEGMENT IDENTIFIER

Parameters:

Out	response validity	(INVALID, VALID)	E
Out	segment identifier		SN

Effect:

The CGI returns a *segment identifier* different from that of any existing segment. This *segment identifier* may be used as a parameter to create (CREATE SEGMENT) or rename (RENAME SEGMENT) a segment.

If a new segment identifier is not available, a response validity of INVALID is returned.

Errors:

Error identifier:	3:405
Cause:	Maximum number of segment identifiers already in use
Reaction:	Function ignored.

5.2.2 CREATE SEGMENT

Parameters:

In	segment identifier	SN

Effect:

A new segment corresponding to the specified identifier is opened. Subsequent graphic objects are stored in the segment. The Segment Open State in the Segment State List is set to YES. The Identifier of Open Segment entry in the Segment State List is set to the specified *segment identifier*. The *segment identifier* is added to the List of Segment Identifiers in the Segment State List.

The display of graphic objects stored in the segment depends on Visibility, Highlighting, Implicit Segment Regeneration Mode, and Display Priority. If the Visibility attribute of a segment has the value VISIBLE, then each new object is rendered.

Errors:

Error identifier:	3:401
Cause:	A segment with requested segment identifier already exists
Reaction:	Function ignored.
Error identifier:	5:401
Cause:	Function not allowed if Segment Open State is YES or OVERFLOW
Reaction:	Function ignored; graphic objects following the segment overflow and before the segment is closed are rendered as non-retained data.
Error identifier:	6:401
Cause:	Segment storage overflow
Reaction:	Function ignored; graphic objects following the segment overflow and prior to segment closure are rendered as non-retained data.

5.2.3 REOPEN SEGMENT

Parameters:

In	segment identifier	SN

Effect:

The segment corresponding to the specified identifier is reopened. Subsequent graphic objects are appended to the segment. The Segment Open State in the Segment State List is set to YES and the Identifier of Open Segment entry is set to the specified *segment identifier*.

The display of graphic objects added to an existing segment depends on Visibility, Highlighting, Implicit Segment Regeneration Mode, and Display Priority. If the Visibility attribute of a segment has the value VISIBLE, then each new object is rendered.

Errors:

Error identifier:	3:402
Cause:	Identified segment does not exist
Reaction:	Function ignored.
Error identifier:	5:401
Cause:	Function not allowed if Segment Open State is YES or OVERFLOW
Reaction:	Function ignored; graphic objects following the segment overflow and before the segment is closed are rendered as non-retained data.
Error identifier:	6:401
Cause:	Segment storage overflow
Reaction:	Function ignored; graphic objects following the segment overflow and prior to segment closure are rendered as non-retained data.

5.2.4 CLOSE SEGMENT

Parameters:

None

Effect:

The open segment is closed. Subsequent graphic objects are no longer stored in the segment. The Segment Open State in the Segment State List is set to NO and the Identifier of Open Segment becomes undefined.

Errors:

Error identifier: 5:402
Cause: Function not allowed if Segment Open State is NO
Reaction: Function ignored.

5.2.5 COPY SEGMENT

Parameters:

In	source segment identifier		SN
In	copy transformation:		
	scaling and rotation portion		2x2R
	translation portion		2x1VDC
In	segment transformation association	(NO, YES)	E

Effect:

All graphic objects in the identified segment are re-entered into the CGI pipeline. The identified segment is referred to as the source segment. With the possible exception of the associated Segment Transformation, the segment attributes of the source segment are ignored.

If a segment is open when COPY SEGMENT is invoked, the copied graphic objects are stored in the open segment and are rendered with it. The segment attributes of the open segment are unchanged by the COPY SEGMENT function. If no segment is open, the copied graphic objects are rendered immediately as non-retained data.

The INHERITANCE FILTER function allows for control of the attribute values which are used when copying segments. This filter controls whether individual attribute values from the CGI state lists are reapplied to the graphic objects or whether the graphic objects are unaffected by the CGI state list values. (See 5.4.2.)

If the parameter *segment transformation association* is YES, the segment transformation of the source segment is concatenated with the existing transformations associated with each graphic object of the source segment. The *copy transformation* is concatenated after the segment transformation. Clip rectangles are never transformed by the segment transformation.

The *copy transformation* is concatenated with the transformation associated with each graphic object of the source segment before being rendered or copied into an open segment. If the objects in the source segment have previously been copied, any new transformations resulting from the current copy segment operation are concatenated with those already present, in such a way that these new transformations are applied, during the rendering process, after any previously associated ones.

The clipping applied to these graphic objects is the result of the combination of state list and copied clipping information according to the Clip Inheritance Filter. If this filter is set to STATE LIST, the associated clip rectangle and clip indicator attributes are replaced by the values of the Clip Indicator and Clip Rectangle state list entries.

The following assumes that the Clip Inheritance Filter is set to INTERSECTION; that is, the clip rectangle and clip indicator attributes are not replaced by corresponding state list values.

If the clip indicator associated with a graphic object in the source segment is ON, then the *copy transformation* is associated with the clip rectangle. It is not required that all implementations of CGI be able to support transforming the clip rectangle if the scaling and rotation portion of the matrix representing the *copy transformation* has non-zero off-diagonal elements (see below). The Transformed Clip Region Effectiveness entry in the Segment Description Table indicates the level of support for the attempt to apply such an associated transformation to a graphic object (see below).

If the Clip Indicator in the General Attributes and Output Control State List is ON, then the corresponding Clip Rectangle in that state list also affects the associated clip rectangle attribute value of the copied graphic object. If the associated clip indicator attribute is OFF, the Clip Rectangle in the state list is associated with the copied object. If both clip indicators are ON, both clip rectangles are associated with the copied object (i.e. the Clip Rectangle in the state list and the clip rectangle attribute value with its associated copy transformation).

During rendering, the combination by intersection of all the associated clip rectangles is determined and this may yield a convex polygonal region. If the Segment Description Table entry for Transformed Clip Region Effectiveness is FULL, this convex polygonal region will be applied as the effective clip region. If the Transformed Clip Region Effectiveness is CIRCUMSCRIBED, the effective clip region is defined to be the smallest axis-aligned rectangle which circumscribes the convex polygonal region. The new clip indicator attribute associated with the copied graphic object will be ON if either the clip indicator attribute of the object or the Clip Indicator entry in the General Attributes and Output Control State List is ON.

Errors:

Error identifier: 3:402
Cause: Identified segment does not exist
Reaction: Function ignored.

Error identifier: 5:403
Cause: Identified segment is the current open segment
Reaction: Function ignored.

Error identifier: 6:401
Cause: Segment storage overflow
Reaction: Function ignored; graphic objects following the segment overflow and prior to segment closure are rendered as non-retained data.

5.2.6 DELETE SEGMENT

Parameters:

In segment identifier SN

Effect:

The identified segment is deleted from segment storage. The *segment identifier* may be reused as a parameter of the CREATE SEGMENT function. The *segment identifier* is also available to be returned via the GET NEW SEGMENT IDENTIFIER function. The *segment identifier* is removed from the List of Segment Identifiers in the Segment State List.

Display changes resulting from segment deletion are dependent on the Implicit Segment Regeneration Mode and Dynamic Modification Accepted For Segment Deletion flag. (See 3.4.2.)

If the identified segment is the open segment, it is implicitly closed before being deleted. In this case, the Segment Open State in the Segment State List is set to NO and the Identifier of Open Segment becomes undefined.

Errors:

Error identifier: 3:402
Cause: Identified segment does not exist
Reaction: Function ignored.

5.2.7 DELETE ALL SEGMENTS

Parameters:

None

Effect:

All segments are deleted from segment storage. Any segment identifier may be reused as a parameter to the CREATE SEGMENT function and is available to be returned by the GET NEW SEGMENT IDENTIFIER function. The List of Segment Identifiers in the Segment State List will be empty.

Display changes resulting from segment deletion are dependent on the Implicit Segment Regeneration Mode and Dynamic Modification Accepted For Segment Deletion flag. (See 3.4.2.)

If a segment is the open segment, it is implicitly closed before being deleted. In this case, the Segment Open State in the Segment State List is set to NO and the Identifier of Open Segment becomes undefined.

5.2.8 RENAME SEGMENT

Parameters:

In	old segment identifier	SN
In	new segment identifier	SN

Effect:

An existing segment is identified by its *old segment identifier*. The RENAME SEGMENT function replaces the *old segment identifier* with the *new segment identifier*. Future references to the segment should be made with this *new segment identifier*. After RENAME SEGMENT, the *old segment identifier* may be reused as a parameter to the CREATE SEGMENT function. If the *old segment identifier* is the open segment then the Identifier of Open Segment entry in the Segment State List is set to the *new segment identifier*. The *old segment identifier* is also available to be returned by the GET NEW SEGMENT IDENTIFIER function.

Errors:

Error identifier:	3:403
Cause:	Identified old segment does not exist
Reaction:	Function ignored.
Error identifier:	3:404
Cause:	The segment with the requested new segment identifier already exists
Reaction:	Function ignored.

5.2.9 DRAW ALL SEGMENTS

Parameters:

None

Effect:

All existing segments with Visibility attribute set to VISIBLE are rendered, such that segments with higher Display Priority appear to cover overlapping segments of lower Display Priority.

The Regeneration Pending entry in the Segment State List is not affected by this function. Segments are rendered without regard to the Implicit Segment Regeneration Mode. Non-retained data is not erased. An explicit PREPARE DRAWING SURFACE can be performed to remove any non-retained data.

5.2.10 IMPLICIT SEGMENT REGENERATION MODE

Parameters:

In	implicit segment regeneration mode	(SUPPRESSED, UQUM, ALLOWED)	E

Effect:

The Implicit Segment Regeneration Mode entry in the Segment State List is set to the value specified. This function may be issued at any time.

While the mode is ALLOWED, if a change is made for which the Dynamic Modification Accepted For flag is CBS or IRG, an implicit segment regeneration will be performed as a result of the function causing the change. Any non-retained data will be lost.

While the mode is UQUM, if a change is made for which the Dynamic Modification Accepted For flag is CBS, an implementation-dependent quick update method will be used to realize the change. In such cases, the Regeneration Pending entry in the Segment State List is set to YES.

While the mode is SUPPRESSED, if a change is made for which the Dynamic Modification Accepted For flag is CBS or IRG, the Regeneration Pending entry will be set to YES and no implicit regeneration will be performed. However, changes for which the Dynamic Modification Accepted For flag is IMM, will still be reflected in the picture without any regeneration.

If the Regeneration Pending entry in the Segment State List is YES at the time of a transition of the Implicit Segment Regeneration Mode to ALLOWED from either UQUM or SUPPRESSED, a regeneration will occur as a result of the IMPLICIT SEGMENT REGENERATION MODE function and the Regeneration Pending entry in the Segment State List is set to NO.

5.2.11 RESET REGENERATION PENDING

Parameters:

None

Effect:

The Regeneration Pending entry in the Segment State List is set to NO. This has the effect that if the Implicit Segment Regeneration Mode is subsequently changed to ALLOWED, there may be no regeneration even if there exists a discrepancy between the picture described by the contents of segment storage and what is actually rendered on the drawing surface.

5.2.12 PICK IDENTIFIER

Parameters:

In	pick identifier		PN

Effect:

The Pick Identifier entry in the Segment State List is set to the value specified.

A pick identifier is associated with each graphic object, see 3.8 and clause 3 of ISO/IEC 9636-3.

5.3 Segment attribute functions

5.3.1 SEGMENT VISIBILITY

Parameters:

In	segment identifier		SN
In	visibility	(INVISIBLE, VISIBLE)	E

Effect:

The Visibility attribute of the identified segment is set to the value specified.

This function is used to turn a segment's visibility on or off. Any affect of this function on the rendered picture is dependent on the current Implicit Segment Regeneration Mode entry in the Segment State List, and on the Dynamic Modification Accepted For Change of Visibility to VISIBLE and Dynamic Modification Accepted For Change of Visibility to INVISIBLE flags. (See 3.4.2, 3.4.3 and 3.8.)

When the Visibility attribute is set to INVISIBLE, the segment cannot be picked.

Errors:

Error identifier:	3:402
Cause:	Identified segment does not exist
Reaction:	Function ignored.

5.3.2 SEGMENT TRANSFORMATION

Parameters:

In	segment identifier	SN
In	segment transformation:	
	scaling and rotation portion	2x2R
	translation portion	2x1VDC

Effect:

The Segment Transformation attribute of the identified segment is replaced by the specified *segment transformation.*

During rendering of the objects belonging to the identified segment, the segment's Segment Transformation attribute value will be concatenated with the object's associated transformation. The resulting transformation will be applied to the graphic object as described in 3.3.7.

A segment transformation may also be used during the operation COPY SEGMENT (see 3.5).

Any affect of this function on the rendered picture is dependent on the current Implicit Segment Regeneration Mode entry in the Segment State List, and on the Dynamic Modification Accepted For Change of Segment Transformation flag. Refer also to 3.4.2, 3.4.3, and 5.2.5.

Errors:

Error identifier:	3:402
Cause:	Identified segment does not exist
Reaction:	Function ignored.

5.3.3 SEGMENT HIGHLIGHTING

Parameters:

In	segment identifier		SN
In	highlighting	(NORMAL, HIGHLIGHTED)	E

Effect:

The Highlighting attribute of the identified segment is set to the specified value.

Any affect of this function on the rendered picture is dependent on the current Implicit Segment Regeneration Mode entry in the Segment State List, and on the Dynamic Modification Accepted For Change of Highlighting flag. Refer also to 3.4.2 and 3.4.3.

Errors:

Error identifier:	3:402
Cause:	Identified segment does not exist
Reaction:	Function ignored.

5.3.4 SEGMENT DISPLAY PRIORITY

Parameters:

In	segment identifier		SN
In	display priority	(0..n)	I

Effect:

The Display Priority attribute of the identified segment is set to the specified value.

Segments that have the value 0 for Display Priority have the lowest display priority.

Any affect of this function on the rendered picture is dependent on the current Implicit Segment Regeneration Mode entry in the Segment State List, and on the Dynamic Modification Accepted For Change of Display Priority flag. See also 3.3.5, 3.4.2, and 3.4.3.

Errors:

Error identifier: 3:402
Cause: Identified segment does not exist
Reaction: Function ignored.

5.3.5 SEGMENT DETECTABILITY

Parameters:

In	segment identifier		SN
In	detectability	(UNDETECTABLE, DETECTABLE)	E

Effect:

The Detectability attribute of the identified segment is set to the value specified.

A segment is eligible to be picked if its Detectability attribute is set to DETECTABLE. See 3.3.4 and 3.8.

Errors:

Error identifier: 3:402
Cause: Identified segment does not exist
Reaction: Function ignored.

5.3.6 SEGMENT PICK PRIORITY

Parameters:

In	segment identifier		SN
In	pick priority	(0..n)	I

Effect:

The Pick Priority attribute of the identified segment is set to the value specified.

Pick priority does not affect the rendering of segments. It only affects pick input or simulated picking if segments overlap. Segments that have the value 0 for Pick Priority have the lowest pick priority.

See 3.3.6 and 3.8.

Errors:

Error identifier: 3:402
Cause: Identified segment does not exist
Reaction: Function ignored.

5.4 Miscellaneous segment functions

5.4.1 SIMULATE PICK

Parameters:

In	pick point		P
In	pick aperture (height, width)	(>0)	2VDC
In	number of requested pick values	(1..n)	I
Out	response validity	(INVALID, VALID)	E
Out	total number of available pick values	(1..n)	I
Out	list of pick values		nPV

Effect:

This function emulates the behaviour of the PICK input function defined in ISO/IEC 9636-5.

The *pick point* specifies the location at which the SIMULATE PICK function will search for pickable segments. The *pick point* may be outside of the VDC Extent. The *pick aperture* specifies a VDC region around the *pick point*. See 3.8 for an explanation of the interpretation of the *pick aperture* parameter and the conceptual picking process. Any segment which intersects the pick aperture region is eligible to be picked.

The returned *list of pick values* contains distinct segment/pick identifier pairs, identifying picked segments with the identically highest pick priority. The list is returned in order of decreasing display priority.

The maximum length of the list is defined in the Segment Description Table. but the client may restrict its length by the parameter *number of requested pick values*. If the returned *total number of available pick values* is greater than the length of the list, the remaining pick values may be retrieved by use of the function GET ADDITIONAL PICK DATA, defined in ISO/IEC 9636-5.

NOTE – An implementation that supports this function is expected also to support GET ADDITIONAL PICK DATA.

Errors:

Error identifier:	3:406
Cause:	Too many pick values
Reaction:	Excess pick values discarded.

5.4.2 INHERITANCE FILTER

Parameters:

In	filter selection list	*(See below)*	nE
In	selection setting	(STATE LIST, SEGMENT)	E

Effect:

The List of Attribute Designators in the Segment State List is modified for those attributes and ASFs indicated in the given *filter selection list*. The INHERITANCE FILTER function is used in conjunction with the COPY SEGMENT function (see 5.2.5).

For those attributes or ASFs whose *selection setting* is STATE LIST, the corresponding attribute or ASF values are replaced, in those graphic objects to which they may be associated, by corresponding state list entries during a copy segment operation.

For those attributes or ASFs whose *selection setting* is SEGMENT, the corresponding attribute or ASF values are unaffected, in all graphic objects to which they may be associated, during the copy segment operation.

Symbolic names for elements of the *filter selection list* parameter appear in tables 3 and 4. Table 3 lists symbolic names for ASFs; table 4 lists symbolic names for attributes. It is possible to specify either individual names (as listed in the second column of each table), or group names (as listed in the first column of each table). A group name represents the collection of associated individual names.

The exact enumeration is given in annex A (Formal grammar). It is ordered in the following way:

a) all Individual ASF Names (in order of table 3);

b) all Individual Attribute Names (in order of table 4);

c) all ASF Group Names (in order of table 3);

d) all Attribute Group Names (in order of table 4).

Table 3 – Inheritance Filter Selection Names for Aspect Source Flags

ASF Group Name	Individual ASF Name
LINE ASFS	LINE TYPE ASF LINE WIDTH ASF LINE COLOUR ASF
MARKER ASFS	MARKER TYPE ASF MARKER SIZE ASF MARKER COLOUR ASF
TEXT ASFS	TEXT FONT INDEX ASF TEXT PRECISION ASF CHARACTER EXPANSION FACTOR ASF CHARACTER SPACING ASF TEXT COLOUR ASF
FILL ASFS	INTERIOR STYLE ASF FILL COLOUR ASF HATCH INDEX ASF PATTERN INDEX ASF FILL BITMAP ASF
EDGE ASFS	EDGE TYPE ASF EDGE WIDTH ASF EDGE COLOUR ASF EDGE VISIBILITY ASF
ALL ASFS	All aspect source flags

Table 4 – Inheritance Filter Selection Names for Attributes

Attribute Group Name	Individual Attribute Name
LINE ATTRIBUTES	LINE BUNDLE INDEX LINE TYPE LINE WIDTH LINE COLOUR LINE CLIPPING MODE
MARKER ATTRIBUTES	MARKER BUNDLE INDEX MARKER TYPE MARKER SIZE MARKER COLOUR MARKER CLIPPING MODE
LOCAL TEXT ATTRIBUTES	TEXT BUNDLE INDEX TEXT FONT INDEX TEXT COLOUR CHARACTER EXPANSION FACTOR CHARACTER SPACING CHARACTER HEIGHT
GLOBAL TEXT ATTRIBUTES	CHARACTER ORIENTATION TEXT PRECISION TEXT PATH TEXT ALIGNMENT

Table 4 – Inheritance Filter Selection Names for Attributes – (concluded)

Attribute Group Name	Individual Attribute Name
FILL ATTRIBUTES	FILL BUNDLE INDEX INTERIOR STYLE FILL COLOUR HATCH INDEX PATTERN INDEX FILL BITMAP
EDGE ATTRIBUTES	EDGE BUNDLE INDEX EDGE TYPE EDGE WIDTH EDGE COLOUR EDGE VISIBILITY EDGE CLIPPING MODE
PATTERN ATTRIBUTES	FILL REFERENCE POINT PATTERN SIZE
OUTPUT CONTROL	AUXILIARY COLOUR TRANSPARENCY DRAWING MODE
PICK ATTRIBUTES	PICK IDENTIFIER
ALL ATTRIBUTES	All attributes controlled
ALL	All attributes and ASFs controlled

NOTE –

1) Line Width, Edge Width and Marker Size are implicitly accompanied by the corresponding Size Specification Mode which was current at the time the size value was set in its state list. Similarly, for the colour attributes and corresponding colour selection mode.

5.4.3 CLIPPING INHERITANCE

Parameters:

In selection setting (STATE LIST, INTERSECTION) E

Effect:

The Clip Inheritance Filter in the Segment State List is set to the specified *selection setting*. The CLIPPING INHERITANCE function is used in conjunction with the COPY SEGMENT function.

See 5.2.5 for a description of the effect of this state list entry.

6 Segment inquiry functions

6.1 Introduction

This clause describes the abstract functional specification of the Segment Inquiry functions of the CGI.

The abstract names of these functions start with INQUIRE and will only return the values of one or more entries in a description table or state list.

See ISO/IEC 9636-1, 5.2.7.

6.1.1 Data types employed

The abstract specifications of functions detail the functions in terms of input and output parameters. The data type of each parameter is selected from a standard set and is identified in the functional specification by a standard abbreviation. Both the data types and the abbreviations are taken from the complete list of data types given in ISO/IEC 9636-1, 5.2.10.

6.1.2 Validity of returned information

For all the inquiry functions specified in this clause, if any of the inquired information is available, the response validity flag is returned as VALID and the values specified in the output parameters are returned. In the case of a VALID response, if there is any possibility that any of the individual parameters within the response are not valid, then the response itself will contain additional (always valid) parameters which indicate which of the other returned parameters are valid.

If the inquired information is not available or the inquiry function is unsupported, the response validity flag is returned as INVALID and the specified output parameters are undefined. No other meaning should be applied to these other output parameters.

6.2 Segment description table

6.2.1 INQUIRE SEGMENT CAPABILITY

Parameters:

Out	response validity	(INVALID, VALID)	E
Out	number of supported display priorities	(-1, 0..n)	I
Out	number of supported pick priorities	(-1, 0..n)	I
Out	maximum number of simultaneously existing segments	(-1, 0..n)	I
Out	maximum length of simulate pick list	(-1, 0..n)	I
Out	dynamic modification accepted for adding objects to open segment	(IRG, CBS, IMM)	E
Out	dynamic modification accepted for segment deletion	(IRG, CBS, IMM)	E
Out	dynamic modification accepted for change of segment transformation	(IRG, CBS, IMM)	E
Out	dynamic modification accepted for change of display priority	(IRG, CBS, IMM)	E
Out	dynamic modification accepted for change of visibility to invisible	(IRG, CBS, IMM)	E
Out	dynamic modification accepted for change of visibility to visible	(IRG, CBS, IMM)	E
Out	dynamic modification accepted for change of highlighting	(IRG, CBS, IMM)	E
Out	effective relative display priority	(OTHER, TIME ORDER-REOPEN, TIME ORDER)	E
Out	pick aperture shape	(RECTANGLE, CIRCLE, ELLIPSE)	E
Out	transformed clip region effectiveness	(NONE, CIRCUMSCRIBED, FULL)	E

Effect:

See 6.1.2.

6.3 Segment state list

6.3.1 INQUIRE SEGMENT STATE

Parameters:

Out	response validity	(INVALID, VALID)	E
Out	segment open state	(NO, YES, OVERFLOW)	E
Out	identifier of open segment		SN
Out	pick identifier		PN
Out	implicit segment regeneration mode	(SUPPRESSED, UQUM, ALLOWED)	E
Out	regeneration pending	(NO, YES)	E

Effect:

See 6.1.2.

6.3.2 INQUIRE LIST OF INHERITANCE FILTER SETTINGS

Parameters:

In	number of list elements requested	(0..n)	I
In	index (within list) of first element to return	(1..n)	I
Out	response validity	(INVALID, VALID)	E
Out	total number of list elements present	(0..n)	I
Out	list of attribute designators	(*(See tables 3, 4)* (STATE LIST, SEGMENT))	n[E,E]

Effect:

See 6.1.2.

See 5.4.2 and annex A for the enumeration values of the filter selector component of the *list of filter attribute designators* parameter.

6.3.3 INQUIRE CLIPPING INHERITANCE

Parameters:

Out	response validity	(INVALID, VALID)	E
Out	clip inheritance filter	(STATE LIST, INTERSECTION)	E

Effect:

See 6.1.2.

6.3.4 INQUIRE LIST OF SEGMENT IDENTIFIERS IN USE

Parameters:

In	number of list elements requested	(0..n)	I
In	index (within list) of first element to return	(1..n)	I
Out	response validity	(INVALID, VALID)	E
Out	total number of list elements present	(0..n)	I
Out	list of segment identifiers		nSN

Effect:

See 6.1.2.

6.4 Individual segment state list

6.4.1 INQUIRE INDIVIDUAL SEGMENT STATE

Parameters:

In	segment identifier		SN
Out	response validity	(INVALID, VALID)	E
Out	segment transformation:		
	scaling and rotation portion		2x2R
	translation portion		2x1VDC
Out	visibility	(INVISIBLE, VISIBLE)	E
Out	highlighting	(NORMAL, HIGHLIGHTED)	E
Out	display priority	(0..n)	I
Out	detectability	(UNDETECTABLE, DETECTABLE)	E
Out	pick priority	(0..n)	I

Effect:

See 6.1.2.

Errors:

Error identifier:	3:402
Cause:	Identified segment does not exist
Reaction:	Function ignored.

7 CGI description tables and state lists

This clause contains the description tables and state lists which define the portion of the Virtual Device relating to the segment function groups.

If this part of ISO/IEC 9636 allows latitude for certain description table entries, the preferred entries are designated by the entry being underlined.

The information in the description tables and state lists is available to a client by means of inquiry functions. The abstract specification of these functions are found in clause 6.

7.1 Description tables

Table 5 – Segment Description Table

Entry	Possible Values	Data Type
Number of Supported Display Priorities	(-1,0..n) (Note 1)	I
Number of Supported Pick Priorities	(-1,0..n) (Note 1)	I
Maximum Number of Simultaneously Existing Segments	(-1,0..n) (Note 1)	I
Maximum Length of Simulate Pick List	(-1,0..n) (Note 1)	I
Dynamic Modification Accepted For	(IRG, CBS, IMM)	E
Adding Objects to Open Segment		
Segment Deletion	(IRG, CBS, IMM)	E
Change of Segment Transformation	(IRG, CBS, IMM)	E
Change of Display Priority	(IRG, CBS, IMM)	E
Change of Visibility to INVISIBLE	(IRG, CBS, IMM)	E
Change of Visibility to VISIBLE	(IRG, CBS, IMM)	E
Change of Highlighting	(IRG, CBS, IMM)	E
Effective Relative Display Priority (Note 2)	(OTHER, TIME ORDER-REOPEN, TIME ORDER)	E
Pick Aperture Shape	(RECTANGLE, CIRCLE, ELLIPSE)	E
Transformed Clip Region Effectiveness (Note 3)	(NONE, CIRCUMSCRIBED, FULL)	E

NOTES

1) For entries denoting the maximum number of supported entities, zero indicates no support, -1 indicates either no restriction on the number supported or support is limited only by the available memory (for implementations using dynamic memory allocation).

2) The effective relative display priority for segments with equal display priority.

3) The value NONE should appear if there is no support for object clipping. This can also be determined by invoking LOOKUP FUNCTION SUPPORT.

7.2 State lists

7.2.1 Segment state list

Table 6 – Segment State List

Entry	Possible Values	Data Type	Default
Segment State:			
Segment Open State	(NO, YES, OVERFLOW)	E	NO
Identifier of Open Segment	(0..n) (Note 1)	SN	*undef.*
Pick Identifier	(0..n)	PN	0
Implicit Segment Regeneration Mode	(SUPPRESSED, UQUM, ALLOWED)	E	SUPPRESSED
Regeneration Pending	(NO, YES)	E	NO
Inheritance Filter Settings:			
List of Attribute Designators containing:			
Attribute or ASF Name	(Note 2)	E	empty
Selection Setting	(STATE LIST, SEGMENT)	E	SEGMENT
Clipping Inheritance:			
Clip Inheritance Filter	(STATE LIST, INTERSECTION)	E	STATE LIST
Segment Identifiers In Use:			
List of Segment Identifiers	(0..n)	nSN	empty

NOTES

1) When the Segment Open State is NO, the Identifier of Open Segment is undefined (the default condition).

2) See the abstract specification of the INHERITANCE FILTER function for the range of values of the filter attribute designator.

7.2.2 Individual segment state list

An Individual Segment State List exists for each stored segment and for the open segment, if one exists. This table defines a dynamic state list. The presence of an instance of this state list in an implementation may be determined from the presence of its Segment Identifier in the Segment State List.

Table 7 – Individual Segment State List

Entry	Possible Values	Data Type	Default
Segment Transformation		(2x2R, 2x1VDC)	*identity*
Visibility	(INVISIBLE, VISIBLE)	E	VISIBLE
Highlighting	(NORMAL, HIGHLIGHTED)	E	NORMAL
Display Priority	(0..n)	I	0
Detectability	(UNDETECTABLE, DETECTABLE)	E	UNDETECTABLE
Pick Priority	(0..n)	I	0

Annex A

(normative)

Formal grammar of the functional specification

A.1 Introduction

This grammar is a formal definition of the Segment portion of standard CGI syntax. It shows all productions regardless of the encoding scheme. The terminal symbols correspond to the CGI basic abstract data types. Encoding and representation details of these can be found in ISO/IEC 9637.

A.2 Notation used

<symbol>	- nonterminal
<SYMBOL>	- terminal
<symbol>*	- 0 or more occurrences
<symbol>+	- 1 or more occurrences
<symbol>o	- 0 or 1 occurrences
<symbol>(n)	- exactly n occurrences; n=2,3,...
<symbol-1> ::= <symbol-2>	- symbol-1 has the syntax of symbol-2
<symbol-1> | <symbol-2>	- symbol-1 or alternatively symbol-2
<symbol: meaning>	- symbol with the stated meaning
comment	- explanation of a symbol or a production
returned: <symbol>*	- output parameter(s)

A.3 Detailed grammar

```
<segment operations>::=   <segment manipulation function>
                        | <segment attribute function>
                        | <miscellaneous segment function>
                        | <segment inquiry function>
```

A.3.1 Segment manipulation functions

```
<segment manipulation function>
                    ::=   <get new segment identifier function>
                        | <create segment function>
                        | <reopen segment function>
                        | <close segment function>
                        | <copy segment function>
                        | <delete segment function>
                        | <delete all segments function>
                        | <rename segment function>
                        | <draw all segments function>
                        | <implicit segment regeneration mode function>
                        | <reset regeneration pending function>
                        | <pick identifier function>
```

Formal grammar of the functional specification **Detailed grammar**

<get new segment identifier function>
::= <GET NEW SEGMENT IDENTIFIER>
returned:
<response validity>
<segment identifier>

<segment identifier> ::= <segment name>

<segment name> ::= <BDT_CSN_VALUE>

<response validity> ::= <validity flag>

<validity flag: enumerated>
::= <INVALID> | <VALID>

<create segment function>
::= <CREATE SEGMENT>
<segment identifier>

<reopen segment function>
::= <REOPEN SEGMENT>
<segment identifier>

<close segment function>
::= <CLOSE SEGMENT>

<copy segment function>
::= <COPY SEGMENT>
<segment identifier>
<transformation: copy transformation>
<yes no flag: segment transformation association>

<yes no flag: enumerated>
::= <NO> | <YES>

<delete segment function>
::= <DELETE SEGMENT>
<segment identifier>

<delete all segments function>
::= <DELETE ALL SEGMENTS>

<rename segment function>
::= <RENAME SEGMENT>
<segment identifier: old segment>
<segment identifier: new segment>

<draw all segments function>
::= <DRAW ALL SEGMENTS>

<implicit segment regeneration mode function>
::= <IMPLICIT SEGMENT REGENERATION MODE>
<implicit segment regeneration mode>

<implicit segment regeneration mode: enumerated>
::= <SUPPRESSED> | <UQUM> | <ALLOWED>

<reset regeneration pending function>
::= <RESET REGENERATION PENDING>

<pick identifier function>
::= <PICK IDENTIFIER>
<pick identifier>

<pick identifier> ::= <pick name>

<pick name> ::= <BDT_CSN_VALUE>

A.3.2 Segment attribute functions

<segment attribute function>
::= <SEGMENT TRANSFORMATION>
<segment identifier>
<transformation: segment transformation>
| <SEGMENT VISIBILITY>
<segment identifier>
<visibility>
| <SEGMENT HIGHLIGHTING>
<segment identifier>
<highlighting>
| <SEGMENT DISPLAY PRIORITY>
<segment identifier>
<display priority>
| <SEGMENT DETECTABILITY>
<segment identifier>
<detectability>
| <SEGMENT PICK PRIORITY>
<segment identifier>
<pick priority>

<transformation> ::= <BDT_2x2_MATRIX_OF_REAL>
<BDT_2x1_MATRIX_OF_VDC>

<visibility: enumerated>
::= <INVISIBLE> | <VISIBLE>

<highlighting: enumerated>
::= <NORMAL> | <HIGHLIGHTED>

<display priority> ::= <BDT_INTEGER>

<detectability: enumerated>
::= <UNDETECTABLE> | <DETECTABLE>

<pick priority> ::= <BDT_INTEGER>

A.3.3 Miscellaneous segment functions

<miscellaneous segment function>
::= <simulate pick function>
| <inheritance filter function>
| <clipping inheritance function>

<simulate pick function>
::= <SIMULATE PICK>
<point: pick point>
<vdc value: pick aperture height, width>(2)
<BDT_INTEGER: number of requested pick values>
returned:
<response validity>
<BDT_INTEGER: total number of available pick values>
<pick values>*

<point> ::= <vdc value>(2)

<vdc value> ::= <BDT_REAL> | <BDT_INTEGER>

Formal grammar of the functional specification **Detailed grammar**

<pick value> ::= <segment identifier>
<pick identifier>

<inheritance filter function>
::= <INHERITANCE FILTER>
<filter selection list>*
<selection setting>

<clipping inheritance function>
::= <CLIPPING INHERITANCE>
<clip inheritance filter>

<filter selection list> ::= <asf name>
<attribute name>
<asf group>
<attribute group>

<asf name: enumerated>
::= <LINE TYPE ASF>
| <LINE WIDTH ASF>
| <LINE COLOUR ASF>
| <MARKER TYPE ASF>
| <MARKER SIZE ASF>
| <MARKER COLOUR ASF>
| <TEXT FONT INDEX ASF>
| <TEXT PRECISION ASF>
| <CHARACTER EXPANSION FACTOR ASF>
| <CHARACTER SPACING ASF>
| <TEXT COLOUR ASF>
| <INTERIOR STYLE ASF>
| <FILL COLOUR ASF>
| <HATCH INDEX ASF>
| <PATTERN INDEX ASF>
| <FILL BITMAP ASF>
| <EDGE TYPE ASF>
| <EDGE WIDTH ASF>
| <EDGE COLOUR ASF>
| <EDGE VISIBILITY ASF>

<attribute name: enumerated>
::= <LINE BUNDLE INDEX>
| <LINE TYPE>
| <LINE WIDTH>
| <LINE COLOUR>
| <LINE CLIPPING MODE>
| <MARKER BUNDLE INDEX>
| <MARKER TYPE>
| <MARKER SIZE>
| <MARKER COLOUR>
| <MARKER CLIPPING MODE>
| <TEXT BUNDLE INDEX>
| <TEXT FONT INDEX>
| <TEXT COLOUR>
| <CHARACTER EXPANSION FACTOR>
| <CHARACTER SPACING>
| <CHARACTER HEIGHT>
| <CHARACTER ORIENTATION>
| <TEXT PRECISION>
| <TEXT PATH>